

Der Begriff der Information in der Informatik

Günter Hotz

Einleitung

Informatik wird oft als Lehre von der Informationsverarbeitung bezeichnet, was in der Bezeichnung unseres Faches durch das Wort "Informatik" ja auch zum Ausdruck gebracht wird. Also sollte der Informationsbegriff in dieser Disziplin eine ausgezeichnete Rolle spielen. Aber für einen Fisch ist Wasser so alltäglich, daß er vieles im Wasser wahrnimmt, aber daß Wasser flüssig ist, wohl zuletzt als bemerkenswert empfindet.

Als Herr Hackl mir die Idee dieses Symposiums schilderte und mir nahelegte, über mein Thema etwas dazu beizutragen, habe ich spontan zugesagt, bin aber, je länger ich mich mit der Rolle des Informationsbegriffes in der Informatik auseinandersetze, in umso größere Verlegenheit geraten.

Um die Verarbeitung von Informationen, sei es in Gestalt von Programmen, sei es in Form von Daten, geht es stets in der Informatik. Programme sollen die Informationen, die sie tragen, für den Leser deutlich zum Ausdruck bringen, und das in unmißverständlicher Weise. Informationen müssen in Speichern so abgelegt werden, daß sie wiedergefunden werden und zwar speicherplatzsparend und zeitsparend. Informationen können mittels verschiedener Trägern ausgetauscht werden, sie können akustischer oder graphischer Natur sein oder, wie heute meist, lineare Zeichenfolgen.

Zur Deutlichkeit der Programme gehört es, daß die Programmiersprache in Übereinstimmung ist mit dem, was sich ein naiver Leser unter der Bedeutung der Programme vorstellt, und die Programme müssen von einem Compiler mit vertretbarem Aufwand interpretiert werden können. Allerdings gibt es den naiven Leser nicht und es ist oft schwer, sich für ein bestimmtes Konzept zu entscheiden.

Die Frage, wie eine bestimmte Struktur interpretiert werden sollte, ist häufig Gegenstand von mathematischen Untersuchungen, die abklären, in welchen Beziehungen verschiedene Interpretationen zueinander stehen. Es sei in diesem Zusammenhang an die vielen nur zum Teil fruchtbaren Diskussionen über die "richtigen" Konzepte der Parameterübergaben bei Prozeduren und die Interpretation der rekursiven Prozeduren erinnert.

Unser Thema hat eine solche Breite, daß wir uns im Rahmen dieses Vortrages auf bestimmte Aspekte beschränken müssen, und ich bitte Sie um Nachsicht, wenn ich mich hierbei solchen Seiten des Themas zuwende, die meinen persönlichen Erfahrungen am nächsten liegen. Hierzu gehören die Anwendungen des Informationsbegriffes in der Fassung von Shannon und der von Kolmogoroff, Chaitin, Martin-Löf, Schnorr und anderen entwickelte Dualismus zwischen Zufälligkeit und Berechenbarkeit. Wir gehen auf einige dieser Anwendungen ein. Die Relevanz dieser Ideen auch für die oben angesprochenen Fragen der Deutlichkeit von Programmen wird durch Experimente nahegelegt, auf die ich am Ende dieses Beitrages zu sprechen komme.

Zwei Ansätze zur Fassung des Informationsbegriffes

Die Frage nach dem Informationsgehalt von Nachrichten stellt sich in informaler Weise wohl schon seit eh und je, denn Weitschweifigkeit und Dichtung hat es so lange gegeben als uns Nachrichten durch die Geschichte überliefert werden. Sehr konkret stellt sich die Frage, wenn wir möglichst viel Information pro Zeiteinheit transportieren wollen. Vergleicht man das Problem des Messens von Information einer Nachricht mit dem Problem, einen Körper durch numerische Angaben zu beschreiben, dann sehen wir, daß solche Angaben im allgemeinen nicht erschöpfend sein werden und daß uns dies auch gar nicht interessiert. Die Masse eines Körpers gibt z.B. eine Information darüber, wie sich der Körper beim Werfen im luftleeren Raum verhält, aber z.B. nur wenig Information darüber, ob der Körper eine Katze ist. So hat Shannon die Frage nach einem Maß für Information allein unter dem Gesichtspunkt des Informations-Transportes unter Voraussetzungen über das statistische Verhalten von Informationsquellen gestellt. Das Verständnis der Information spielt hierbei keine Rolle, sondern nur ihre ein-eindeutige Kodierung.

Im Gegensatz hierzu spielt die Kodierung in der Informatik eine Rolle. Informationen müssen verständlich sein, das heißt, sie müssen sich ohne zu großen Aufwand und ohne zu große Zeitverzögerungen aus ihren Kodierungen entnehmen lassen. Das gewünschte sequentielle Verständnis (keine großen Zeitverzögerungen) bedingt gewisse syntaktische Formen für die Nachrichten. So kommen hier im Vergleich zur Nachrichtenübertragung zusätzlich Interpretationsfragen und syntaktische Fragen ins Spiel.

Komplexitätsfragen spielen hier als auch bei der Nachrichtenübertragung eine wichtige Rolle. Die im Verhältnis zu Problemen der Nachrichtenübertragung neuen Gesichtspunkte sollten zu wesentlichen Veränderungen einer Informationstheorie führen. Ist die Nachrichtentheorie allein auf wahrscheinlichkeitstheoretische Gesichtspunkte gegründet, so muß in der Informatik auch das "Verständnis" der Nachrichten mit einbezogen werden.

Nun, wie sich zeigt, erhält man einen Zugang zu einer solchen Theorie schon dadurch, daß man versucht, dem Begriff der zufälligen Folgen eine mehr konstruktive Fassung zu geben, als dies gewöhnlich geschieht.

Kolmogoroff hatte die Versuche von v. Mises, dem Begriff der zufälligen Folgen eine bessere intuitive Begründung zu geben, durch seinen axiomatischen Ansatz beendet. Mit der Theorie der Berechenbarkeit vertraut geworden, greift er dieses Problem in den sechziger Jahren jedoch wieder auf und schlägt eine Fassung des Begriffes auf der Basis der universellen Rechenmaschinen vor, die von M. Löf, Salomonoff, C.P. Schnorr und Chaitin zu einer Theorie ausgearbeitet wurden. Herr Schnorr brachte auf meine Anregung hin die bekannten Komplexitätshierarchien ins Spiel und erhielt so eine Differenzierung des Begriffes der zufälligen Folgen. Berechenbare Folgen hoher Komplexität mögen vom Standpunkt einer niederen Komplexitätsklasse aus gesehen als zufällig erscheinen. Nun, wenn dies also so ist, dann sollten umgekehrt Maße für Zufälligkeit, wie sie z.B. die Entropie darstellt, auch dazu dienen können, die Komplexität von Folgen abzuschätzen. Entropie und Information sind eng verwandte Begriffe. Wir wollen hier die beiden Informationsbegriffe, den der Nachrichtentheorie und die mathematische Fassung des Begriffes im Rahmen der Theorie der Berechenbarkeit, skizzenhaft entwickeln und drei wichtige Anwendungen dieser Konzepte in der Informatik schildern.

