Kolmogorow Komplexität

Hagen Mahnke

February 1, 2011

1 Eine Definition von Information

Eines der wichtigsten Konzepte der Informatik, die Information, ist in vielfacher Weise beschrieben, jedoch gibt es keine eindeutige und allgemeingültige Definition. Die hier verwendete Definition von Information betrachtet die Länge der minimalen Beschreibung einer Information als Komplexität. Zur Veranschaulichung betrachten wir die folgenden binären Sequenzen:

$$\text{A} = 01010101010101010101010101010101$$
$$\text{B} = 10100101110101000101011101001011$$

Intuitiv ist die Menge an Information die A enthält sehr gering, man kann Sie als 16 mal *01* beschreiben. Bei B dagegen gibt es keine derart offensichtliche Beschreibung. Wir betrachten nur die kürzeste Beschreibung von Information, weil eine Beschreibung die länger als die Information oder genauso lang ist, nichts über die Menge an Information sagt. Eine Beschreibung die wesentlich kürzer ist zeigt, dass man die Information bedeutend komprimieren kann. Natürlich ist dann die kürzeste Beschreibung die interessanteste. Nun wollen wir diese Idee formalisieren. Dazu legen wir zuerst fest, dass nur Objekte betrachtet werden die binäre Strings sind und auch deren Beschreibung nur als binärer String geschieht. Da beliebige andere Objekte als binäre Strings repräsentiert werden können, wird die Reichweite der Theorie dadurch nicht eingeschränkt.

2 Beschreibung durch Algorithmen

Zur Definition von Information können verschiedene Beschreibungssprachen verwendet werden. Die verwendete Sprache beeinflußt dabei die Charakteristik der Definition. Wir werden unsere Definition auf Algorithmen aufbauen.

2.1 Turing Maschinen als Beschreibungssprache

Dazu betrachten wir Turing-Maschinen mit einer Eingabe w die, wenn sie auf dieser Eingabe gestartet werden, einen String x ausgeben. Die TM M selber wird ebenfalls als binärer String $\langle M \rangle$ repräsentiert. Dann ist die Konkatenation der beiden $\langle M, w \rangle$ eine Beschreibung eines Strings x nach unserer Definition. Dabei muss der String eindeutig nach TM und Eingabe unterscheidbar sein. Zu diesem Zweck kann man alle Zeichen der TM verdoppeln und den Übergang zwischen TM und Eingabe mit *01* codieren.

$$\langle M, w \rangle = \underbrace{1100111111001100000}_{\langle M \rangle}\mathbf{01}\underbrace{01101010010001110}_{\mathbf{w}}$$

2.2 Definition

Sei x ein binärer String. Die ***minimale Beschreibung*** von x ist der kürzeste String $\langle M, w \rangle$ bei dem die TM M bei Eingabe w mit x auf dem Band hält und wird mit $d(x)$ bezeichnet. Gibt es mehrere solcher Strings wird der lexikographisch erste gewählt. Die ***Beschreibungskomplexität*** $K(x)$ ist die Länge von $d(x)$.

$$K(x) = |d(x)|$$

2.3 Theorem

Die Beschreibungskomplexität eines Strings ist höchstens eine feste Konstante länger als der String. Diese Konstante ist universell, d.h. sie hängt nicht vom String ab.

$$\exists c \forall x[K(x) \leq |x| + c]$$

Beweis Um eine obere Grenze für $K(x)$ zu beweisen muss man nur eine Beschreibung von x finden die nicht länger als die genannte Grenze ist. Betrachten wir folgende TM Id, die die Identitätsfunktion berechnet. Dann kann jeder String x durch $\langle Id, x\rangle$ beschrieben werden. Mit $|\langle Id\rangle| = c$ ist der Beweis vollständig.

An dieser Stelle wird auch klar, warum wir eine TM mit Eingabe benutzen. Es gibt Strings deren Beschreibung, wären sie in der Überführungsfunktion einer TM gespeichert, wesentlich länger als der String selber wären. Das widerspricht aber der intuitiven Annahme, das ein String nicht wesentlich mehr Informationen enthalten kann, als er lang ist.

2.4 Theorem

Verdoppelt man einen String so erhöht man die Beschreibungskomplexität höchstens um eine feste Konstante. Auch diese Konstante ist universell.

$$\exists c \forall x[K(xx) \leq K(x) + c]$$

Beweis Betrachten wir folgende TM M welche als Eingabe $\langle N, w\rangle$ erwartet, wobei N eine TM ist und w eine Eingabe für N.

M = Bei Eingabe $\langle N, w\rangle$:

1. Führe N auf w aus bis N hält und Ausgabe a produziert.
2. Gib aa aus.

Dann ist $\langle M\rangle d(x)$ eine Beschreibung von xx. Mit $|\langle M\rangle| = c$ und $K(x) = |d(x)|$ ist der Beweis komplett.

