Einführung in die Informations- und Lerntheorie

0. Motivation und Ziele

- Wie extrahiert man aus einer großen Menge von Daten die relevante Information?
- Wie erklärt man die gewonnene Information?
- Welche unter den möglichen Erklärungen wählt man als "die richtige Erklärung" aus?

⋮

Die Erforschung von Information in Bezug auf ihre Generierung, ihre Extraktion und ihre Behandlung gehört zu den größten Herausforderungen unserer Zeit. Aufgrund der immer größer werdenden Datenmengen benötigt man Systeme, die Fragen wie die obigen automatisch beantworten. Derartige Systeme heißen <u>induktive Inferenzsysteme</u>.

<u>Ziel</u>:
Lernen der notwendigen Bausteine und Methoden zur Konstruktion von induktiven Inferenzsysteme

Einfacher als die Extraktion unbekannter Information aus Daten ist die Generierung von Daten zur Speicherung bekannter Information. Mit Problemen dieser Art beschäftigt sich die

sogenannte "klassische" Informationstheorie.
Diese hat ihren Ursprung in der Kommuni=
kationstheorie. Fundamental für die Kom=
munikationstheorie sind folgende zwei Fragen:

- Welche Datenkompression kann ohne In=
  formationsverlust erreicht werden?
- Welche Übertragungsrate kann bestenfalls
  bei der Datenübermittlung erreicht werden?

Mit beiden Fragestellungen hat sich die klassische
Informationstheorie intensiv beschäftigt und diese
auch abschließend behandelt. Dabei hat man
die Semantik einer Botschaft ignoriert und war
ausschließlich an dem Kommunikationsproblem
der Übermittlung einer Botschaft von einem
Sender zu einem Empfänger interessiert. Dies
geschah unter der Annahme, dass beide Be=
teiligten das Universum aller möglichen Bot=
schaften kennen. Zentral hierbei war die in=
formationstheoretische Quantität Entropie,
die diejenige Information misst, die benötigt
wird, damit der Empfänger aus dem be=
kannten Universum aller Botschaften die rich=
tige Botschaft auswählt. Die hierbei ent=
wickelten Methoden und Konzepte finden auch
bei der Konstruktion von induktiven Inferenz=
systemen ihre Anwendung. Daher werden wir

uns zunächst mit der klassischen Informations-theorie beschäftigen.

Einen anderen Weg geht die sogenannte "algo-rithmische" Informationstheorie. Diese definiert das Maß an Information nur in Abhängigkeit der individuellen Botschaft, also nicht in Relation zu einem gegeben Universum. Sie nimmt als Maß für die in einem String enthaltenen Information die Länge eines kürzesten Programms, das ohne Eingabe diesen String ausgibt. Die hier bei entwickelten Konzepte sind zentral bei der Konstruktion von induktiven Inferenzsysteme. Daher werden wir uns nach der klassischen mit der algorithmischen Informationstheorie beschäftigen.

Induktive Inferenzsysteme versuchen unter anderen Regelmäßigkeiten in Daten zu entdecken um dann hieraus Information zu gewinnen. Zufällige Daten enthalten keine solche Regelmäßigkeiten. Methoden und Konzepte, die bei der Charakterisierung von binären Zufallsfolgen entwickelt wurde finden direkt ihre Anwendung bei der Konstruktion von induktiven Inferenzsystemen. Aus diesem Grund werden wir uns nach der algorithmischen Informationstheorie mit der Charakterisierung von binären Zufallsfolgen beschäftigen.

Nun können wir uns endlich den induktiven Inferenzsystemen widmen. Ausgehend von einem sehr abstrakten universellen induktiven Inferenzsystems werden wir auf praktische induktive Inferenzsysteme hinarbeiten.

Literatur:

- Norbert Blum, Einführung in Formale Sprachen, Berechenbarkeit, Informations- und Lerntheorie, Oldenbourg 2007.
- Literatur, die ich bei der Konzeption der Vorlesung verwendet habe, werden zu Beginn der betreffenden Kapitel genannt.
- weitere Literatur in
  - Sonderfach der Vorlesung, Handapparat der Institutsbibliothek.

Hinweis auf:

Stasys Jukna, Crashkurs Mathematik für Informatiker, Teubner 2008.

1. Die klassische Informationstheorie

Literatur:

- Claude E. Shannon, Warren Weaver, The Mathematical Theory of Communication, University of Illinois Press, 1949.
- Norman Abramson, Information Theory and Coding, McGraw-Hill, 1963.
- Silviu Guiaşu, Information Theory with Applications, McGraw-Hill, 1977.
- Thomas M. Cover, Joy A. Thomas, Elements of Information Theory, John Wiley & Sons, 1991.

1.1 Die Entropie

Seien

E ein Ereignis

$p(E) > 0$ die Wahrscheinlichkeit, dass E eintritt.

Die Mitteilung, dass das Ereignis E eingetreten ist, enthält dann

$$I(E) := \log \frac{1}{p(E)}$$

Informationseinheiten.

Frage: Inwieweit entspricht obige Definition unserer Intuition?

- sicheres Ereignis, d.h., $p(E) = 1$:

  $I(E) = \log 1 = 0.$ ✓

- Je kleiner $p(E)$ umso größer ist $I(E)$. ✓

- $p(E) = \frac{1}{2} \Rightarrow I(E) = 1.$
  Ein Bit Information erhalten wir, wenn eine von zwei gleichwahrscheinlichen Alternativen, nämlich E und $\neg E$ spezifiziert wird. ✓

Zufallsexperiment:

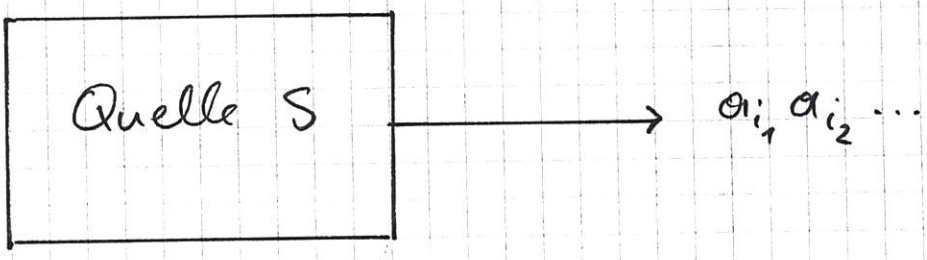
- n elementare Ereignisse $a_1, a_2, \dots, a_n$, die mit den Wahrscheinlichkeiten $p_1, p_2, \dots, p_n$ eintreten.
  Dabei gilt

  $p_i \geq 0$ für $1 \leq i \leq n$ und

  $\sum_{i=1}^{n} p_i = 1.$

$p(a_i)$ Wahrscheinlichkeit, dass a_i eintritt.

Beispiel: (gedächtnislose Informationsquelle)



Sukzessive Generierung von Symbolen
- gemäß Wahrscheinlichkeitsverteilung
$p(a_i) = p_i \quad , 1 \leq i \leq n$

- unabhängig von der bisher generierten Symbolfolge.

Falls das Symbol a_i generiert wird, dann erhalten wir

$$I(a_i) = \log \frac{1}{p_i}$$

Bits an Information.

$\Rightarrow$

<u>mittlere Information / Symbol</u>:

$$\sum_{i=1}^{n} p_i I(a_i) = \sum_{i=1}^{n} p_i \log \frac{1}{p_i}.$$

◆

$\rightsquigarrow$

Für eine endliche Wahrscheinlichkeitsverteilung $p = p_1, p_2, \dots, p_n$ definieren wir die <u>Entropie</u> H_n durch

$$H_n := H_n(p_1, p_2, \dots, p_n) := -\sum_{i=1}^{n} p_i \log p_i.$$

Dabei definieren wir $p_i \log p_i = 0$ falls $p_i = 0$.
Wegen $\lim_{x \to 0} x \log x = 0$ ist dies sinnvoll.

<u>Beispiel</u>:

$S = \{a_1, a_2, a_3\}$, $p(a_1) = \frac{1}{2}$, $p(a_2) = p(a_3) = \frac{1}{4}$.

$\rightsquigarrow$

$$H_3 = \frac{1}{2}\log 2 + \frac{1}{4}\log 4 + \frac{1}{4}\log 4$$

$$= \frac{3}{2}.$$

Interpretation:

- $I(a_i)$ diejenige Information, die für die Spezifikation, dass das Ereignis a_i eintritt, benötigt wird.

- $H_n(p_1, p_2, \dots, p_n)$
  - mittlere Informationsgröße pro Ausgang eines Zufallsexperimentes oder
  - mittlere Unsicherheit eines Beobachters vor Ausgang des Zufallsexperimentes.

Frage: Besitzt die Entropie diejenige Eigenschaften, die gemäß unserer Intuition mittlere Unsicherheit bzw. mittlere Information pro Ausgang eines Zufallsexperimentes haben sollte?

⇝

Lemma 1.1

Sei p_i, $1 \le i \le n$ eine Wahrscheinlichkeitsverteilung. Dann gilt:

a) $H_n(p_1, p_2, \dots, p_n) \geq 0$.

b) $p_j = 1$ und $p_i = 0$ für $i \neq j$ impliziert
$H_n(p_1, p_2, \dots, p_n) = 0$.

c) $H_{n+1}(p_1, p_2, \dots, p_n, 0) = H_n(p_1, p_2, \dots, p_n)$.

Beweis:

Übung. ∎

Erinnerung:

$\ln x := \log_e x$, wobei e die Eulersche Zahl ist.

Für $x > 0$ gilt:

(1.1) $\ln x \leq x - 1 \iff \ln \frac{1}{x} \geq 1 - x$.
Gleichheit gilt genau dann, wenn $x = 1$.

Übung:

Beweisen Sie

$\ln x < x - 1$ für $x > 0$, $x \neq 1$
$\ln x = x - 1$ für $x = 1$.

Mit Hilfe der Ungleichung (1.1) können wir eine weitere nützliche Ungleichung beweisen.