Der Shannonsche Informationsbegriff

Sei $X = \{x_1, \dots, x_n\}$ eine Menge von Alphabetzeichen, mittels denen wir unsere Nachrichten $(x_{i_1}, \dots, x_{i_m})$ mitteilen. Wir wollen annehmen, daß das Zeichen x_i mit der Wahrscheinlichkeit $p(x_i)$ auftritt und die Wahrscheinlichkeit für die Nachricht $w = (x_{i_1}, \dots, x_{i_m})$ der Länge m durch $p(w) = p(x_{i_1}) \cdots p(x_{i_m})$ gegeben ist. (X, p) bezeichnet Shannon als Nachrichtenquelle, genauer als eine Nachrichtenquelle ohne Gedächtnis. Will man Nachrichten w über einen Kanal übertragen, der nur über das Alphabet $B = \{0, 1\}$ verfügt, dann muß man eine Kodierung C der Menge der möglichen Nachrichten

$$X^+ = \{(x_{i_1}, \dots, x_{i_m}) \mid m \in \mathbb{N}\}$$

in die Menge B^+ der Wörter über B vornehmen, so daß sich w aus der Kodierung $C(w)$ wiedergewinnen läßt. Es ist offensichtlich von technischem Interesse, C so auszuwählen, daß man im Mittel möglichst viele Nachrichten pro Zeiteinheit übertragen kann, das heißt, C so zu wählen,

$$\frac{1}{m} \sum_{|w|=m} p(w) \cdot |C(w)|$$

ein Minimum wird. Hierin ist $|w|$ die Länge von w . Diese Forderung betrifft den Inhalt der Nachricht so wenig, wie die Form eines Körpers dessen Masse berührt.

Es stellt sich die Frage, ob man aus (X, p) eine Funktion berechnen kann, die Auskunft über die im Sinne unserer Definition der mittleren Länge optimalen Codierungen gibt.

Es sei $I(x_i)$ eine Zahl, die in irgendeiner Weise den Informationsgehalt von x_i angibt. Dann kann man sagen, daß

$$H(X, p) = \sum_{i=1}^n p(x_i) I(x_i)$$

die mittlere Informationsmenge ist, die die Quelle pro Zeichen produziert. Für $H(X, p)$ schreiben wir auch $H(p) = H(p(x_1), \dots, p(x_n))$ oder kürzer $H(p_1, \dots, p_n)$. Es ist natürlich, folgende Forderungen an H zu stellen :

- (1) $H(p) \geq 0$,
- (2) $H(p)$ ist eine stetige Funktion,
- (3) $H(p_1, \dots, p_n)$ ist invariant gegenüber den Permutationen der p_i ,
- (4) $H(\frac{1}{2}, \frac{1}{2}) = 1$.
- (5) $H(p_1, \dots, p_{n-1}, q_1, q_2) = H(p_1, \dots, p_{n-1}, p_n) + p_n H(q'_1, q'_2)$
für $p_n = q_1 + q_2$, $q'_1 = \frac{q_1}{p_n}$, $q'_2 = \frac{q_2}{p_n}$.

Die Forderung (5) besagt das folgende: Sind wir nicht in der Lage, die Zeichen x_n und x_{n+1} zu unterscheiden, dann ist der mittlere Informationsgehalt der Quelle $H(p_1, \dots, p_n)$. Schauen wir uns aber etwa die Zeichen mit einem Vergrößerungsglas

an, dann entdecken wir, daß x_n und x_{n+1} verschieden sind. Das heißt, daß das vermeintlich einzelne Zeichen $[x_n, x_{n+1}]$ eine Informationsquelle ist, dessen Zeichen mit den Wahrscheinlichkeiten q'_1 und q'_2 auftreten. Das heißt, daß $[x_1, x_{n+1}]$ allein die mittlere Information $H(q'_1, q'_2)$ erzeugt. Diese Informationsquelle tritt aber nur mit der Wahrscheinlichkeit p_n in Tätigkeit, so daß sich die Relation (5) ergibt.

Aus (1) bis (5) läßt sich H eindeutig bestimmen, und zwar gilt

$$H(p_1, \dots, p_n) = - \sum_{i=1}^n p_i \log p_i.$$

Hierin ist $p \log p = 0$ zu setzen für $p = 0$, und \log ist der Logarithmus zur Basis 2.

Die heuristische Begründung für $H(p)$ als mittleren Informationsgehalt der Quelle erhält ihre Bestätigung durch den folgenden Sachverhalt:

Es gibt zu der Quelle (X, p) und $n \in \mathbb{N}$ eine Kodierung c_n der Nachrichten w als Wörter der Länge n über X in binäre Folgen $c_n(w)$, so daß gilt

$$H(p) \leq \frac{1}{n} \sum_{|w|=n} p(w) \cdot |c_n(w)| \leq H(p) + \frac{1}{n}.$$

Die linke Ungleichung gilt darüber hinaus für jede Kodierung. Man kann also hinsichtlich der Frage nach der erforderlichen Zeit für die Übertragung der Nachrichten von (X, p) die Größe $H(p)$ als den mittleren Informationsgehalt pro Zeichen der Quelle (X, p) bezeichnen.

Bevor wir auf Anwendungen dieses Informationsbegriffes in der Informatik eingehen, schildern wir eine zweite Fassung dieses Begriffes, die sich aus dem Komplexitätsbegriff der Informatik ableitet.

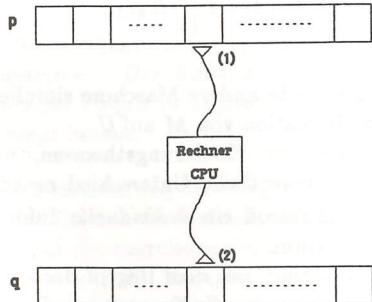
Programmkomplexität und Information

Betrachten wir die unendliche Folge

$$0101 \dots 01 \dots,$$

so genügen uns wenige Buchstaben, wie wir sehen, diese Folge zu beschreiben. Man kann leicht ein Programm für eine Rechenmaschine schreiben, die diese Folge so lange ausdrückt, wie sie funktioniert. Die Länge eines solchen Programmes wird natürlich von der speziellen Maschine abhängen, so daß man also verschiedenen Maschinen i. A. verschiedene viel Information mitteilen muß, damit sie diese Folge ausgeben. Wir versuchen hier also einen anderen Zugang zum Begriff der Information zu gewinnen, indem wir nicht von Konzepten der Wahrscheinlichkeitstheorie ausgehen, sondern von der Idee der Kodierung. Wir drehen also im gewissen Sinne die Betrachtung um: Der Kodierungssatz der Shannonschen Theorie diente uns dort zur Bestätigung des 'Informations'-Konzeptes. Hier starten wir mit dem Konzept der Kodierung. Um zu mathematisch präzisen Aussagen zu kommen, beschreiben wir zunächst den Mechanismus, der den Kodierungen von Folgen durch Programme zugrunde liegen soll.