2.5 Theorem

Die Konkatenation zweier Strings führt zu einer Beschreibungskomplexität die höchstens um $2log(K(x))+c$ länger ist als die Summe der Beschreibungskomplexitäten der beiden Strings. Dabei ist $K(x)$ die Beschreibungskomplexität des ersten Strings und c eine Konstante

$$\exists c \forall x, y[K(xy) \leq 2log_2(K(x)) + K(x) + K(y) + c]$$

Beweis Man konstruiert eine TM M die ihre Eingabe in zwei getrennte Beschreibungen $\langle N, w\rangle, \langle O, v\rangle$ der Strings x und y aufteilt. Dann werden beide ausgeführt und die Ausgaben x und y als xy ausgegeben. Sind die Beschreibungen minimal, dann sind sie $K(x)$ bzw. $K(y)$ lang. Es ist aber notwendig in die Beschreibung der konkatenierten Strings auch eindeutig die Trennung der beiden Eingaben zu codieren. Dieses kann geschehen indem man vor die Beschreibung des ersten Strings $d(x)$ dessen Länge anfügt. Um die Angabe der Länge von $d(x)$ zu trennen werden die Bits wie vorhin beschrieben verdoppelt und eine Trennsequenz eingeführt. Da die binäre Codierung einer Zahl z die Länge $log_2(z)$ hat, ergibt sich hier der zusätzliche Wert $2log_2(K(x))$. Damit ist noch die Länge der Trennsequenz und die der TM M unbeachtet, aber diese sind konstant und somit unser c.

3 Optimalität der Definition

Neben der hier verwendeten Beschreibungssprache sind noch viele andere denkbar. Betrachten wir eine allgemeine Beschreibungssprache als berechenbare Funktion $p : \Sigma^* \longrightarrow \Sigma^*$. Dann ist die minimale Beschreibung von x in Bezug auf p, geschrieben $d_p(x)$, der kürzeste String für den $p(s) = x$ gilt. Dann ist $K_p(x) = |d_p(x)|$. Beispielsweise ist LISP eine Beschreibungssprache und $d_{LISP(x)}$ ist das kürzeste LISP Programm das x ausgibt. Das folgende Theorem zeigt, dass keine andere Beschreibungssprache wesentlich prägnanter ist als die hier verwendeten TM mit Eingabe.

3.1 Theorem

Für jede Beschreibungssprache p, existiert eine feste Konstante c die nur von p abhängig ist, so dass

$$\forall x[K(x) \leq K_p(x) + c]$$

Beweis Für eine beliebige Beschreibungssprache p, konstruiert man eine TM M, die bei Eingabe w, die Ausgabe $p(w)$ erzeugt. Dann ist mit c als Länge von M der Beweis komplett.

Anders gesagt, man konstruiert einen entsprechenden Interpreter in Form einer TM.

4 Nicht-komprimierbare Strings und Zufälligkeit

Im folgenden wird gezeigt, dass es Strings gibt die nicht kürzer beschrieben werden können als durch den String selber.

4.1 Definition

Sei x ein String. Dann wird x als ***c-komprimierbar*** bezeichnet, wenn

$$K(x) \leq |x| - c$$

Falls x nicht c-komprimierbar ist, heißt x ***nicht-komprimierbar durch c***. Falls x nicht-komprimierbar durch 1 ist ist, heißt x ***nicht-komprimierbar***.

4.2 Theorem

Nicht-komprimierbare Strings jeder Länge existieren.

Beweis Es gibt 2^n Strings der Länge n. Jede Beschreibung bezieht sich auf höchstens einen String. Jede Beschreibung ist ein nicht-leerer String, also ist die Anzahl von Beschreibungen mit Länge kleiner n höchstens die Summe aller Strings mit Länge kleiner oder gleich $n - 1$. Das sind genau

$$\sum_{0 \leq i \leq n-1} 2^i = 1 + 2 + 4 + 8 + ... = 2^n - 1$$

Also ist die Anzahl von Strings - und damit auch der potenziellen Beschreibungen - die kürzer als n sind geringer als die Anzahl von Strings der Länge n. Daher ist mindestens ein String der Länge n nicht-komprimierbar.

4.3 Korollar

Mindestens $2^n - 2^{n-c+1} + 1$ Strings der Länge n sind nicht-komprimierbar durch c.

Beweis Es gibt genau $2^{n-c+1} - 1$ Strings der Länge $n - c$. Nur solche Strings können aber eine um mindestens c kürzere Beschreibung eines Strings der Länge n sein. Also sind $2^n - 2^{n-c+1} + 1$ Strings der Länge n nicht-komprimierbar durch c.