Seien $p_1, p_2, \dots, p_n$ und $q_1, q_2, \dots, q_n$ zwei Wahrscheinlichkeitsverteilungen. Wegen

$$\log_a x = \frac{1}{\log_b a} \log_b x$$

gilt

$$\sum_{i=1}^{n} p_i \log \frac{q_i}{p_i} = \frac{1}{\ln 2} \sum_{i=1}^{n} p_i \ln \frac{q_i}{p_i}.$$

Nach Anwendung von 1.1 auf jeden Summanden erhalten wir:

$$\begin{aligned} \sum_{i=1}^{n} p_i \log \frac{q_i}{p_i} &\leq \frac{1}{\ln 2} \sum_{i=1}^{n} p_i \left(\frac{q_i}{p_i} - 1 \right) \\ &= \frac{1}{\ln 2} \left(\sum_{i=1}^{n} q_i - \sum_{i=1}^{n} p_i \right) \\ &= 0. \end{aligned}$$

$$\Leftrightarrow \sum_{i=1}^{n} p_i \log \frac{1}{p_i} + \sum_{i=1}^{n} p_i \log q_i \leq 0$$

$$\Leftrightarrow \sum_{i=1}^{n} p_i \log \frac{1}{p_i} \leq \sum_{i=1}^{n} p_i \log \frac{1}{q_i} \qquad (1.2)$$

In der Ungleichung 1.2 gilt genau dann Gleichheit, wenn $p_i = q_i$ für $1 \leq i \leq n$.

⇝

Lemma 1.2

Sei p_i, $1 \le i \le n$ eine Wahrscheinlichkeitsverteilung. Dann gilt:

a) $\log n \geq H_n(p_1, p_2, \dots, p_n)$

b) $\log n = H_n(p_1, p_2, \dots, p_n)$ genau dann, wenn $p_i = \frac{1}{n}$ für $1 \le i \le n$.

Beweis:

Als erstes werden wir beweisen, dass $\log n$ stets eine obere Schranke für die Entropie ist.

$$\begin{aligned}
&\log n - H_n(p_1, p_2, \dots, p_n) \\
&= \sum_{i=1}^{n} p_i \log n + \sum_{i=1}^{n} p_i \log p_i \\
&= \sum_{i=1}^{n} p_i \log n \cdot p_i \\
&= \log e \sum_{i=1}^{n} p_i \ln n p_i .
\end{aligned}$$

Anwendung von 1.1 ergibt:

$$\begin{aligned}
(*)\quad \log n - H_n(p_1, p_2, \dots, p_n) &\geq \log e \sum_{i=1}^{n} p_i \left(1 - \frac{1}{n p_i}\right) \\
&= \log e \left(\sum_{i=1}^{n} p_i - \frac{1}{n} \sum_{i=1}^{n} \frac{p_i}{p_i} \right) \\
&= 0.
\end{aligned}$$

Also gilt $\log n \geq H_n(p_1, p_2, \dots, p_n)$.

Obige Ungleichung (*) haben wir durch Anwendung von

$$\ln \frac{1}{x} \geq 1 - x$$

erhalten. Wegen

$$\ln \frac{1}{x} = 1 - x \iff x = 1$$

ist die Ungleichung (*) genau dann nicht strikt, wenn nicht $p_i = \frac{1}{n}$ für $1 \leq i \leq n$.

$\Rightarrow$

$H_n(p_1, p_2, \ldots, p_n)$ nimmt genau dann den Wert $\log n$ an, wenn $p_i = \frac{1}{n}$ für $1 \leq i \leq n$. ■

Frage:
Was ist die Entropie eines Zufallsexperimentes, das durch die Kombination zweier Zufallsexperimente entsteht? Entspricht diese stets unserer Intuition über die mittlere Unsicherheit bezüglich des Ausganges zweier kombinierten Zufallsexperimente?

Ziel:
Beantwortung dieser Frage.

Seien

- A und B zwei Zufallsexperimente

mit elementaren Ereignissen

$a_1, a_2, \dots, a_n$ bzw. $b_1, b_2, \dots, b_m$.

Das kombinierte Zufallsexperiment $A \otimes B$ ist dann die Realisierung beider Zufallsexperimente A und B.

Die elementare Ereignisse des Zufallsexperimentes $A \otimes B$ sind gerade die Paare

$$(a_k, b_\ell), \quad 1 \le k \le n, \quad 1 \le \ell \le m.$$

$\pi_{k\ell} := p(a_k, b_\ell)$ Wahrscheinlichkeit, dass das Ereignis (a_k, b_ℓ) eintritt.

$\Rightarrow$

$$(1.3)\quad H_{nm}(A \otimes B) = -\sum_{k=1}^{n} \sum_{\ell=1}^{m} \pi_{k\ell} \log \pi_{k\ell}$$

Somit ergeben sich folgende Wahrscheinlichkeiten:

a) Die Wahrscheinlichkeit

$$(1.4)\quad p_k = \sum_{\ell=1}^{m} \pi_{k\ell}$$

für das Ereignis a_k im ersten Experiment, unabhängig vom Ausgang des zweiten Experimentes.

b) Die Wahrscheinlichkeit

$$(1.5) \qquad q_\ell = \sum_{k=1}^{n} \pi_{k\ell}$$

für das Ereignis b_ℓ im zweiten Experiment, unabhängig vom Ausgang des ersten Experimentes

c) Für $q_\ell > 0$ die Wahrscheinlichkeit

$$(1.6) \qquad p_{\ell k} = \frac{\pi_{k\ell}}{q_\ell}$$

für das Ereignis a_k im Experiment A unter der Voraussetzung, dass im Experiment B das Ereignis b_ℓ eintritt.

d) Für $p_k > 0$ die Wahrscheinlichkeit

$$(1.7) \qquad q_{k\ell} = \frac{\pi_{k\ell}}{p_k}$$

für das Ereignis b_ℓ im Experiment B unter der Voraussetzung, dass im Experment A das Ereignis a_k eintritt.

Falls im Experiment A das Ereignis a_k mit Wahrscheinlichkeit $p_k > 0$ eintritt, dann erhalten wir somit die <u>bedingte Entropie</u> $H_m(B|a_k)$ des Experimentes B unter der Annahme, dass im Experiment A das Ereignis a_k eintritt, durch

$$(1.8) \qquad H_m(B|a_k) = -\sum_{\ell=1}^{m} q_{k\ell} \log q_{k\ell}.$$

Falls $p_k = 0$, dann gilt $H_m(B|a_k) = 0$.

Die durch das Experiment A bedingte Entropie $H_m(B|A)$ des Experimentes B ist dann

$$H_m(B|A) = \sum_{k=1}^{n} p_k H_m(B|a_k)$$

$$(1.9) \qquad = -\sum_{k=1}^{n}\sum_{\ell=1}^{m} p_k q_{k\ell} \log q_{k\ell}.$$

Analog erhalten wir

$$(1.10) \qquad H_n(A|b_\ell) = -\sum_{k=1}^{n} p_{\ell k} \log p_{\ell k}$$

$$(1.11) \qquad H_n(A|B) = -\sum_{k=1}^{n}\sum_{\ell=1}^{m} q_\ell p_{\ell k} \log p_{\ell k}.$$

Ziel:
Beweis einiger Eigenschaften der oben definierten Entropien.

Lemma 1.3

$$H_{nm}(A \otimes B) = H_n(A) + H_m(B|A)$$

$$= H_m(B) + H_n(A|B)$$

Beweis:

Gemäß 1.3, 1.7 und 1.9 gilt

$$H_{nm}(A \otimes B) \underset{(1.3)}{=} -\sum_{k=1}^{n}\sum_{\ell=1}^{m} \pi_{k\ell} \log \pi_{k\ell}$$

$$\underset{(1.7)}{=} -\sum_{k=1}^{n}\sum_{\ell=1}^{m} p_k q_{k\ell} \log(p_k q_{k\ell})$$

$$= -\sum_{k=1}^{n} p_k \underbrace{\left(\sum_{\ell=1}^{m} q_{k\ell}\right)}_{=1} \log p_k$$

$$-\sum_{k=1}^{n}\sum_{\ell=1}^{m} p_k q_{k\ell} \log q_{k\ell}$$

$$\underset{(1.3)}{=} H_n(A) + H_m(B|A).$$

Die andere Gleichung beweist man analog. ∎

Übung:

Beweisen Sie $H_{nm}(A \otimes B) = H_m(B) + H_n(A|B)$.

Lemma 1.3 impliziert direkt folgendes Lemma:

Lemma 1.4

Für zwei Zufallsexperimente A und B gilt

$$H_n(A) - H_n(A|B) = H_m(B) - H_m(B|A).$$

Seien A und B zwei unabhängige Zufallsexperimente. Dann gilt:

$\pi_{kl} = p_k \cdot q_l$, $q_{kl} = q_l$, $p_{kl} = p_k$

Somit erhalten wir für unabhängige Zufalls-experimente

$$H_m(B|A) = H_m(B) \text{ und } H_n(A|B) = H_n(A)$$

Also folgt aus Lemma 1.3 direkt:

Lemma 1.5

Seien A und B unabhängige Zufallsexperimente Dann gilt

$$H_{nm}(A \otimes B) = H_n(A) + H_m(B).$$

Frage:

Gemäß unserer Intuition kann zusätzliches Wissen nur nützlich sein. Folgt die Entropie dieser Intuition?

Folgendes Lemma beantwortet diese Frage positiv:

Lemma 1.6

Für zwei beliebige Zufallsexperimente A und B gilt

$$H_m(B|A) \leq H_m(B).$$

Wir beweisen das Lemma unter Verwendung von "Jensen's Ungleichung". Hierzu benötigen wir zunächst einige Notationen.