Es sei M eine abstrakte Rechenmaschine, die ein Eingabeband besitzt, auf dem das Programm p steht, die eine Steuereinheit besitzt, die man sich als ganz gewöhnliche Rechenmaschine vorstellen darf, und einen Rechenspeicher, der stets zwar endlich ist, aber nach Bedarf verlängert werden kann.



Die momentane Inschrift des Rechenspeichers sei q . Im Anfangszustand stehen der Lesekopf (1) und der Lese-Schreib-Kopf (2) am linken Ende der Bänder. In einem Rechenschritt darf ein Kopf stets nur um höchstens ein Feld versetzt werden. Gelesen bzw. geschrieben wird stets nur auf die Felder, auf denen ein Kopf steht. Der Zeichenvorrat ist endlich, und der Speicher der CPU ebenfalls. Die Berechnung ist beendet, wenn der Lesekopf auf dem letzten Feld von p steht. Unser Maschinenmodell ist das im wesentlichen erstmals von Turing angegebene Modell einer Rechenmaschine und wird nach ihm benannt.

Erzeugt unsere Maschine aus dem Programm p und der bereits im Rechenspeicher stehenden Inschrift q die Ausgabe s , dann schreiben wir

$$s = M(p, q).$$

Nun ordnen wir die Programme in ihrer lexikographischen Reihenfolge (\leq) an und definieren

s^* ist das erste Programm, das mit leerem Rechenspeicher ($q = \epsilon$) gestartet, die Ausgabe s erzeugt.

Wir haben also

$$s = M(p, \epsilon) \Rightarrow s^* \leq p.$$

Nun setzen wir weiter

$$I_M(s) = |s^*|.$$

$I_M(s)$ ist also die Länge des ersten Programmes und damit auch die Länge eines kürzesten Programmes, das s mittels M auf das zunächst leere Rechenband schreibt.

Wir bezeichnen $I_M(s)$ als die *Kolmogoroff-Komplexität* von s bezüglich M . $I_M(s)$ ist also ein Maß für die Information, die M benötigt, um s zu erzeugen.

Diese Definition hängt für Folgen s_1, s_2, \dots mit $|s_i| \rightarrow \infty$ nur unwesentlich von M ab. Es gilt nämlich der Satz von Kolmogoroff:

Für je zwei universelle Maschinen M und U gibt es eine Konstante $K_{M,U}$, so daß gilt:

$$\frac{1}{|s|} I_M(s) \leq \frac{1}{|s|} I_U(s) + \frac{1}{|s|} K_{M,U}.$$

Eine Maschine ist universell, wenn sie jede andere Maschine simulieren kann. $K_{M,U}$ ist die Länge des Programmes zur Simulation von M auf U .

Damit haben wir eine Analogie zum Kodierungstheorem, wenn wir H und $\frac{1}{|s|} I_M(s)$ in Beziehung bringen. Der wesentliche Unterschied zwischen beiden Konzepten besteht aber darin, daß Kolmogoroff eine *individuelle* Information, Shannon aber nur eine *mittlere* Information gewinnt.

Kolmogoroff verwendet dieses Konzept, um dem Begriff der zufälligen Folge eine intuitiv gut faßbare Form zu geben. Kolmogoroffs Konzept wurde von G. Chaitin, M. Löf und C.P. Schnorr in den Jahren 1966 bis 1971 ausgearbeitet und weitergeführt. Chaitin hat 1975 die Analogie von H und I herausgearbeitet, indem er die fundamentalen Relationen für H auch für I nachgewiesen hat. Hierzu definiert er noch eine bedingte Information $I_M(p|q)$, indem er anstelle des leeren Rechenspeichers auch eine beliebige Inschrift q zuläßt. Ansonsten verwendet er die Definition von I , die wir kennengelernt haben.

Wir wollen hier nicht alle sieben Relationen aufführen, sondern uns mit zweien begnügen:

$$(1) \quad \begin{cases} H(X \times Y \times Z) = H(Z) + H(Y | Z) + H(X | Y \times Z) \\ I_M(s, r, t) = I_M(t) + I_M(r | t) + I(s, r, t) + O(1) \end{cases}$$

$$(2) \quad \begin{cases} H(X) \leq H(X | Y) + H(Y | Z) + H(Z) \\ I_M(s) \leq I_M(s | r) + I(r | t) + I(t) + O(1) \end{cases}$$

Hierin sind X, Y, Z Informationsquellen, d.h. Wahrscheinlichkeitsräume, \times bezeichnet die Bildung von Produkträumen. Der senkrechte Strich $|$ bezeichnet die Bildung von bedingten Wahrscheinlichkeiten. Die Relationen zwischen den Kolmogoroff-Komplexitäten enthalten noch Konstanten $O(1)$, die für große $|s|$ vernachlässigt werden dürfen. Chaitin und Schnorr zeigen über diese formale Ähnlichkeit hinausgehend, daß diese Konzepte auch wesensmäßig verwandt sind. Schnorrs Ansatz ist grundsätzlicher, aber auch schwieriger zu vermitteln. Wir folgen hier weiter Chaitin:

Chaitin setzt

$$P_M(s) = \sum_{p=M(p, \epsilon)} 2^{-|p|}.$$

Bildet die Menge der Programme einen präfixfreien Code, d.h. ist kein Programm p Anfangsabschnitt eines Programmes $p' \neq p$, dann erhält man Wahrscheinlichkeiten, die wie folgt interpretiert werden können:

$P_M(s)$ gibt die Wahrscheinlichkeit dafür an, daß durch Münzwürfe Programme p erzeugt werden, die s definieren.

Nun kann man

$$H_M = \sum P_M(s) I_M(s)$$

setzen und erhält wegen

$$I_M(s) \sim -\log P_M(s)$$

eine Größe, die im Sinne Shannons interpretiert werden kann.

Wir fassen zusammen: *Der Kolmogoroff'sche Ansatz erlaubt es, ein individuelles Informationsmaß für Folgen zu definieren und geht hierdurch wesentlich über das Shannon'sche Konzept hinaus.*

Wir geben nun eine Relativierung des Konzeptes auf Komplexitätsklassen.

Wir haben in der Theorie der Kolmogoroff-Komplexität nur an einer Stelle davon Gebrauch gemacht, daß die zugrundegelegten Maschinen universelle Turingmaschinen sind, nämlich dort, wo wir die Komplexität einer Folge relativ zu zwei Maschinen verglichen haben. Falls sich M auf U durch ein Programm der Länge $K_{M,U}$ nachbilden läßt, dann hätten wir

$$I_M(s) \leq I_U(s) + K_{M,U}. \quad (*)$$

Ersetzen wir die Klasse der universellen Turingmaschinen durch eine Teilklasse K , dann gilt zwar $(*)$ auch für $M \in K$, aber es muß

$$I_U(s) \leq I_M(s) + K_{U,M}$$

nicht gelten. Gibt es in K aber universelle Maschinen \tilde{U} in dem Sinn, daß jede Maschine $M \in K$ auf \tilde{U} simuliert werden kann, dann gilt zwischen den Maschinen aus K und den K -universellen eine Beziehung wie $(*)$.