Strings die nicht-komprimierbar durch c sind besitzen viele ***berechenbare Eigenschaften*** die man auch bei zufällig erzeugten Strings antrifft. So hat jeder nicht-komprimierbare String der Länge n eine ungefähr gleiche Anzahl an Nullen und Einsen, und die längste Folge von Nullen ist $O(log\ n)$ lang. Für diese Behauptungen werden wir keinen Beweis liefern, aber wir werden ein Theorem beweisen, welches die Grundlage für diese Behauptungen bildet. Das Theorem zeigt das jede berechenbare Eigenschaft, die ***für fast alle Strings gilt***, auch für alle ausreichend langen Strings die nicht-komprimierbar durch c sind gilt. Eine berechenbare Eigenschaft von Strings ist eine Funktion f die Strings auf $\{TRUE, FALSE\}$ abbildet. Wir sagen das eine berechenbare Eigenschaft für fast alle Strings gilt, wenn der Anteil von Strings der Länge n, für die sie $FALSE$ ist gegen 0 strebt wenn n groß wird. Es ist wahrscheinlich, dass ein zufällig erzeugter String eine Eigenschaft erfüllt, wenn diese für fast alle Strings gilt. Daher teilen Strings die nicht-komprimierbar durch c sind und zufällig erzeugte Strings diese Eigenschaften.

4.4 Theorem

Sei f eine berechenbare Eigenschaft die für fast alle Strings gilt. Dann gilt für jedes $b > 0$, die Eigenschaft f ist FALSE für nur endlich viele Strings die nicht-komprimierbar durch b sind.

Beweis Sei M der folgende Algorithmus.

M = Bei Eingabe eines binären Integers i:

1. Finde den $i$ten String s für den $f(s) = FALSE$, wobei die Strings lexikographisch geordnet sind.
2. Gib String s aus.

M kann folgendermaß genutzt werden, um kurze Beschreibungen von Strings die f nicht besitzen zu finden. Sei x ein solcher String und i_x der Index (binär) von x in einer lexikographisch geordneten Liste aller Strings die Eigenschaft f nicht besitzen. Dann ist $\langle M, i_x\rangle$ eine Beschreibung von x, mit der Länge $|i_x| + c$, wobei $c = |\langle M\rangle|$.

Wähle ein festes $b > 0$. Wähle n so, dass der Anteil der Strings mit Länge kleiner oder gleich n, die f nicht besitzen höchstens $1/2^{b+c}$ ist. Da f für fast alle Strings gilt, gibt es ein solches n. Dann gilt

$$i_x \leq \frac{2^n}{2^{b+c}} = 2^{n-b-c}$$

Daher ist $|i_x| \leq n - b - c$ und die Länge von $\langle M, i_x\rangle$ höchstens $(n - b - c) + c = n - b$. Also ist

$$K(x) \leq n - b$$

Ein String für den gilt, dass $K(x) \leq |x| - b$ ist b-komprimierbar. Also ist jeder String x für den $n - b \leq |x| - b \Leftrightarrow n \leq |x|$ gilt, b-komprimierbar. Somit sind von den Strings für die f $FALSE$ ist, nur Strings mit Länge kleiner n nicht-komprimierbar durch b, also nur endlich viele.

4.5 Theorem

Die Beschreibungskomplexität ist nicht berechenbar.

Beweis durch Widerspruch Angenommen es gibt einen Algorithmus P der die Beschreibungskomplexität $K(s)$ eines beliebigen Strings s berechnet. Betrachten wir jetzt einen Algorithmus Q mit Integer n als Eingabe, welcher, aufsteigend nach der Länge i, alle Strings durchgeht und ihre Beschreibungskomplexität berechnet bis ein $K(s) \geq n$ gefunden wird. Dieser Algorithmus hat eine feste Länge U und zusätzlich die Länge des Integers $log_2(n)$. Also zusammen $U + log_2(n)$. Da n stärker wächst als $log_2(n)$, gibt es ein n so, dass

$$U + log_2(n) < n$$

Aber damit ist die Beschreibungkomplexität des Strings kleiner als n, was ein Widerspruch zu der Annahme ist das P die Beschreibungskomplexität korrekt berechnet.

4.6 Theorem

Es gibt eine Konstante b so, dass für alle Strings x, die minimale Beschreibung $d(x)$ nicht-komprimierbar durch b ist.

Beweis durch Widerspruch Betrachten wir folgende TM M

M = Bei Eingabe $\langle R, y\rangle$, wobei R eine TM ist und y ein String ist:

1. Führe R auf y aus und halte ablehnend, falls die Ausgabe nicht die Form $\langle S, z\rangle$ hat
2. Führe S auf z aus und halte mit der Ausgabe von S auf dem Band

Sei $b = |\langle M\rangle| + 1$. Angenommen für einen beliebigen String x sei $d(x)$ b-komprimerbar. Dann gilt

$$|d(d(x))| \leq |d(x)| - b$$

Aber dann ist $\langle M\rangle d(d(x))$ eine Beschreibung von x mit einer maximalen Länge

$$|\langle M\rangle| + |d(d(x))| \leq (b - 1) + (|d(x)| - b) = |d(x)| - 1$$

Diese Beschreibung von x ist kürzer als $d(x)$, was ein Widerspruch zur Minimalität von $d(x)$ ist.