Eine Funktion $f(x)$ heißt <u>konvex</u> über ein Intervall $[a,b]$, falls für alle $x_1, x_2 \in [a,b]$ und alle $0 \leq \lambda \leq 1$

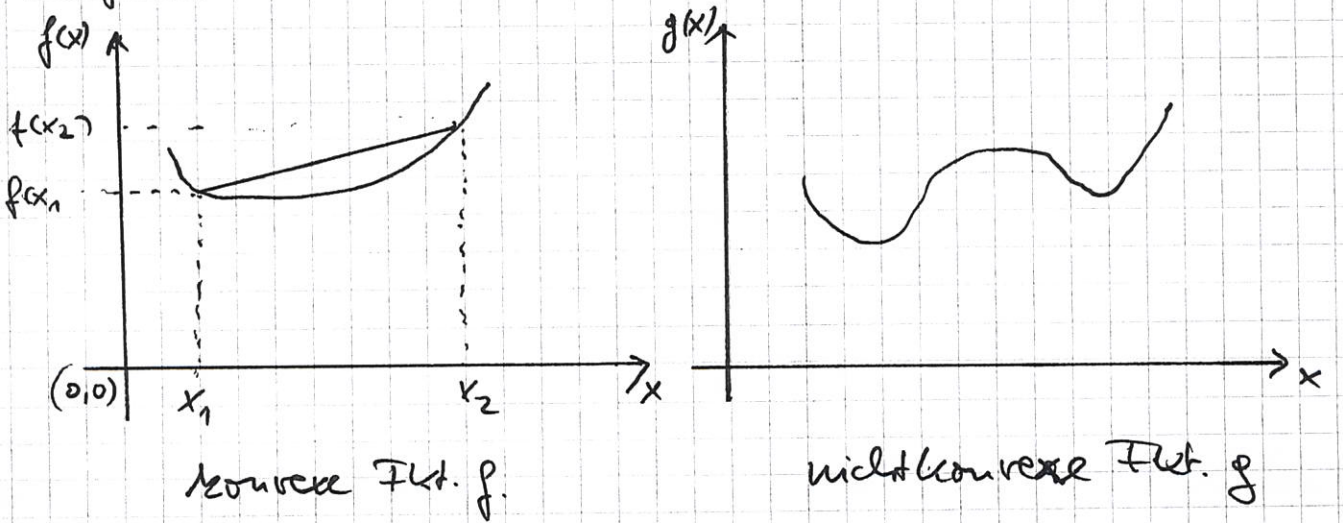
$$f(\lambda x_1 + (1-\lambda) x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2).$$

f heißt <u>strikt konvex</u> falls in obiger Ungleichung nur in den Fällen $\lambda = 0$ und $\lambda = 1$ die Gleichheit gilt. Eine Funktion f heißt (<u>strikt</u>) <u>konkav</u>, falls die Funktion $-f$ (strikt) konvex ist.

<u>geometrische Interpretation</u>:

Alle Punkte auf der Verbindungslinie zweier beliebiger Punkte $(x_1, f(x_1))$ und $(x_2, f(x_2))$ des Graphen einer konvexen Funktion f liegen auf oder oberhalb des Graphen von f.

<u>Beispiel:</u>



Übung:
Beweisen Sie, dass die Logarithmusfunktion konkav ist.

Satz 1.1 (Jensen's Ungleichung)

Sei $f: [a,b] \to \mathbb{R}$ eine stetige konvexe Funktion. Dann gilt für alle $n \in \mathbb{N}$, $x_1, x_2, \dots, x_n \in [a,b]$ und $\lambda_1, \lambda_2, \dots, \lambda_n \in [0,1]$ mit $\sum_{k=1}^{n} \lambda_k = 1$

$$f\left(\sum_{k=1}^{n} \lambda_k x_k\right) \leq \sum_{k=1}^{n} \lambda_k f(x_k).$$

Beweis:
Wir beweisen den Satz durch Induktion über n.

$n = 2$:
Dann folgt wegen $\lambda_2 = 1 - \lambda_1$ die Behauptung direkt aus der Definition von konvexen Funktionen

Annahme:
Die Behauptung ist für $n = \ell \geq 2$ wahr.

$\ell \rightsquigarrow \ell+1$:

Betrachte $\lambda_1, \lambda_2, \dots, \lambda_{\ell+1} \in [0,1]$ mit $\sum_{k=1}^{\ell+1} \lambda_k = 1$.

Falls $\lambda_{\ell+1} = 1$, dann ist die Behauptung trivialerweise erfüllt. Nehmen wir also $\lambda_{\ell+1} \neq 1$ an.

Für $k = 1, 2, \dots, \ell$ definieren wir

$$\lambda_k' := \frac{\lambda_k}{1 - \lambda_{\ell+1}}$$

Dann gilt:

$$f\left(\sum_{k=1}^{\ell+1} \lambda_k x_k\right) = f\left(\lambda_{\ell+1} x_{\ell+1} + (1-\lambda_{\ell+1}) \sum_{k=1}^{\ell} \lambda_k' x_k\right)$$

$$\leq \lambda_{\ell+1} f(x_{\ell+1}) + (1-\lambda_{\ell+1}) f\left(\sum_{k=1}^{\ell} \lambda_k' x_k\right)$$

Def. konvexe Fkt.

$$\leq \lambda_{\ell+1} f(x_{\ell+1}) + (1-\lambda_{\ell+1}) \sum_{k=1}^{\ell} \lambda_k' f(x_k)$$

Ind. ann.

$$= \sum_{k=1}^{\ell+1} \lambda_k f(x_k).$$

■

Aus obigem Satz folgt direkt folgendes Korollar:

<u>Korollar 1.1</u>

Sei $f: [a,b] \to \mathbb{R}$ eine stetige konkave Funktion. Dann gilt für alle $n \in \mathbb{N}$, $x_1, x_2, \dots, x_n \in [a,b]$ und $\lambda_1, \lambda_2, \dots, \lambda_n \in [0,1]$ mit $\sum_{k=1}^{n} \lambda_k = 1$

$$f\left(\sum_{k=1}^{n} \lambda_k x_k\right) \geq \sum_{k=1}^{n} \lambda_k f(x_k).$$

Nun ist der Beweis des Lemmas 1.6 einfach.

<u>Übung:</u>

Zeigen Sie, dass die Funktion $f(x) = -x \log x$ stetig und konkav ist.

Beweis (Lemma 1.6):

Die Anwendung des Korollars 1.1 mit

$$a = 0, \quad b = 1, \quad f(x) = -x \log x$$
$$\lambda_k = p_k \text{ und } x_k = q_{k\ell}$$

ergibt für jedes $1 \le \ell \le m$:

$$-\sum_{k=1}^{n} p_k q_{k\ell} \log q_{k\ell} \le -\left(\sum_{k=1}^{n} p_k q_{k\ell}\right) \log \left(\sum_{k=1}^{n} p_k q_{k\ell}\right)$$
$$\underset{1.7}{=} -\left(\sum_{k=1}^{n} \pi_{k\ell}\right) \log \left(\sum_{k=1}^{n} \pi_{k\ell}\right)$$
$$\underset{1.5}{=} -q_\ell \log q_\ell .$$

Wegen 1.9 erhalten wir

$$H_m(B|A) = -\sum_{k=1}^{n} \sum_{\ell=1}^{m} p_k q_{k\ell} \log q_{k\ell}$$
$$\le -\sum_{\ell=1}^{m} q_\ell \log q_\ell$$
$$= H_m(B).$$

■

Bemerkung:

- A und B unabhängige Zufallsexperimente
  $\Rightarrow$ in Lemma 1.6 gilt Gleichheit.
- Ausgang des Experimentes A bestimmt eindeutig den Ausgang des Experimentes B.
  $\rightsquigarrow$ Dann müsste $H_m(B|A) = 0$ sein.

Übung:
Zeigen Sie, dass $H_m(B|A) = 0$, falls der Ausgang des Experimentes A eindeutig den Ausgang des Experimentes B bestimmt.

Wegen Lemma 1.3 gilt dann auch

$$H_{nm}(A \otimes B) = H_n(A).$$

Betrachten wir die Quantität

$$H_n(A) - H_n(A|B) \underset{\text{Le. 1.3}}{=} H_n(A) + H_m(B) - H_{nm}(A \otimes B)$$

$H_n(A)$ ist die mittlere Information pro Ausgang des Experimentes A.

$H_n(A|B)$ ist die mittlere Information pro Ausgang des Experimentes A unter Kenntnis des Ausganges des Experimentes B.

Also ist

$$I(A,B) := H_n(A) - H_n(A|B)$$

diejenige Information, die in B über A enthalten ist.

Aus Lemma 1.3 folgt direkt, dass $I(A,B)$ auch diejenige Information ist, die in A über B enthalten ist.

Frage:
Ist die Entropie die einzige Funktion, die als Maß der Unsicherheit über den Ausgang eines Zufallsexperimentes stets unserer Intuition folgt?

Zur Beantwortung dieser Frage stellen wir einige Axiome, die eine Funktion für das Maß der Unsicherheit über den Ausgang eines Zufallsexperimentes erfüllen sollte, auf und beweisen, dass alle Funktionen, die diese Axiome erfüllen, als Produkt der Entropie mit einer positiven Konstanten geschrieben werden können.

Satz 1.2 (Eindeutigkeitssatz für die Entropie)

Sei $H_1(1)$, $H_2(p_1,p_2)$, ..., $H_n(p_1,p_2,\dots,p_n)$, ... eine Folge von Funktionen, so dass für jedes n die nichtnegative Funktion $H_n(p_1,p_2,\dots,p_n)$ auf der Menge $\{(p_1,p_2,\dots,p_n) \mid p_i \geq 0, \sum_{i=1}^{n} p_i = 1\}$ definiert ist. Ferner seien folgende Axiome erfüllt:

(A1) Für jedes n ist die Funktion $H_n(p_1,p_2,\dots,p$ eine bezüglich all ihrer Argumente stetige und symmetrische Funktion.

(A2) Für jedes n gilt

$$H_{n+1}(p_1,p_2,\dots,p_n,0) = H_n(p_1,p_2,\dots,p_n).$$

(A3) Für jedes n gilt

$$H_n(p_1, p_2, \dots, p_n) \leq H_n\left(\frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n}\right)$$

und Gleichheit genau dann, wenn $p_i = \frac{1}{n}$ für $1 \leq i \leq n$.

(A4) Falls $\pi_{kl} \geq 0$ für $1 \leq k \leq n$, $1 \leq l \leq m$, $\sum_{k=1}^{n} \sum_{l=1}^{m} \pi_{kl} = 1$ und $p_k = \sum_{l=1}^{m} \pi_{kl}$, dann gilt

$$H_{nm}(\pi_{11}, \dots, \pi_{nm}) = H_n(p_1, p_2, \dots, p_n) + \sum_{k=1}^{n} p_k \, H_m\left(\frac{\pi_{k1}}{p_k}, \dots, \frac{\pi_{km}}{p_k}\right).$$

Dann gilt für alle n

$$H_n(p_1, p_2, \dots, p_n) = -c \sum_{k=1}^{n} p_k \log p_k,$$

wobei c eine positive Konstante ist.

Bemerkung:
Dass die Entropie obige Axiome erfüllt, haben wir bereits bewiesen. Das vierte Axiom korrespondiert zu zusammengesetzten Zufallsexperimenten.

Beweis:
Wir beweisen den Satz zunächst für den Spezialfall $p_k = \frac{1}{n}$, $1 \leq k \leq n$, dann für be-

liebige rationale Wahrscheinlichkeiten und schließlich für den allgemeinen Fall.