Ein Beispiel für eine solche Klasse sind die linear bandbeschränkten Turingmaschinen von Myhill. Die Existenz von universellen linear bandbeschränkten Maschinen wurde von Hartmanis gezeigt.

Die Bedeutung der Verfeinerung der Kolmogoroff-Komplexität in diesem Sinne für eine Verfeinerung des Begriffes der Zufälligkeit von Folgen hat C.P. Schnorr in seiner Habilitationsschrift [Sch] untersucht. Es ist aufgrund des Gesagten klar, daß dies auch Konsequenzen für den Informationsbegriff besitzt. Der Informationsgehalt einer Folge mag bezüglich einer Klasse wesentlich höher erscheinen als bezüglich einer anderen Klasse. Wir verfolgen dieses Thema hier aber nicht weiter, sondern betrachten Anwendungen dieser Konzepte auf Probleme der Informatik [Hu].

Eine Anwendung des Shannonschen Informationsbegriffes zur Abschätzung der Zeitkomplexität von bijektiven Funktionen.

Wir haben gesehen, daß die Länge von Programmen zur Beschreibung von Folgen eine informationstheoretische Bedeutung besitzt. Die auf der Kolmogoroff-Komplexität basierende Begründung des Informationsbegriffes geht insofern über den

Shannonschen Ansatz hinaus, als dort eine "individuelle" und hier nur eine mittlere Informationsgröße berechnet wird. Da man aber die Entropie einer Quelle, d.h. die Funktion H leichter berechnen kann als etwa $I_M(s)$, so kann man versuchen, inwieweit sich $I_M(s)$ durch eine geeignet definierte Funktion H abschätzen lässt. Hierzu müßte man zunächst der Folge s eine Wahrscheinlichkeitsverteilung P zuordnen, so daß die Entropie $H(p) = I_M(s)$ von unten abschätzt.

Es stellt sich also zunächst das Problem, Abbildungen eine Entropie zuzuordnen. Wir betrachten hier den einfachsten Fall. Es sei $[1 : N]$ die Menge der natürlichen Zahlen n mit $1 \leq n \leq N$. $\pi : [1 : N] \rightarrow [1 : N]$ sei eine Permutation. Man ordne π eine Entropie $H(\pi)$ zu, die unseren Zwecken dienlich ist. Hierzu definieren wir ein Maß p auf $[1 : N]$, indem wir für $Q \subset [1 : N]$ definieren

$$p(Q) = \frac{|Q|}{N},$$

worin $|Q| =$ Anzahl der Elemente von Q .

Nun betrachten wir Partitionen α von $[1 : N]$; α besteht also aus einer Menge von Mengen $Q_1, \dots, Q_j \subset [1 : N]$ mit

$$Q_i \neq \emptyset, \quad Q_i \cap Q_l = \emptyset \quad \text{für } i \neq l, \quad i, l = 1, \dots, j; \quad \text{und} \quad \bigcup_{i=1}^j Q_i = [1 : N].$$

Nun ordnen wir jedem Q_i eine Informationsquelle zu, indem wir definieren

$$P^{(i)}(Q_l) = \frac{P(Q_l \cap \pi(Q_i))}{P(Q_i)} \quad \text{für } i = 1, \dots, j.$$

Also wir geben an, mit welcher Häufigkeit ein Element von Q_i durch die Permutation π in die Menge Q_l geworfen wird. Wir definieren nun

$$H(\alpha, \pi) = \sum_{i=1}^j P(Q_i) H(P^{(i)})$$

und haben damit α und π eine mittlere Entropie der Entropien $H(P^{(i)})$ zugeordnet. Nun bildet man

$$H(\pi) = \max_{\alpha} H(\alpha, \pi).$$

α zulässige Partition. $H(\pi)$ ist die maximale Entropie, die unsere Konstruktion der Abbildung π zuordnen kann. Nun zeigt man: Sind π_1, π_2 zwei Permutationen von $[1 : N]$, dann gilt

$$H(\pi_1 \cdot \pi_2) \leq H(\pi_1) + H(\pi_2),$$

worin $\pi_1 \cdot \pi_2$ die durch Hintereinanderanwendung von π_1 und π_2 erzeugte Permutation ist.

Dem Rechner mögen als Elementaroperationen die Menge $E = \{\epsilon_1, \epsilon_2, \dots\}$ von Permutationen zur Verfügung stehen, und E gestatte es, jede Permutation von $[1 : N]$ zu erzeugen. Wir können dann also schreiben:

$$\pi = \eta_1 \cdot \eta_2 \cdot \dots \cdot \eta_k \quad \text{mit } \eta_i \in E.$$

Nun folgt aus obiger Ungleichung

$$H(\pi) \leq H(\eta_1) + H(\eta_2) + \dots + H(\eta_k).$$

Ist

$$\lambda = \max_{\eta \in E} H(\eta),$$

dann gilt also

$$H(\pi) \leq k \cdot \lambda \quad \text{oder} \quad k \geq \frac{H(\pi)}{\lambda}.$$

Damit haben wir eine Abschätzung für die Anzahl der Permutation gewonnen, die man braucht, um π mittels E zu erzeugen. Das ist aber auch eine untere Schranke für die Anzahl der Operationen, die unser Rechner benötigt, um π zu berechnen.

Wir betrachten zunächst eine konkrete Anwendung. Es sei eine große quadratische Matrix A gegeben, die man an ihrer Hauptdiagonalen spiegeln soll. π wird also durch folgende Vorschrift beschrieben.

$$\left(\begin{array}{cccc} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ | & | & \ddots & | \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{array} \right) \xrightarrow{\pi} \left(\begin{array}{cccc} a_{11} & a_{21} & \cdots & a_{n1} \\ a_{12} & a_{22} & \cdots & a_{n2} \\ | & | & \ddots & | \\ a_{1n} & a_{2n} & \cdots & a_{nn} \end{array} \right)$$

Die Elemente der Matrix seien paarweise verschieden. Diese Matrix sei zu groß, um sie in den Hauptspeicher des Rechners zu schreiben. Aber zwei Zeilen der Matrix mögen im Hauptspeicher Platz haben. Da die Transportzeiten zum Auslagern von Zeilen auf den Hintergrundspeicher im Vergleich zu einer Elementaroperation des Rechners sehr viel Zeit erfordern, wollen wir nur die Anzahl der Auslagerungen zählen. Dementsprechend nehmen wir als Menge E die Menge aller Permutationen der Elemente zweier Zeilen. Die zulässigen Partitionen bestehen in der Zerlegung der Matrix in Zeilen.