Für $n \in \mathbb{N}$ seien

- $p_k = \frac{1}{n}$ für $1 \leq k \leq n$ und

$$L(n) := H_n\left(\tfrac{1}{n}, \tfrac{1}{n}, \dots, \tfrac{1}{n}\right).$$

Wegen Lemma 1.2 gilt

$$-\sum_{k=1}^{n} \frac{1}{n} \log \frac{1}{n} = \log n.$$

Somit haben wir folgendes zu beweisen:

$$L(n) = c \log n, \text{ für eine Konstante } c > 0.$$

Für alle $c > 0$ ist die Funktion $f(n) = c \cdot \log n$ streng monoton wachsend und es gilt

$$f(n^r) = r \cdot f(n).$$

Zunächst werden wir beweisen, dass die Funktion $L(n)$ auch diese beiden Eigenschaften besitzt. Danach werden wir mit Hilfe dieser beiden Eigenschaften

$$L(n) = c \log n, \text{ für eine Konstante } c > 0$$

beweisen. Es gilt

$$L(n) = H_n\left(\tfrac{1}{n}, \tfrac{1}{n}, \dots, \tfrac{1}{n}\right)$$

$$\underset{A2}{=} H_{n+1}\left(\tfrac{1}{n}, \tfrac{1}{n}, \dots, \tfrac{1}{n}, 0\right)$$

$$\overset{<}{\underset{A3}{}} H_{n+1}\left(\frac{1}{n+1}, \frac{1}{n+1}, \dots, \frac{1}{n+1}\right)$$

$$= L(n+1).$$

Also ist die Funktion $L(n)$ streng monoton wachsend.

Wir beweisen (1.12) $L(n^r) = r \cdot L(n)$ $\forall\, r, n \in \mathbb{N}$ mittels Induktion über r.

$\underline{r = 1}$: trivial

$\underline{\text{Annahme}}$:

Die Behauptung ist für r, $r \geq 1$ wahr.

$\underline{r \rightsquigarrow r+1}$:

Anwendung des Axioms A4 für

$$m = n^r, \quad \pi_{kl} = \frac{1}{n^{r+1}} \quad \text{für } 1 \leq k \leq n,\ 1 \leq l \leq m$$

und $p_k = \frac{1}{n}$

ergibt

$$H_{n^{r+1}}\left(\frac{1}{n^{r+1}}, \frac{1}{n^{r+1}}, \dots, \frac{1}{n^{r+1}}\right)$$

$$= H_n\left(\frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n}\right) + \sum_{k=1}^{n} \frac{1}{n} H_{n^r}\left(\frac{1}{n^r}, \dots, \frac{1}{n^r}\right.$$

$$= L(n) + L(n^r)$$

$$= L(n) + r\,L(n)$$

$$\overset{\text{Ind. Ann}}{=} (r+1) \cdot L(n).$$

Um zu zeigen, dass

$$L(n) = c \cdot \log n \text{ für eine Konstante } c > 0$$

wählen wir $v, r, t \in \mathbb{N}$ derart, dass ein $s \in \mathbb{N}$ mit

$$r^s \leq v^t \leq r^{s+1} \tag{1.13}$$

existiert.

Übung:
Charakterisieren Sie die möglichen Wahlen von $v, r, t \in \mathbb{N}$, so dass ein $s \in \mathbb{N}$ mit $r^s \leq v^t \leq r^{s+1}$ existiert.

Logarithmieren der Ungleichungen (1.13) ergibt

$$s \log r \leq t \log v \leq (s+1) \log r$$

$$\Leftrightarrow \quad \frac{s}{t} \leq \frac{\log v}{\log r} \leq \frac{s+1}{t}. \tag{1.14}$$

Da L monoton wachsend ist folgt aus (1.13)

$$L(r^s) \leq L(v^t) \leq L(r^{s+1})$$

$$\overset{(1.12)}{\Leftrightarrow} \quad s \cdot L(r) \leq t \cdot L(v) \leq (s+1) L(r)$$

$$\Leftrightarrow \quad \frac{s}{t} \leq \frac{L(v)}{L(r)} \leq \frac{s+1}{t}. \tag{1.15}$$

Aus 1.14 und 1.15 folgt

$$\left| \frac{L(v)}{L(r)} - \frac{\log v}{\log r} \right| \leq \frac{1}{t}.$$

Da die linke Seite der obigen Ungleichung unabhängig von t ist und wir t beliebig groß wählen können, gilt

$$\frac{L(v)}{L(r)} = \frac{\log v}{\log r}$$

$$\Rightarrow \frac{L(v)}{\log v} = \frac{L(r)}{\log r}.$$

Da wir v und r "nahezu" beliebig wählen können, folgt somit

$$L(n) = c \log n,$$

wobei c eine Konstante ist. Da $L(n)$ streng monoton wachsend ist, gilt $c > 0$.

Insgesamt haben wir die Behauptung für den Spezialfall $p_k = \frac{1}{n}$, $1 \leq k \leq n$ $\forall n \in \mathbb{N}$ bewiesen.

Nehmen wir nun an, dass alle p_k rationale Zahlen sind. Wegen Axiom A2 können wir o.B.d.A. annehmen, dass alle p_k größer null sind. D.h.,

$$p_k = \frac{m_k}{m} \quad \text{für} \quad 1 \leq k \leq n,$$

wobei $m_k \in \mathbb{N}$, $1 \leq k \leq n$ und $m = \sum_{k=1}^{n} m_k$.

Wir beweisen die Behauptung, indem wir geeignete π_{kl}, $1 \leq k \leq n$, $1 \leq l \leq m$ wählen und das Axiom A4 anwenden. Sei stets

$$(1.16) \qquad q_{kl} := \frac{\pi_{kl}}{p_k}$$

Anstatt π_{kl} definieren wir q_{kl}. Der Wert von π_{kl} ergibt sich dann wegen 1.16 aus

$$\pi_{kl} = q_{kl} p_k .$$

Betrachten wir folgende Werte für q_{kl}:

$$(1.17) \qquad q_{kl} := \begin{cases} 0 & \text{falls } 1 \leq l \leq \sum_{s=1}^{k-1} m_s \\ \frac{1}{m_k} & \text{falls } \left(\sum_{s=1}^{k-1} m_s\right) + 1 \leq l \leq \sum_{s=1}^{k} m_s \\ 0 & \text{falls } \left(\sum_{s=1}^{k} m_s\right) + 1 \leq l \leq m \end{cases}$$

Dann erhalten wir folgende Werte für π_{kl}:

$$(1.18) \qquad \pi_{kl} := \begin{cases} 0 & \text{falls } 1 \leq l \leq \sum_{s=1}^{k-1} m_s \\ \frac{1}{m} & \text{falls } \left(\sum_{s=1}^{k-1} m_s\right) + 1 \leq l \leq \sum_{s=1}^{k} m_s \\ 0 & \text{falls } \left(\sum_{s=1}^{k} m_s\right) + 1 \leq l \leq m \end{cases}$$

Mittels Anwendung des Axioms A4 für diese Werte erhalten wir:

$$H_{nm}(\pi_{11}, \dots, \pi_{nm}) = H_n(p_1, \dots, p_n) + \sum_{k=1}^{n} p_k H_m\left(\frac{\pi_{k1}}{p_k}, \dots, \frac{\pi_{km}}{p_k}\right)$$

$\Longleftrightarrow$
1.18 u. A2

$$H_m\left(\frac{1}{m}, \frac{1}{m}, \dots, \frac{1}{m}\right) = H_n(p_1, \dots, p_n) + \sum_{k=1}^{n} p_k H_m\left(\frac{\pi_{k1}}{p_k}, \dots, \frac{\pi_{km}}{p_k}\right)$$

$\Longleftrightarrow$
1.17 u. A2

$$H_m\left(\frac{1}{m}, \frac{1}{m}, \dots, \frac{1}{m}\right) = H_n(p_1, \dots, p_n) + \sum_{k=1}^{n} p_k H_{m_k}\left(\frac{1}{m_k}, \dots, \frac{1}{m_k}\right)$$

$\Longleftrightarrow$

$$L(m) = H_n(p_1, \dots, p_n) + \sum_{k=1}^{n} p_k L(m_k)$$

Oben haben wir gezeigt, dass $L(n) = c \log n$ für alle $n \in \mathbb{N}$. Also gilt:

$$c \cdot \log m = H_n(p_1, \dots, p_n) + c \cdot \sum_{k=1}^{n} p_k \log m_k$$

$$= H_n(p_1, \dots, p_n) + c \cdot \sum_{k=1}^{n} p_k \log m \cdot p_k$$

$$= H_n(p_1, \dots, p_n) + c \cdot \sum_{k=1}^{n} p_k \log m + c \sum_{k=1}^{n} p_k \log p_k$$

Nach Auflösung der obigen Gleichung nach $H_n(p_1, \dots, p_n)$ erhalten wir

$$(1.19) \qquad H_n(p_1, \dots, p_n) = -c \sum_{k=1}^{n} p_k \log p_k$$

Damit haben wir die Behauptung für rationale p_k, $1 \le k \le n$ bewiesen.

Aufgrund der in Axiom A1 geforderten Stetigkeit der Funktionen H_n, $n \in \mathbb{N}$ gilt die für rationale Wahrscheinlichkeiten bewiesene Gleichung 1.19 auch für beliebige Wahrscheinlichkeitsverteilungen.

■

1.2 Einführung in die Kodierungstheorie

Wir werden uns nun mit der Kodierung von Information beschäftigen. Zunächst benötigen wir einige Definitionen.

Sei Σ ein Alphabet. Eine <u>binäre Kodierung</u> φ von Σ ist eine injektive Abbildung $\varphi : \Sigma \to \{0,1\}^+$. $\varphi(\Sigma)$ ist dann ein <u>Binärkode</u> für Σ. Ein Binärkode heißt <u>präfixfrei</u>, wenn $\varphi(a)$ kein Präfix von $\varphi(b)$ ist, für beliebige $a, b \in \Sigma$, $a \neq b$. Sei $x := a_1 a_2 \dots a_n \in \Sigma^+$ ein Text. Dann kodieren wir den Text x durch

$$\psi(x) := \psi(a_1)\psi(a_2)\cdots\psi(a_n).$$

Ein präfixfreier Binärkode $\psi(\Sigma)$ heißt <u>optimal</u> für den Text $x \in \Sigma^+$, falls $\psi(x)$ minimale Länge hat. D.h.,

$$|\psi(x)| = \min\{\, |\psi'(x)| \mid \psi'(\Sigma) \text{ ist ein präfix-freier Binärkode für } \Sigma.\}$$

Ein präfixfreier Binärkode ψ (oder kurz <u>Präfix kode</u>) für Σ kann durch einen Binärbaum $T_\Sigma(\psi)$ derart repräsentiert werden, dass gilt:

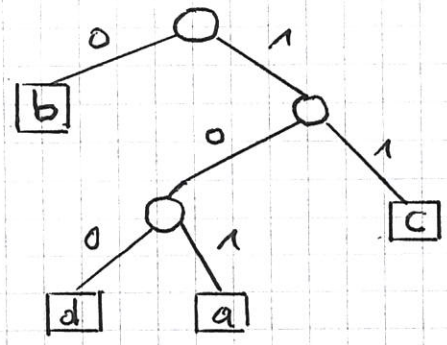
1. $T_\Sigma(\psi)$ besitzt $|\Sigma|$ Blätter, die jeweils mit einem paarweise verschiedenen Symbol aus Σ beschriftet sind.