Nun rechnet man leicht nach:

$$H(\pi) = \log n$$

und

$$H(\eta) \leq \frac{k \cdot \log k}{n} \quad \text{für } \eta \in E$$

wenn wir erlauben, anstelle von zwei Zeilen k Zeilen zugleich im Hauptspeicher zu halten. Also erhalten wir für $k = 2$ das Resultat

$$H(\pi) \geq \frac{n}{2} \cdot \log n.$$

Der soweit beschriebene Ansatz stammt von H.J. Stoß (1972). Diese Theorie ist allerdings nicht unter dem hier geschilderten informationstheoretischen Aspekt entwickelt worden. Die hier gegebene Sicht des Problems wurde von W. Paul in seiner Dissertation (1973) für allgemeinere Maschinenmodelle entwickelt, während Stoß sich ganz auf eine spezielle Turingmaschine bezog. Daß diese untere Schranke auch scharf ist, ergibt sich aus einem von Floyd angegebenen Algorithmus. W. Paul konnte darüber hinaus zeigen, daß man alle optimalen Algorithmen zur Lösung dieses Problems erhält, indem der Algorithmus $\eta \in E$ stets so wählt, daß H maximal ansteigt.

Wir wollen den Ansatz nun nochmals kritisch anschauen: Wir haben nirgends Gebrauch davon gemacht, daß der Algorithmus deterministisch ist. Wir haben überhaupt zur Organisation des Rechners nichts gesagt. Unsere Abschätzung gilt also auch für nichtdeterministische Programme. Damit sollte der Ansatz für die Komplexität bijektiver Funktionen f und ihrer Inversen eine untere Schranke liefern. Das heißt, es sollte $H(f)$ und $H(f^{-1})$ eine untere Schranke für die Komplexität von f liefern.

Wir sehen unsere Konstruktion unter dem Aspekt der Symmetrie zwischen α und $\pi\alpha$ nochmals an. Seien α und β zwei beliebige Partitionen von $[1 : N]$. Wir betrachten dann

$$P_{(Q')}^{(Q)} = \frac{P(Q \cap Q')}{P(Q)} \quad \text{für } Q \in \alpha, \quad Q' \in \beta$$

und

$$H(\beta | \alpha) = \sum_{Q \in \alpha} P(Q) \cdot H(P_{(Q')}^{(Q)}).$$

Nun bilden wir

$$|\alpha, \beta| = H(\beta | \alpha) + H(\alpha | \beta)$$

und finden für Partitionen α, β, γ von $[1 : N]$

$$|\alpha, \gamma| \leq |\alpha, \beta| + |\beta, \gamma|.$$

Wir haben also auf der Potenzmenge von $[1 : N]$ einen Abstand definiert.

Nun setzt man für Permutationen π von $[1 : N]$

$$|\pi| = \max_{\alpha} |\alpha, \pi(\alpha)|$$

und erhält wieder

$$|\pi_1 \cdot \pi_2| \leq |\pi_1| + |\pi_2|.$$

Dieser Ansatz läßt sich auf wesentlich allgemeinere Situationen übertragen. Die Tragweite des Konzeptes ist noch nicht ausgelotet.

Eine Anwendung der Informationstheorie zur Konstruktion nahezu optimaler Suchbäume

Die Verwaltung großer Datenmengen spielt in Rechnern eine hervorragende Rolle. Wir betrachten hier als Beispiel den Fall, der durch die Verwendung eines Wörterbuches im Rechner gut charakterisiert wird. Wir haben eine Menge U , etwa die

Menge der deutschen Wörter, die als Schlüssel eines deutsch-englischen Wörterbuches dienen. Die lexikographische Ordnung der Elemente $u, v \in U$ sei durch $u \leq v$ bezeichnet. Die Benutzung eines Wörterbuches wird durch die Ausnutzung dieser Relation möglich, wie man rasch bemerkt, wenn man das deutsch-englische Wörterbuch in der Richtung englisch-deutsch verwenden will. Wie sucht man günstig in einem Wörterbuch, das in der Maschine abgespeichert ist? Sei etwa das Wort u gegeben und es sei nach einer Übersetzung v von u ins Englische gefragt. Das Nachschlagen kann auf folgende Weise geschehen: Man schlägt das Wörterbuch in der Mitte auf und schaut nach, ob man das Wort direkt gefunden hat, oder ob es in der linken oder in der rechten Hälfte liegt.

Liegt u in der linken Hälfte U_l , dann verfährt man nun mit u und U_l wie vorher mit u und U . Entsprechend verfährt man, wenn u in der rechten Hälfte U_r liegt. Hat das Wörterbuch N Seiten, und steht auf jeder Seite genau ein Wort, dann hat man nach spätestens $\lfloor \log N \rfloor + 1$ Suchschritten den Eintrag u gefunden, oder man weiß, daß u nicht in U vorkommt. Natürlich könnte man auf die Idee kommen, u als Adresse für den Wörterbucheintrag im Speicher zu verwenden. Wenn der Speicher groß genug ist, geht dies auch, doch da bei weitem nicht jede Adresse als Wort in U vorkommt, würde dies zu einer riesigen Speicherplatzverschwendungen führen. Und die Zugriffszeit eines so riesigen physikalischen Speichers wächst auch logarithmisch mit seiner Größe.

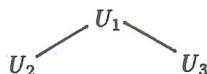
Das von uns beschriebene Suchverfahren ist sicher nicht optimal hinsichtlich seiner mittleren Suchzeit, wenn die verschiedenen Wörter mit sehr verschiedener Wahrscheinlichkeit nachgeschlagen werden, was im praktischen Gebrauch von Wörterbüchern ja auch der Fall ist. Wir gehen deshalb davon aus, daß uns auch eine Wahrscheinlichkeitsverteilung $p: U \rightarrow [0, 1]$ gegeben ist, die gerade zum Ausdruck bringt, wie häufig ein Wort $w \in U$ nachgeschlagen wird. Wir wollen also annehmen, daß jedes Wort, das nachgeschlagen wird, auch in U vorkommt. Dies ist keine Einschränkung der Allgemeinheit, da man die nicht in U vorhandenen Wörter durch "symbolische" Wörter in U einschieben kann, wobei ein symbolisches Wort alle Wörter einer Lücke zusammenfaßt.

Wir beschreiben nun unsere Suchschemata durch Bäume. Unser Ziel ist, unter diesen Suchverfahren die Güte der besten Verfahren abzuschätzen.