2. Wenn wir die linke ausgehende Kante eines inneren Knotens mit 0 und die rechte ausgehende mit 1 beschriften, dann ist $\psi(a)$ für alle $a \in \Sigma$ gleich der Beschriftung des Pfades von der Wurzel zu dem Blatt, das a enthält.

$T_\Sigma(\psi)$ ist ein Trie, der eine präfixfreie Menge $S \subset \{0,1\}^+$ repräsentiert. Dabei gilt $|S| = |\Sigma|$ und jedes Element in S korrespondiert zu einem paarweise verschiedenen Element in Σ.

Beispiel:

Seien $\Sigma := \{a, b, c, d\}$, $\varphi(a) := 101$, $\varphi(b) :=$ $\varphi(c) := 11$ und $\varphi(d) := 100$



Die umgekehrte Richtung gilt auch:

binärer Trie $\hat{=}$ präfixfreien Binärkode.

Betrachte $\Sigma' = \{a, b, c, d\}$. Der Binärkode $\{0, 01, 10, 11\}$ wäre nicht präfixfrei. Mehr noch, es existiert für Σ' kein Präfixkode mit Kodewortlängen 1, 2, 2, 2. Die berühmte Kraft-Ungleichung charakterisiert exakt, für welche vorgegebene Folgen von Kodewortlängen ein Präfixkode existiert.

Satz 1.3 (Kraft 1949)

Sei $\ell_1, \ell_2, \ldots, \ell_n$ eine Folge von natürlichen Zahlen. Es gibt genau dann einen präfixfreien Binärkode mit Kodewortlängen $\ell_1, \ell_2, \ldots, \ell_n$, wenn

$$\sum_{i=1}^{n} 2^{-\ell_i} \leq 1.$$

Beweis:

„⇒"

Oben haben wir uns überlegt, dass jeder Präfixkode zu einem binären Trie korrespondiert. Dabei entsprechen die Kodewortlängen den Tiefen der Blätter im Trie. Also genügt es zu zeigen, dass für jeden binären Trie T mit n Blättern

$$S(T) := \sum_{i=1}^{n} 2^{-\ell_i} \leq 1, \qquad (1.20)$$

wobei ℓ_i die Tiefe des i-ten Blattes bezeichnet.

Beweis mittels Induktion über die maximale Tiefe t_{max} eines Blattes in T:

$t_{max} = 0$:

Dann besteht T nur aus einem Blatt der Tiefe 0. Es gilt:

$$S(T) = 2^{-0} = 1. \quad \checkmark$$

$t_{max} = 1$:

Dann besitzt T entweder ein oder zwei Blätter der Tiefe 1. Also gilt:

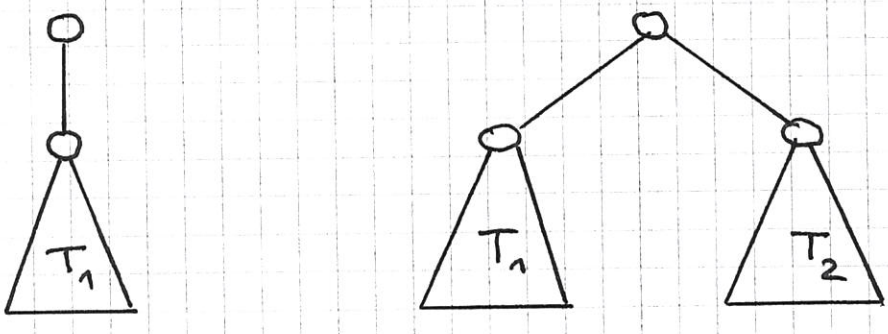
$$S(T) \leq \sum_{i=1}^{2} 2^{-1} = 2 \cdot \tfrac{1}{2} = 1 \quad \checkmark$$

Annahme:

(1.20) ist erfüllt für alle Tries mit $t_{max} \leq k$, $k \geq 1$.

$k \rightsquigarrow k+1$:

Sei T ein beliebiger Trie mit $t_{max} = k+1$. Dann besitzt T eines der folgenden beiden Aussehen:



In T_i, $i \in \{1,2\}$ ist die maximale Tiefe eines Blattes $\leq k$, so dass auf jeden der Unterbäume die Induktionsannahme anwendbar ist.

Da die Tiefe der Blätter in T_i, $i \in \{1,2\}$ exakt um eins geringer ist als in T, erhalten wir

$$S(T) = \frac{1}{2}\left(S(T_1) + S(T_2)\right)$$
$$\leq \frac{1}{2}(1+1) = 1. \quad \checkmark$$

„$\Leftarrow$"

Es genügt zu zeigen, dass für jede Folge $L = l_1, l_2, \ldots, l_n$ mit $\sum_{i=1}^{n} 2^{-l_i} \leq 1$ ein binärer Trie T mit n Blättern und $S(T) = \sum_{i=1}^{n} 2^{-l_i}$ existiert. T heißt dann L-passend.

Wir beweisen dies mittels Induktion über den maximalen Wert l_{max} in der Folge L. O.B.d.A können wir $l_1 \leq l_2 \leq \ldots \leq l_n$ annehmen

$\ell_{max} = 1$:

Wegen $\sum_{i=1}^{n} 2^{-\ell_i} \leq 1$ gilt $L = 1$ oder $L = 1,1$, so dass folgende Tries L-passend sind:

Annahme:
Für jede Folge L mit $\ell_{max} \leq k$, $k \geq 1$ existiert ein L-passender Trie.

$k \rightsquigarrow k+1$:

Idee:
Wir teilen L in zwei Teilfolgen L_1 und L_2 auf und reduzieren in jeder dieser Folgen die Werte um eins. Dann wenden wir auf jede der sich ergebenden Teilfolgen die Induktionsannahme an und bauen aus den sich resultierenden Tries einen Trie für L.

Frage:
Wo soll die Folge L in zwei Teilfolgen L_1 und L_2 aufgeteilt werden?

Folgendes Lemma beantwortet diese Frage:

Lemma 1.7

Sei $L = \ell_1 \leq \ell_2 \leq \dots \leq \ell_n$ eine Folge von n natürlichen Zahlen mit $\sum_{i=1}^{n} 2^{-\ell_i} \leq 1$. Dann existiert $p \in \{1, 2, \dots, n\}$ mit $\sum_{i=1}^{p} 2^{-\ell_i} = \frac{1}{2}$ oder es gilt $\sum_{i=1}^{n} 2^{-\ell_i} < \frac{1}{2}$.

Bevor wir obiges Lemma beweisen, führen wir den Beweis des Satzes zu Ende.

Setze

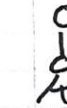
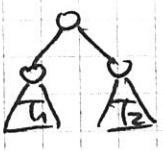
$$\begin{cases} L_1 = \ell_1, \ell_2, \dots, \ell_p \\ L_2 = \ell_{p+1}, \dots, \ell_n \end{cases} \quad \text{falls } p \text{ existiert}$$

$$L_1 = L \quad \text{sonst.}$$

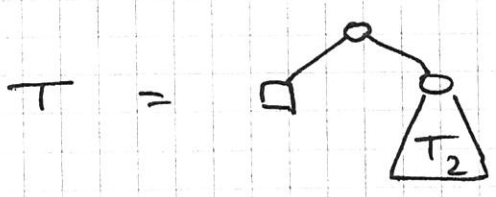
Annahme: $L_1 \neq 1$.

Den Sonderfall $L_1 = 1$ behandeln wir ausschließen

Für $i \in \{1, 2\}$ reduzieren wir in L_i alle Werte um eins. Sei L_i' die resultierende Liste. Dann ist der maximale Wert in L_i' sicher $\leq k$, so dass auf L_i' die Induktionsannahme anwendbar ist. Sei T_i ein L_i'-passender Trie. Folgender Trie T ist dann L-passend:

$$T = \begin{cases} \text{  } & \text{falls } p \text{ nicht existiert} \\ \text{  } & \text{sonst} \end{cases}$$

Falls $L_1 = 1$, dann wenden wir auf L_2' die Induktionsannahme an. Sei T_2 der konstruierte L_2'-passende Trie. Dann ist folgender Trie T L-passend:



■

Offen ist noch der Beweis des Lemmas 1.7.

Beweis (Lemma 1.7):

Wir konstruieren sukzessive L_1. Hierzu arbeiten wir L von links nach rechts ab.

Bezeichne l_q stets den letzten Wert aus L, der zu L_1 hinzugenommen worden ist. D.h., in L_1 sind bisher $l_1, l_2, \ldots, l_q$.

Während der Konstruktion wird stets folgende Invariante erfüllt sein:

- Falls $\sum_{i=1}^{q} 2^{-l_i} \neq 2^{-1}$ und $q < n$,

  dann gilt

  $$2^{-1} - \sum_{i=1}^{q} 2^{-l_i} = s \cdot 2^{-l_{q+1}}$$

  für ein $s \geq 1$.

Wegen $\ell_1 \geq 1$ gilt $2^{\ell_1 - 1} \geq 1$. Somit folgt aus

$$2^{-1} = 2^{(\ell_1 - 1)} 2^{-\ell_1},$$

dass die Invariante vor Hinzunahme von ℓ_1 erfüllt ist.

Annahme:
Die Invariante ist vor Hinzunahme von ℓ_{q+1}, $q \geq 0$ erfüllt.

- Falls $\sum_{i=1}^{q} 2^{-\ell_i} = 2^{-1}$, dann gilt $p = q$ und wir sind fertig.

- Falls $\sum_{i=1}^{q} 2^{-\ell_i} \neq 2^{-1}$ und $q = n$, dann gilt $\sum_{i=1}^{q} 2^{-\ell_i} < 2^{-1}$ und wir sind auch fertig.

- Andernfalls folgt aus der Invariante

$$2^{-1} - \sum_{i=1}^{q} 2^{-\ell_i} = s \cdot 2^{-\ell_{q+1}} \text{ für ein } s \geq 1.$$

$$\Rightarrow 2^{-1} - \sum_{i=1}^{q+1} 2^{-\ell_i} = (s-1)\, 2^{-\ell_{q+1}} \qquad (1.2)$$

  - Falls $s = 1$, dann gilt $p = q+1$ und wir sind fertig.

- Falls $s > 1$ und $q+1 = n$, dann gilt
$\sum_{i=1}^{n} 2^{-\ell_i} < 2^{-1}$ und wir sind auch fertig.

- Falls $s > 1$ und $q+1 < n$, dann impliziert
$\ell_{q+2} = \ell_{q+1} + r$ für ein $r \geq 0$, dass
$2^{-\ell_{q+1}} = 2^{r} 2^{-\ell_{q+2}}$.

Also folgt aus Gleichung 1.2.1

$$2^{-1} - \sum_{i=1}^{q+1} 2^{-\ell_i} = \underbrace{(s-1)\,2^{r}}_{\geq 1} 2^{-\ell_{q+2}}.$$

Somit ist die Invariante auch nach Hinzunahme von ℓ_{q+1} erfüllt.

Irgendwann ist p gefunden oder das Ende der Folge L erreicht.

■

Übung:
Verallgemeinern Sie die Kraft-Ungleichung für Kodealphabete beliebiger Größe $r \geq 2$ und beweisen Sie diese.

Ein Kode φ heißt eindeutig dekodierbar, falls jede mittels φ kodierte Nachricht exakt eine Interpretation besitzt. Demzufolge sind Präfixkodes eindeutig dekodierbar.

Beispiel: (nicht eindeutig dekodierbarer Kode)

$\Sigma = \{a, b, c, d\}$

$\psi(a) = 0$, $\psi(b) = 01$, $\psi(c) = 11$, $\psi(d) = 00$

Die Nachricht 0011 besitzt zwei mögliche Interpretationen. Nämlich dc und aac.

$\Rightarrow$

ψ ist nicht eindeutig dekodierbar.

Folgender Satz besagt, dass jeder eindeutig dekodierbare Kode die Kraft-Ungleichung erfüllt.

Satz 1.4 (McMillan 1953)

Sei ψ ein eindeutig dekodierbarer Binärkode für ein Alphabet Σ der Größe n. Dann erfüllt ψ die Kraft-Ungleichung. D.h.,

$$\sum_{i=1}^{n} 2^{-\ell_i} \leq 1,$$

wobei $L = \ell_1, \ell_2, \ldots, \ell_n$ die Folge der Kodewortlängen ist.

Beweis (Karush 1961)

Sei $c := \sum_{i=1}^{n} 2^{-\ell_i}$.

Idee:

Wir betrachten c^N für beliebiges N und schätzen c^N nach oben ab. Wenn die Sich

ergebende obere Schranke polynomiell in N ist, dann folgt hieraus $c \leq 1$. Wäre $c > 1$, dann würde c^N exponentiell wachsen, so dass für N groß genug c^N größes als die obere Schranke wäre.

Es gilt:

$$c^N = \left(\sum_{i=1}^{n} 2^{-\ell_i} \right)^N$$

$$(1.22) \quad = \sum_{i_1=1}^{n} \sum_{i_2=1}^{n} \cdots \sum_{i_N=1}^{n} 2^{-(\ell_{i_1} + \ell_{i_2} + \ldots + \ell_{i_N})}$$

Beobachtung:

Die Größe des Exponenten $(\ell_{i_1} + \ell_{i_2} + \ldots + \ell_{i_N})$ ist gleich der Länge der Kodierung des Strings

$$x = a_{i_1} a_{i_2} \cdots a_{i_N}.$$

$\Rightarrow$

Jeder Summand in der Summe 1.22 korrespondiert zu paarweise verschiedenen Strings aus Σ^N und umgekehrt.

Bezeichne

- A_k die Anzahl der Strings aus Σ^N, deren Kodierungen die Länge k haben.
- $\ell_{min} = \min \{ \ell_i \mid 1 \leq i \leq n \}$

- $\ell_{max} = \max\{\ell_i \mid 1 \le i \le n\}$.

Da γ eindeutig dekodierbar ist, haben unterschiedliche Strings in Σ^N auch verschiedene Kodierungen, woraus $A_k \le 2^k$ für alle k folgt. Also gilt für alle N

$$c^N \le \sum_{k=N\ell_{min}}^{N\ell_{max}} 2^{-k} A_k$$

$$\le \sum_{k=N\ell_{min}}^{N\ell_{max}} 2^{-k} 2^k$$

$$\le N \cdot \ell_{max}.$$

Da ℓ_{max} eine Konstante ist, wächst c^N höchstens linear in N, woraus $c \le 1$ folgt. ∎

Wir interessieren uns nun für optimale Kodes. Ein eindeutig dekodierbarer Kode heißt <u>vollständig</u>, falls die beliebige Hinzunahme eines neuen Kodewortes zu einem nicht eindeutig dekodierbaren Kode führen würde.

<u>Übung:</u>
Zeigen Sie, dass ein Kode genau dann vollständig ist, wenn er seine korrespondierende Kraft-Ungleichung mit Gleichheit erfüllt.

Sei $\varphi : \Sigma \to \{0,1\}^+$ ein Präfixkode. Für $a \in \Sigma$ bezeichne $p(a)$ die Wahrscheinlichkeit, dass a in einem Text auftritt. Sei $l_a := |\varphi(a)|$ die Länge des Kodewortes von a. Wünschenswert wäre es, die Anzahl der zur Kodierung eines Textes benötigten Bits zu minimieren. D.h., wir möchten die <u>mittlere Kodewortlänge</u>

$$L_{\varphi,p} := \sum_{a \in \Sigma} p(a)\, l_a$$

minimieren. Die <u>minimale mittlere Kodewortlänge</u> ist dann definiert durch

$$L_p := \min \{ L_{\varphi,p} \mid \varphi \text{ ist Präfixkode für } \Sigma \}$$

Ein Präfixkode φ mit $L_{\varphi,p} = L_p$ heißt <u>optimal bezüglich der Wahrscheinlichkeitsverteilung p für Σ</u>.

<u>Bemerkung</u>

- Die Sätze 1.3 und 1.4 implizieren, dass jeder eindeutig dekodierbare Kode durch einen Präfixkode ersetzt werden kann, ohne dadurch die Länge der Kodeworte zu ändern. Demzufolge können wir uns bei der Suche nach eindeutig dekodierbaren Kodes mit minimaler mittlerer Kodewortlänge auf die Teilklasse der Präfixkodes beschränken.

- Der sogenannte Huffman-Kode ist ein optimales Präfixkode und kann leicht mittels eines Greedyalgorithmus konstruiert werden.

Der Huffman-Kode gibt eine positive Antwort auf folgende Frage:

- Gegeben eine Quelle, die Quellworte aus dem Universum gemäß einer Wahrscheinlichkeitsverteilung p generiert. Ist es möglich, den Quellworten derart Kodeworte zuzuweisen, dass jede beliebige Kodewortfolge eindeutig dekodierbar und die mittlere Kodewortlänge minimal sind?

Folgender Satz charakterisiert in Abhängigkeit von der Wahrscheinlichkeitsverteilung die minimale mittlere Kodewortlänge.

Satz 1.6 (Störungsfreie Kodierungstheorem)

Sei Σ ein Quellalphabet und p eine Wahrscheinlichkeitsverteilung auf Σ. Seien $H(p) = -\sum_{a \in \Sigma} p(a) \log p(a)$ die Entropie und L_p die minimale mittlere Kodewortlänge bezüglich Σ und p. Dann gilt

$$H(p) \leq L_p \leq H(p) + 1.$$

Beweis:

$L_p \leq H(p) + 1$:

Sei $l_a := \lceil -\log p(a) \rceil$, $a \in \Sigma$. Dann gilt

$$\sum_{a \in \Sigma} 2^{-l_a} \leq \sum_{a \in \Sigma} p(a) = 1.$$

Die Kraft-Ungleichung impliziert, dass ein Präfixkode mit Kodewortlängen l_a, $a \in \Sigma$ existiert. Also gilt

$$\begin{aligned} L_p &\leq \sum_{a \in \Sigma} p(a) \cdot l_a \\ &\leq \sum_{a \in \Sigma} p(a) \left(-\log p(a) + 1 \right) \\ &= H(p) + 1. \end{aligned}$$

<u>$H(p) \leq L_p$</u>:

Sei $L_p = \sum_{a \in \Sigma} p(a) l_a$ (Beachte, l_a wird hier neu definiert).

Es gilt:

$$\sum_{a \in \Sigma} \frac{2^{-l_a}}{\sum_{b \in \Sigma} 2^{-l_b}} = 1 \quad \text{und} \quad \sum_{a \in \Sigma} p(a) = 1$$

Wegen

$$\sum_{a \in \Sigma} p(a) \log \frac{1}{p(a)} \leq \sum_{a \in \Sigma} p(a) \log \frac{1}{q(a)}$$

für alle Wahrscheinlichkeitsverteilungen q für Σ gilt:

$$-\sum_{a\in\Sigma} p(a)\log p(a) \le -\sum_{a\in\Sigma} p(a)\log \frac{2^{-\ell_a}}{\sum_{b\in\Sigma} 2^{-\ell_b}}$$

$$= \sum_{a\in\Sigma} p(a)\,\ell_a + \left(\sum_{a\in\Sigma} p(a)\right)\log \sum_{b\in\Sigma} 2^{-\ell_b}.$$

Wegen $\sum_{a\in\Sigma} p(a) = 1$, $L_p = \sum_{a\in\Sigma} p(a)\,\ell_a$ und

$$H(p) = -\sum_{a\in\Sigma} p(a)\log p(a)$$

kann obige Ungleichung wie folgt geschrieben werden:

$$(1.23)\quad H(p) \le L_p + \log \sum_{b\in\Sigma} 2^{-\ell_b}.$$

Da γ Präfixcode folgt aus der Kraft-Ungleichung

$$\sum_{b\in\Sigma} 2^{-\ell_b} \le 1.$$

$$\Rightarrow \log \sum_{b\in\Sigma} 2^{-\ell_b} \le 0$$

woraus wegen 1.23

$$H(p) \le L_p$$

folgt. ∎

Bemerkung über

- weitere Resultate der klassischen Informations- und Codierungstheorie wie z.B fehlerkorrigierende Codes. Keine Relevanz für mod. Inf.sys ⇝ Literatur.