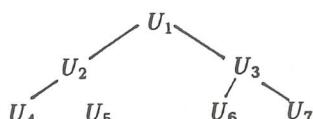
Wir ordnen (U, \leq) in folgender Weise Bäume zu: Man wähle aus U ein Element u_1 aus und bilde

$$U_2 = \{u \in U \mid u < u_1\}, \quad U_3 = \{u \in U \mid u > u_1\}$$

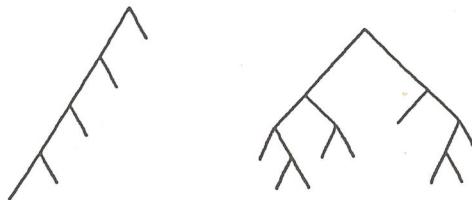
und betrachte den Baum



Nun verfahre mit U_2 und U_3 wie vorher mit U . Wähle in U_2 ein Element u_2 und in U_3 ein u_3 aus und bilde die entsprechenden Teilmengen U_4, U_5, U_6, U_7 , wie dies der folgende Baum zeigt:



Dieses Verfahren setzt man solange fort, bis die Mengen U_i einelementig geworden sind. Je nach Auswahl der u_i können die Bäume sehr verschieden aussehen. Zwei Beispiele geben die folgenden Bäume:



Haben wir U auf die beschriebene Weise auf einen Baum verteilt, dann können wir wie folgt suchen: Wir fragen: Ist $u = u_1, u < u_1, u > u_1$. Gilt etwa $u < u_1$, dann suchen wir in U_2 weiter, d.h. wir fragen $u = u_2, u < u_2, u > u_2$. Je nach Resultat setzen wir unsere Suche in einem der beiden Unterbäume von u_2 fort oder beenden unser Suchen, da $u = u_2$ ist.

Zu einem Suchvorgang gehört genau ein Suchpfad im Baum, der eindeutig beschrieben wird durch die Angabe, welche der Abfragen $=, <, >$ an dem jeweiligen Knoten erfüllt ist. So ordnet jeder unserer Suchbäume B jedem $u \in U$ ein Wort $w = (x_1, x_2, \dots, x_n)$ zu mit $x_i \in \{<, >\}$ für $i = 1, \dots, n - 1$ und $x_n \in \{=\}$, das den Suchpfad für u in B bezeichnet. Wir schreiben in diesem Fall

$$w = C_B(u), \quad C_B : U \rightarrow \{<, >, =\}^*.$$

Die Länge $|w|$ von w gibt die Suchzeit an und

$$m(B) = \sum_{u \in U} p(u) |C_B(u)|$$

also die mittlere Suchzeit des durch B definierten Suchverfahrens. Nun ist C_B aber eine Kodierung der Informationsquelle (U, p) . Also wissen wir aufgrund des oben angegebenen Kodierungssatzes

$$\frac{H(p)}{\log 3} \leq m(B).$$

Der Term $\log 3$ kommt daher, daß wir hier anstelle eines binären ein ternäres Alphabet haben. Würden wir beliebige ternäre Kodierungen zulassen, dann wüßten wir aufgrund des Kodierungssatzes, daß sich auch

$$m(B) \leq \frac{H(p)}{\log 3} + 1$$

erreichen ließe. Unsere Kodierungen sind insofern speziell, als sie bis auf das letzte Zeichen binär sind, dieses aber stets " $=$ " ist. Unter einer sehr schwachen Voraussetzung über p kann man zeigen, daß für balancierte Bäume

$$m(B) \leq H(p)$$

gilt [Ho]. Ein Baum ist balanciert bezüglich p , falls die Auswahl von u so erfolgt, daß $|p(U_2) - p(U_3)|$ ein Minimum wird und wenn alle übrigen u_i nach dem gleichen Prinzip gewählt werden.

Zur Theorie der Suchbäume sei auf Knuth [Kn] verwiesen.

Wir sehen, daß hier die Informationstheorie in einfacher Weise sehr nützliche Hinweise auf eine optimale Lösung wichtiger Probleme der Informatik vermittelt.

Eine Anwendung der Kolmogoroff-Komplexität

Ein wichtiges, aber leider sehr schwieriges Problem der Informatik besteht darin, die Leistungsfähigkeit verschiedener Rechnerkonzepte gegeneinander abzuwagen. Dies kann unter verschiedenen Einsatzumgebungen der Rechner zu verschiedenen Resultaten führen. Wir wollen dieses Problem hier für zwei vereinfachte Rechnermodelle betrachten. Der Vergleich der Modelle soll hinsichtlich eines "Realzeitbetriebes" getroffen werden.

Wir nehmen an, daß das Modell M_1 mit zwei Speichern versehen ist, die wie üblich zweidimensional organisiert sind. Der Speicherinhalt wird also wie ein Element einer Matrix aus zwei zueinander orthogonalen Richtungen angesteuert. Das Modell M_2 hingegen ist mit nur einem solchen Speicher versehen. Die Einträge in die Zellen des Speichers lassen genau zwei Werte 0 und 1 zu und diese Werte können in jede Zelle beliebig, ohne Rücksicht auf Nachbareinträge, eingeschrieben oder gelesen werden.

Nun nehmen wir eine gegenüber den käuflichen Rechenmaschinen wesentliche Einschränkung vor. Das Adressenregister kann nur so verändert werden, daß in zwei aufeinanderfolgenden Rechenschritten nur Nachbarfelder der Speicher erreicht werden können. Man veranschaulicht sich dies, indem man sich einen Lese-Schreibkopf auf der gerade betrachteten Speicherzelle vorstellt. Dieser Kopf kann in einem Rechenschritt, also höchstens zu einer unmittelbar benachbarten Zelle, links, rechts, oben oder unten übergehen; er darf aber auch auf der Zelle verweilen.

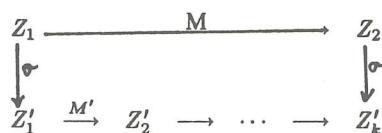
Um den Nachteil von M_2 gegenüber M_1 etwas auszugleichen, nehmen wir an, daß zu M_2 zwei voneinander unabhängige Adressregister gehören, die es gestatten, zwei verschiedene Speicherzellen zugleich zu besuchen. Dies ist bis jetzt technisch nur schwer zu realisieren. Wir fragen deshalb: Lohnt sich dieser Aufwand, kann dann M_2 mehr als M_1 ? W. Paul, J.I. Seiferas und J. Simon haben gezeigt, daß M_2 beim Realzeitbetrieb in der Tat stärker ist als M_1 .

Wir wollen hier den Gedankengang, der zu dieser Einsicht führt, grob schildern.

Zunächst müssen wir sagen, was eine Realzeitsimulation einer Maschine M auf einer Maschine M' sein soll.

Die Menge der internen Zustände der Maschine M sei Z , der von M' sei Z' . Ein Rechenschritt führt etwa z_1 in z_2 über. Es sei $\sigma : Z \rightarrow Z'$ eine Abbildung, die jedem $z \in Z$ ein $z' \in Z'$ zuordnet. σ heißt eine *Realzeitsimulierung*, wenn (1) und (2) gilt:

(1) Es gibt eine Konstante $c \in \mathbb{N}$, so daß das folgende Diagramm für ein $k \leq c$ erfüllt ist.