2. Die algorithmische Informationstheorie

Literatur:

Ming Li, Paul Vitányi, An Introduction to Kolmogorov Complexity and Its Applications, 2nd edn., Springer, 1997.

Alexander Shen, Lecture Notes zur Vorlesung "Kolmogorov Complexity and its Applications", gehalten im Herbst 2000 an der Universität Uppsala (elektronisch erhältlich von Homepage der Vorlesung)

Wie kann man die in einem String x enthaltene Information nur in Abhängigkeit vom String x messen?

Idee:

① Die Größe der enthaltenen Information hat etwas mit der Komprimierbarkeit des Strings x zu tun.

Beispiel:

$x = x_1 x_2 \dots x_n$ mit $x = 0101\dots01$ kann durch "nimm $n/2$ Wiederholungen von 01" beschrieben werden, was für große n wesentlich kompakter ist, als den String x explizit hinzuschreiben. ◆

Falls der String x keine Regelmäßigkeiten

enthält, dann gibt es in der Regel keine kompaktere Möglichkeit, als den String explizit hinzuschreiben. Demzufolge liefert die Größe der in einem String x enthaltenen Information etwas mit der Beschreibungskomplexität des Strings x zu tun.

Ziel:
Formalisierung der Beschreibungskomplexität von Strings.

Übung:
Wiederholen Sie in Ihrer Grundstudiums-vorlesung die Kapitel über rekursive Funktionen, Turing Maschinen, universelle Turingmaschine, Gödelisierung, Halteproblem etc.

2.1 Die Kolmogorov-Komplexität

Seien
- Σ ein endliches Alphabet
- $|x|$, $x \in \Sigma^+$ die Länge des Strings x
- σ ein Algorithmus, der Binärstrings endlicher Länge auf Elemente endlicher Länge von Σ^+ abbildet (z.B eine TM)

Die algorithmische Komplexität $K_\sigma(x)$ eines Strings $x \in \Sigma^+$ bezüglich Algorithmus σ ist die Länge einer kürzesten Eingabe $p \in \{0,1\}^+$, so

dass σ bei Eingabe p den String x ausgibt. D.h.,

$$K_\sigma(x) := \min \{ |p| \mid \sigma(p) = x \}.$$

Falls keine solche Eingabe existiert, d.h., $\sigma(p) \neq x$ $\forall p \in \{0,1\}^+$, dann definieren wir

$$K_\sigma(x) := \infty.$$

Interpretation:

- p als komprimierte Version von x $\rightsquigarrow$ σ Dekompressionsalgorithmus.

Bemerkung:
Obiges Komplexitätsmaß hängt wesentlich vom gewählten Dekompressionsalgorithmus ab.

Beispiel:
$\Sigma = \{0,1\}$, Dekompressionsalgorithmus σ_0 mit $\sigma_0(p) = p$ $\forall p \in \{0,1\}^+$.

$\Rightarrow$

$$K_{\sigma_0}(x) = |x| \quad \forall x \in \Sigma^+.$$

Man kann versuchen, Dekompressionsalgorithmen zu konstruieren, die kürzere Beschreibungen von x liefern. Aufgrund der begrenzten Anzahl von "kurzen" Beschreibungen haben alle Dekompressionsalgorithmen die Eigenschaft, dass "fast alle" $x \in \Sigma^+$ notwendigerweise die algorith-

mische Komplexität ungefähr 1×1 haben.

Ziel:
Vergleich zweier Dekompressionsalgorithmen O_1 und O_2.

Ein Algorithmus O_1 ist asymptotisch nicht schlechter als ein Algorithmus O_2, falls eine Konstante c_{O_2} existiert, so dass für alle $x \in \Sigma^+$

$$K_{O_1}(x) \leq K_{O_2}(x) + c_{O_2}.$$

Bemerkung:

- Eine universelle TM kann, gegeben ein Simulationsprogramm für eine beliebige TM, diese simulieren.
- allgemein: universelle Algorithmen.

Ein Algorithmus, der asymptotisch nicht schlechter ist als jeder beliebige andere Algorithmus, heißt asymptotisch optimal.

Satz 2.1 (Invarianztheorem)

Seien U ein universeller Algorithmus und $x \in \Sigma^+$. Dann gilt für jeden anderen Algorithmus O

$$K_U(x) \leq K_O(x) + c_O,$$

wobei c_{σ} eine Konstante ist, die von U und σ, nicht jedoch von x, abhängt.

Beweis:

Sei $p_{\sigma x}$ eine Eingabe für den Algorithmus σ mit $\sigma(p_{\sigma x}) = x$. Wir fügen der Eingabe $p_{\sigma x}$ ein Simulationsprogramm s_{σ} hinzu. Dieses beschreibt dem universellen Algorithmus U, wie er den Algorithmus σ zu simulieren hat. Das Simulationsprogramm hängt nur vom Algorithmus σ und nicht von der Eingabe $p_{\sigma x}$ ab. Da σ eine feste endliche Größe besitzt, ist die Länge c_{σ} von s_{σ} konstant.

Die Eingabe p für U mit $U(p) = x$ ist dann

$$p = s_{\sigma} p_{\sigma x}.$$

Es gilt dann

$$|p| = |s_{\sigma}| + |p_{\sigma x}| = c_{\sigma} + |p_{\sigma x}|.$$

Also gilt $\forall x \in \Sigma^*$

$$\begin{aligned} K_U(x) &= \min\{|q| \mid U(q) = x\} \\ &\leq \min\{|q| \mid \sigma(q) = x\} + c_{\sigma} \\ &= K_{\sigma}(x) + c_{\sigma}. \end{aligned}$$

∎

Bemerkung:

- Das Invarianztheorem besagt, dass ein universeller Algorithmus asymptotisch nicht schlechter ist, als jeder beliebige andere Algorithmus. Also ist ein universeller Algorithmus asymptotisch optimal.
- Falls U_1 und U_2 universelle Algorithmen sind, dann gilt $\forall x \in \Sigma^+$

$$|K_{U_1}(x) - K_{U_2}(x)| \leq c,$$

wobei $c = \max\{c_{U_1}, c_{U_2}\}$.

Die algorithmische Komplexität $K_U(x)$ bezüglich eines asymptotisch optimalen Algorithmus U heißt Kolmogorov-Komplexität $K(x)$ von x.

Bemerkung:

- Aus obigen Betrachtungen folgt, dass der Wechsel von einem asymptotisch optimalen Algorithmus U zu einem anderen asymptotisch optimalen Algorithmus U' die algorithmische Komplexität höchstens um einen additiven konstanten Term verändert. Daher vernachlässigen wir den konkreten universellen Algorithmus bei der Definition der Kolmogorov-Komplexität $K(x)$ eines Strings x.
- Falls es einen speziellen Algorithmus O_1 gibt, der die algorithmische Komplexität eines Strings x sehr klein macht, dann bildet $K_{O_1}(x)$ plus

die Länge des Simulationsprogramms für σ stets eine obere Schranke für $K(x)$.

Folgendes Lemma fasst einige weitere grundlegende Eigenschaften der Kolmogorov-Komplexität zusammen:

Lemma 2.1

Sei $\Sigma = \{0,1\}$. Die Kolmogorov-Komplexität bezüglich Strings in Σ^+ besitzt folgende Eigenschaften.

a) Für alle $x \in \Sigma^+$ gilt

$$K(x) \leq |x| + c_{\sigma_0},$$

wobei c_{σ_0} eine Konstante ist.

b) Es existiert eine Konstante c, so dass

$$2^{n-c} \leq |\{x \in \Sigma^+ \mid K(x) \leq n\}| \leq 2^{n+1}$$

für alle $n \in \mathbb{N}$.

c) Für jede berechenbare Funktion f existiert eine Konstante c_f, so dass für alle x, für die $f(x)$ definiert ist

$$K(f(x)) \leq K(x) + c_f.$$

d) Für $n \in \mathbb{N}$ sei V_n eine endliche Menge mit maximal 2^n Elementen. Ferner sei die Relation $x \in V_n$ rekursiv aufzählbar.

D.h., es existiert ein Algorithmus, der die Liste aller Paare $\langle x, n\rangle$ mit $x \in V_n$ produziert. Dann existiert eine Konstante c, so dass $\forall\, n \in \mathbb{N}$ alle Elemente in V_n Kolmogorov-Komplexität $\leq n + c$ besitzen.

e) Der "typische" Binärstring der Länge n hat Kolmogorov-Komplexität nahe bei n. D.h., es existiert eine Konstante c, so dass für beliebiges n mehr als 99 % aller Strings der Länge n Kolmogorov-Komplexität zwischen $n-c$ und $n+c$ besitzen.

Beweis:

a) Ein asymptotisch optimaler Algorithmus kann asymptotisch nicht schlechter sein, als der triviale Dekompressionsalgorithmus σ_0 mit $\sigma_0(p) = p \;\forall\, p \in \Sigma^+$. Hieraus folgt direkt

$$K(x) \leq |x| + c_{\sigma_0} \quad \forall\, x \in \Sigma^+,$$

wobei c_{σ_0} die Länge eines Simulationsprogramms für σ_0 ist.

b) Sei

$$A_n := \{ x \in \Sigma^+ \mid K(x) \leq n \}$$

Offensichtlich kann A_n nicht mehr Strings enthalten, als Anzahl der Binärstrings der Länge $\leq n$. Also gilt

$$|A_n| \leq 2^{n+1}$$

Ferner impliziert a), dass alle Strings der Länge $n - c_{\text{Ort}_0}$ Kolmogorov-Komplexität $\leq n$ haben. Hiervon gibt es $2^{n-c_{\text{Ort}_0}}$ viele.

$\Rightarrow$

$$|A_n| \geq 2^{n-c} \quad \text{für} \quad c = c_{\text{Ort}_0}.$$

c) Sei U derjenige universelle Dekompressionsalgorithmus, der bei der Definition der Kolmogorov-Komplexitätsfunktion K verwendet wird.

f berechenbar $\Rightarrow$

Es existiert ein Programm s_f konstanter Größe c_f für U, so dass U bei Eingabe $s_f x$ den Funktionswert $f(x)$ ausgibt.

Wenn U eine Eingabe $s_f p$ mit $U(p) = x$ erhält, dann kann U zunächst x mit Hilfe von p und dann unter Verwendung von s_f den Funktionswert $f(x)$ berechnen.