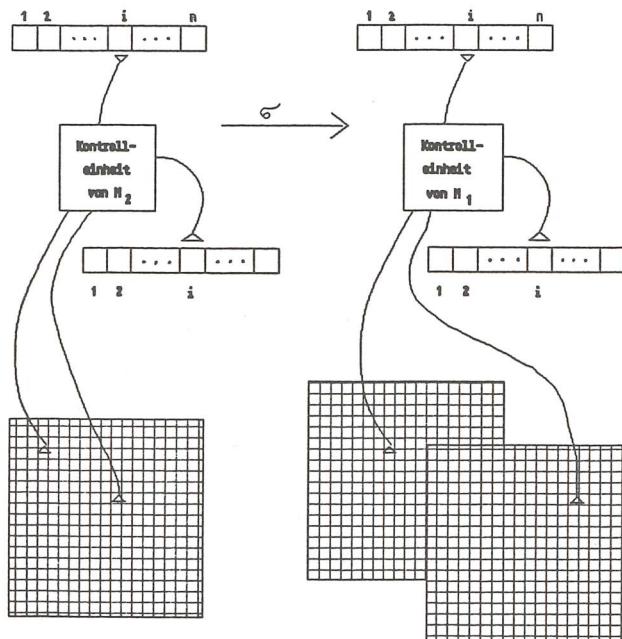
Das heißt: M' bildet die Berechnungen von M mit einer Zeitverzögerung um höchstens den Faktor c nach.

(2) Das Ein- und Ausgabeverhalten von M und M' ist gleich. Das soll folgendes heißen: M und M' erhalten genau die gleiche Folge von Eingabezeichen, eventuell M' mit der Verzögerung um den Faktor c . M und M' haben genau die gleiche Ausgabefolge erzeugt, wenn sie beide zum Lesen des i -ten Eingabezeichens übergehen.

Man stelle sich also einfach vor, daß die Eingaben und die Ausgaben beider Maschinen jeweils zur gleichen Zeit erscheinen, daß aber M' c -mal so schnell rechnen kann wie M .

Der Satz von Paul, Seiferas und Simon besagt, daß M_2 nicht in Realzeit durch M_1 simuliert werden kann. In anderen Worten: Die Speicherorganisation von M_2 gibt M_2 einen solch großen Vorteil über M_1 , daß man diesen nicht dadurch ausgleichen kann, daß man die Rechengeschwindigkeit von M_1 um einen beliebigen, aber konstanten Faktor erhöht.

Wir veranschaulichen zunächst beide Maschinen M_1 und M_2 durch die folgenden Figuren:



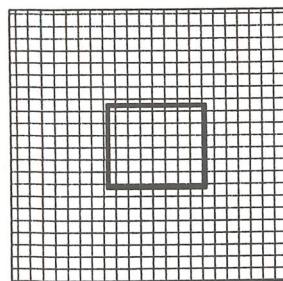
Man sieht leicht, daß M_1 auf M_2 in Realzeit simuliert werden kann. Hierzu färbe man die Zellen des Speichers von M_1 schachbrettartig in weiße und schwarze Felder. Den Inhalt von Speicher 1 von M_1 schreiben wir in die weißen, den Inhalt des Speichers

2 in die schwarzen Felder und zwar den Inhalt der (i, j) -ten Speicherzelle von M_1 in die (i, j) -te schwarze bzw. weiße Speicherzelle von M_2 . Man sieht, daß die hierdurch definierte Abbildung der Speicherzustände zu einer Simulation mit $c = 2$ fortgesetzt werden kann, wenn die Kontrolleinheit von M_2 geeignet programmiert wird.

Wir wollen nun plausibel machen, daß dies in umgekehrter Richtung nicht gilt. Die Beweisidee ist die folgende: Wir lassen M_2 seinen Speicher in spezieller Weise beschreiben, nämlich so, daß sich die beiden Köpfe immer weiter voneinander entfernen. Ist die Entfernung der beiden Köpfe größer als c , dann kann M_1 dieser Rechnung nur folgen, wenn der Speicher 1 von M_1 ganz dem Kopf 1 von M_2 und der Speicher 2 ganz dem Kopf 2 von M_2 zugeordnet wird. Auf diese Weise erreichen wir, daß z.B. nur ein Speicher von M_1 die Information enthält, die etwa in einem hinreichend großen Rechteck L unten rechts im Speicher von M_2 steht. Nun schicken wir beide Köpfe von M_2 in das Rechteck L . Ist L groß genug, dann können beide Köpfe die Speicherzellen von L in einer Weise ausgeben, der M_1 nicht folgen kann, da ihr dafür nur ein Kopf zur Verfügung steht.

Das ist die Idee des Beweises. Um die Schwierigkeiten des Beweises deutlich zu machen, argumentieren wir nun gegen unsere Idee: Es ist zwar richtig, daß die beiden Speicher von M_1 nicht je den ganzen Speicher von M_2 kopieren können, aber es muß doch nicht so sein, daß etwa Speicher 1 von M_1 nur die obere Hälfte von Speicher M_2 enthält. Die beiden Speicher könnten doch von Zeit zu Zeit die Rolle wechseln, so daß in jedem der Speicher von M_1 doch eine recht dichte Information über den Speicher von M_2 enthalten ist. Was dann noch fehlt, wird ein geschicktes Programm *interpolieren* können.

Wir sehen, daß wir nicht darum herum kommen, unsere informationstheoretischen Argumente zu präzisieren. Was hat es mit der Interpolation auf sich? Betrachten wir die durch die untenstehende Figur beschriebene Situation.



Wir beschreiben die große Matrix in zufälliger Weise und nehmen anschließend die Werte der eingerahmten Teilmatrix heraus. Die entsprechende Frage lautet nun, ob wir diese Werte aus der Umgebung rekonstruieren können, wenn dieser Rahmen z.B. nur geeignet hingelegt wird; denn in jeder zufälligen Folge gibt es stets gesetzmäßig erscheinende Teilstufen.

Man sieht, wie hier in ganz natürlicher Weise die Kolmogoroff-Komplexität ins Spiel kommt: Die Beschreibung von Folgen, hier 2-dimensionale Muster, durch kürzeste Programme.

Wir nennen nun ein rechteckiges Muster R *zufällig*, wenn $I_M(R) \geq |R|$ ist, wenn $|R|$ die Anzahl der Zellen von R ist. Ein Abzählargument zeigt, daß es in diesem Sinne zufällige Muster gibt. Ist nun R' ein rechteckiges Teilmuster von R , dann kann man nach $I_M(R'|R - R')$ fragen, das heißt, nach dem kürzesten Programm, das R' aus $R - R'$ zu rekonstruieren vermag. Man zeigt, daß $I_M(R', R - R') = I_M(R')$ im wesentlichen gilt.

Zum Beweis unseres Satzes genügt es, eine spezielle Maschine vom Typ M_2 anzugeben, die keine vom Typ M_1 in Realzeit simulieren kann. Wir wählen M_2 so, daß wir über das Eingabeband den Speicher von M_2 mit einem zufälligen Muster beschreiben. Nach vorherigem Resultat ist es mit der Interpolation von auf den jeweiligen Speichern fehlenden Informationen nichts, so daß unsere Beweisidee als gerettet erscheint. Die Durchführung im Einzelnen erfordert jedoch noch eine trickreiche Argumentation, die wir hier nicht vorführen wollen. Es sei nur angemerkt, daß diese Beweisidee für höherdimensionale Speicher das gleiche Resultat liefert, nicht aber für den eindimensionalen Fall. Letzteres weist uns auf die Notwendigkeit der formalen Durchführung des Beweises hin.

Ich hoffe aber, daß es aus dem Gesagten deutlich wird, wie hier unsere Fassung des Informationsgehaltes einer Folge relativ zu einer Maschine Früchte bringt, indem sie zur Lösung eines Problemes führt, das mehr als 10 Jahre offen war.

Wir wollen noch etwas zur praktischen Bedeutung des Resultates bemerken: In jeder verfügbaren Maschine hat man eine flexible Adressenrechnung, die wir hier nicht haben. Würden wir die Bewegung der Köpfe etwas flexibler gestalten, so daß ein Kopf in einem begrenzten Umfang springen darf, würden wir kein qualitativ verschiedenes Resultat erhalten. Für den Vergleich von Maschinen, deren Adressenrechnung es erlaubt, von jeder Zelle zu jeder zu springen, gibt unsere Überlegung keine Hilfe. Aber immerhin ist ein erster Schritt getan, der vielleicht zu einer Theorie hinführt, die auch die praktisch bedeutsamen Fragen beantwortet.

Ausblicke

Eine systematische Entwicklung der Theorie der Information in den hier skizzierten Bahnen steht noch aus. Ihre Bedeutung zum Verständnis der Probleme der Informatik und ihre praktische Anwendbarkeit in der Informatik sind noch nicht zu übersehen.

Die Ausstrahlung dieser Ideen auf andere Disziplinen hat gerade erst begonnen. Zunächst zeigt dieser Ansatz zur Begründung der Theorie der zufälligen Folgen auf der Basis des algorithmischen Informationsbegriffes, daß hierdurch Wendungen wie "Es ist wahrscheinlich, daß Cäsar in Britannien war" im Gegensatz zur klassischen Wahrscheinlichkeitstheorie eingeordnet werden können: Nämlich die Menge der in diesem Zusammenhang überlieferten Nachrichten erfährt durch die Annahme, daß Cäsar in Britannien war, eine *einfache* Erklärung. Kürzere Programme gelten hier als wahrscheinlicher. Nach dem gleichen Schema verfahren die Naturwissenschaftler, die die Theorie als richtig bezeichnen, die für die Gesamtheit der beobachteten Phänomene die einfachste Erklärung liefert. Hierzu ein Zitat aus einer Ansprache Einsteins an Planck (zu Max Plancks 60. Geburtstag 1918):

"Die Entwicklung hat gezeigt, daß von den denkbar theoretischen Konstruktionen eine einzige jeweils sich als unbedingt überlegen über allen anderen erweist. Keiner, der sich in den Gegenstand wirklich vertieft hat, wird leugnen, daß die Welt der Wahrnehmungen das theoretische System praktisch eindeutig bestimmt, trotzdem kein logischer Weg zu den Grundsätzen der Theorie führt" [We]. Wir möchten hier hinzufügen, daß der Satz von Kolmogoroff über $I_M(s)$ verheißt, daß unterschiedliche persönliche Auffassungen bei der Anpassung der Theorie an in ihrer Anzahl wachsende Erfahrungstatsachen asymptotisch bedeutungslos werden.

Daß die algorithmische Komplexität dem menschlichen Empfinden für Komplexität zumindest nicht zuwiderläuft, zeigen die Experimente, über die Frau A. Grigorjeff [Gr] berichtet: Man hat eine gewisse Menge geometrischer Figuren von Menschen hinsichtlich ihrer empfundenen Komplexität klassifizieren lassen und sie dementsprechend angeordnet. Die gleichen Figuren hat man dann ihrer Kolmogoroff-Komplexität entsprechend angeordnet. Beide Anordnungen zeigten eine größere Übereinstimmung, nämlich eine Korrelation von etwa 60 %. Die Kolmogoroff-Komplexität findet in jüngster Zeit zunehmend an Interesse. Der interessierte Leser sei hier auf [Hu] verwiesen.

Literatur

- [Ch] G.J. Chaitin: "A Theory of Program Size Formally Identical to Information Theory".
J. ACM 22,3 (1974)
- [G] A. Grigorjev: "Mündliche Mitteilung, diese Resultate werden in einem Buch von Frau Grigorjev enthalten sein, das bei Helliday New Jersey, Lawrence Ezlbaum Assoc. erscheint".
- [Ho] G. Hotz: "Schranken für balanced Trees bei ausgewogenen Verteilungen".
TCS 3, pp 51-59
- [HoSt] G. Hotz, M. Stadel: "Network Complexity", Fundamentals of Comp. Theory, Lect. Notes, Vol. 54 in Comp. SC., pp 401-425
- [Hu] D.T. Huynh: "Resource-Bounded Kolmogorov Complexity of Hard Languages"
Iowa State University, Computer Science Department, Report TR 85-10
- [Kn] D. Knuth: "Optimal Binary Search Trees".
Acta Informatica 1, pp 14-25 (1971)
- [K] A.N. Kolmogoroff: "Drei Zugänge zur Definition des Begriffes 'Informationsgehalt'".
(russisch) Probl. Peredeci Inform. 1, pp 3-11 (1965)
- [Ma] P. Martin-Löf: "The definition of random sequences".
Inform. Control 6, pp 602-619 (1966)

[Me] K. Mehlhorn: "Nearly Optimal Binary Search Trees".

Acta Informatica 5, pp 287–295 (1975)

[P] W.J. Paul: "Kolmogoroff Complexity and Lower Bounds".

FCT'79, Ed. L. Budach, Akademia-Verlag, Berlin, pp 325–334

[PSeSi] W.J. Paul, J. Seiferas, J. Simon: "An Information-Theoretic Approach to Time Bounds for On-Line Computation".

J. Comp. and System Science 23 (1981)

[P] W.J. Paul: "Zeitkomplexität von Algorithmen zum Umordnen endlicher Mengen",
Dissertation Saarbrücken 1973

[Sch] C.P. Schnorr: "Zufälligkeit und Wahrscheinlichkeit".

Lecture Notes in Math. Vol. 218 (1971), Springer Verlag.

[Sh] C.E. Shannon: "A Mathematical Theory of Communication".

BSTJ, Vol. 27 (1948), pp 379–423, 623–656

[So] R.I. Solomonoff: "A Formal Theory of Inductive Inference".

Inform. and Control 7, pp 1–22 (1964)

[St] H. J. Stöß: "Rangierkomplexität von Permutationen".

Acta Informatica 2, pp 80–96 (1973)

[We] H. Weyl: "Philosophie der Mathematik und Naturwissenschaft".

Oldenbourg Verlag München, 173 S., (1928) (1965)