$\Rightarrow$

$$K_U(f(x)) \leq K_U(x) + c_f.$$

d) <u>Idee</u>:

Konstruktion eines speziellen Dekompressions-

algorithmus W, so dass $K_W(x) \leq n$ für alle $n \in \mathbb{N}$, $x \in V_n$. Das Invarianztheorem impliziert dann die Behauptung.

Jeder String $x \in V_n$ erhält wie folgt als komprimierte Version einen String der Länge n:

Annahme:
Das Paar $\langle x, n \rangle$ erscheint in der vom Aufzählungsalgorithmus konstruierten Liste unter allen Paaren mit zweiter Komponente n als i-tes Element.

Wir interpretieren Binärstrings $a_{n-1}a_{n-2}\dots a_0$ der Länge n als Zahl

$$\sum_{j=0}^{n-1} a_j \cdot 2^j.$$

x erhält dann als komprimierte Version den String $a_{n-1}a_{n-2}\dots a_0$ mit

$$\sum_{j=0}^{n-1} a_j 2^j = i.$$

Der Dekompressionsalgorithmus W enthält den Aufzählungsalgorithmus als Subroutine.

Annahme:
W erhält als Eingabe $a_{n-1}a_{n-2}\dots a_0$.

Dann ermittelt W aus der Eingabe n und

$$i := \sum_{j=0}^{n-1} a_j 2^j.$$

W zählt alle Paare $\langle y, m \rangle$ auf, bis er das i-te Paar der Form $\langle x, n \rangle$ aufgezählt hat und gibt dann x aus.

Offensichtlich gilt

$$K_W(x) \leq n \quad \forall n \in \mathbb{N} \; \forall x \in V_n.$$

e)

a) $\Rightarrow$

Alle Strings der Länge n haben Kolmogorov-Komplexität $\leq n + c_{\sigma_0}$.

$\Rightarrow$

Für $c \geq c_{\sigma_0}$ gilt: Alle Strings der Länge n haben Kolmogorov-Komplexität $\leq n + c$.

Die Anzahl der Strings mit Kolmogorov-Komplexität $< n - c$ kann nicht größer sein, als die Gesamtanzahl der Strings der Länge $< n - c$, also maximal 2^{n-c}.

Wähle

$$c := \max \{ c_{\sigma_0}, 7 \}.$$

Dann gilt:

1) Alle Strings der Länge n haben Kolmogorov-Komplexität $\leq n + c$.

2) Der Anteil der Strings der Länge n mit Kolmogorov-Komplexität $< n-c$ ist höchstens

$$\frac{2^{n-c}}{2^n} = 2^{-c} \leq 2^{-7} = \frac{1}{128} < 1\%.$$

■

Satz 2.2

Die Kolmogorov-Komplexitätsfunktion K ist nicht berechenbar. Mehr noch, jede berechenbare untere Schranke für K ist von oben beschränkt.

Beweis:

Sei k eine berechenbare untere Schranke der Funktion K. D.h., $k(x) \leq K(x)$ $\forall x \in \Sigma^+$.

Annahme: k ist nicht von oben beschränkt.

$\Rightarrow$

Für jedes $m > 0$ existiert ein x mit

$$K(x) \geq k(x) > m.$$

Gegeben m kann solches x wie folgt berechnet werden:

- Zähle alle x auf und berechne $k(x)$ bis ein x mit $k(x) > m$ gefunden ist.

Betrachten wir die Funktion $f: \mathbb{N} \to \Sigma^+$, wobei $f(m)$ in obiger Aufzählung der erste String $x \in \Sigma^+$ mit $k(x) > m$ ist.

Wir schreiben die natürliche Zahl m in Binärnotation. D.h., $|m| = \lceil \log m \rceil$

Da $k(x) \leq K(x)$ gilt

(2.1) $\qquad K(f(m)) > m$

Da f eine berechenbare Funktion ist, impliziert Lemma 2.1 c)

(2.2) $K(f(m)) \leq K(m) + c_f$,

für eine Konstante c_f.

Lemma 2.1 a) $\Rightarrow$

(2.3) $K(m) \leq |m| + c_{U_0}$,

für eine Konstante c_{U_0}.

Also folgt aus (2.1), (2.2) und (2.3)

$$m < K(f(m)) \leq |m| + c_{U_0} + c_f$$
$$\leq \log m + c',$$

wobei $c' := c_{U_0} + c_f + 1$ eine Konstante ist.

Dies kann für m groß genug nicht erfüllt sein. Demzufolge ist unsere Annahme falsch und somit k von oben beschränkt. ∎

Übung:

a) Sei D ein Dekompressionsalgorithmus, so dass $K_D(x)$ gerade $\forall x \in \Sigma^+$. Kann D asymptotisch optimal sein?

b) Sei D ein Dekompressionsalgorithmus, so dass $K_D(x)$ eine Potenz von 2 ist $\forall x \in \Sigma^+$. Kann D asymptotisch optimal sein?

c) Sei D ein asymptotisch optimaler Dekompressionsalgorithmus. Garantiert dies, dass $D(D(x))$ auch ein asymptotisch optimaler Dekompressionsalgorithmus ist?

Eine Funktion $F: \Sigma^+ \to \mathbb{N}$ heißt <u>von oben rekursiv aufzählbar</u>, falls eine totale berechenbare Funktion $k: \Sigma^+ \times \mathbb{N}_0 \to \mathbb{N}$ existiert mit

i) $k(x,0) \geq k(x,1) \geq k(x,2) \geq \cdots \quad \forall x \in \Sigma^+$

ii) $F(x) = \lim_{n \to \infty} k(x,n)$.

<u>Bemerkung:</u>
Da für alle $x \in \Sigma^+$ und alle $n \in \mathbb{N}_0$ die Werte $k(x,n)$ ganzzahlig sind, existiert für jedes $x \in \Sigma^+$ ein n_x, so dass $k(x,n) = F(x)$ $\forall n \geq n_x$.

<u>Satz 2.3</u>

Die Kolmogorov-Komplexitätsfunktion ist von oben rekursiv aufzählbar.

Beweis:

Sei U derjenige universelle Dekompressionsalgorithmus, der bei der Definition von K verwendet wird.

Idee:

Definiere $k(x,n)$ als die Länge des kürzesten $y \in \Sigma^+$, so dass U mit Eingabe y den String x ausgibt und dabei maximal n Schritte durchführt.

Schwierigkeit:

1. Für zu kleines n kann es sein, dass kein y existiert, für das U den String x ausgibt und hierfür maximal n Schritte verwendet.
2. Es kann sein, dass das kürzeste y mit $U(y) = x$ innerhalb $\leq n$ Schritten sehr lang ist.

Daher modifizieren wir obige Idee und nehmen als Wert von $k(x,n)$ das Minimum aus der Länge eines solchen kürzesten y und einer trivialen oberen Schranke für $K(x)$.

$\rightsquigarrow$

Sei $|x| + c_{U,0}$ die gemäß Lemma 2.1 a) triviale obere Schranke für $K(x)$.

Wir definieren dann $\forall\, x \in \Sigma^+ \; \forall\, n \in \mathbb{N}_0$

$$k(x,n) := \min\left\{ |x| + c_{U,0},\ \min_{y \in \Sigma^+}\left\{ |y| \;\middle|\; \begin{array}{l} U(y) = x \\ \text{in} \leq n \text{ Schritten} \end{array} \right\} \right\}.$$

Übung:
Beweisen Sie, dass die oben definierte Funktion k total und berechenbar ist.

Aus der Definition der Funktion k folgt direkt

$$K(x) = \lim_{n \to \infty} k(x,n).$$

■

Übung:

Kann es einen asymptotisch optimalen Dekompressionsalgorithmus geben, der für jede Eingabe mit einer Ausgabe anhält? Beweisen Sie Ihre Antwort.

Folgender Satz liefert uns eine obere Schranke für die Kolmogorov-Komplexität.

Satz 2.4

Sei $K' : \{0,1\}^+ \longrightarrow \mathbb{N}$ eine von oben rekursiv aufzählbare Funktion, so dass für eine Konstante C für alle $n \in \mathbb{N}$ gilt.

$$|\{x \mid K'(x) < n\}| \leq C 2^n.$$

Dann existiert eine Konstante c mit

$$K(x) \leq K'(x) + c \quad \forall x \in \{0,1\}^+$$

Beweis:

K' von oben rekursiv aufzählbar $\Rightarrow$

$\exists$ totale, berechenbare Funktion $k' : \{0,1\}^+ \times W_0 \to \mathbb{N}$ so dass $\forall x \in \{0,1\}^+$

i) $k'(x,0) \geq k'(x,1) \geq k'(x,2) \geq \ldots$ und

ii) $K'(x) = \lim_{m \to \infty} k'(x,m)$

Sei

$$W_n := \{ x \in \{0,1\}^+ \mid K'(x) < n \}.$$

Folgender Algorithmus zählt für ein gegebenes n die binäre Relation $x \in W_n$ auf:

(1) Berechne " für alle x und m parallel "
$k'(x,m)$.

(2) Falls $k'(x,m) < n$, dann füge x zur Aufzählung von W_n hinzu.

Übung:
Arbeiten Sie obigen Algorithmus aus. Insbesondere beschreiben Sie die Implementierung von " für alle x und m berechne parallel " $k'(x,m)$.

k' in der zweiten Komponente monoton fallend

$\Rightarrow$

$$\left(k'(x,m) < n \Rightarrow K'(x) < n \right).$$

Wegen $K'(x) = \lim_{m\to\infty} k'(x,m)$ existiert ein $m_x \in \mathbb{N}$ so dass

$$K'(x) = k'(x,m) \quad \forall\ m \geq m_x .$$

Also wird jedes Element $x \in W_n$ irgendwann bezüglich W_n aufgezählt.

Setze

$$n' := n + \lceil \log C \rceil \quad \text{und}$$
$$V_{n'} := W_n .$$

Dann gilt:

$$|V_{n'}| \leq C 2^n \leq 2^{n'}.$$

Bezeichne $A(n)$ die Minimalanzahl von Bits, so dass jedes Element in W_n mit $A(n)$ Bits kodiert werden kann

Lemma 2.1 a) $\Rightarrow$

$$A(n) \leq n + \lceil \log C \rceil + c', \qquad (2.4)$$

wobei c' eine Konstante ist, die nicht von n abhängt. Setze

$$\tilde{c} := \lceil \log C \rceil + c'. \qquad (2.5)$$

Betrachten wir $x \in \{0,1\}^+$ beliebig. Zu zeigen ist

$K(x) \leq K'(x) + c$ für eine Konstante c.

Betrachten wir hierzu $n := K'(x) + 1$.

$\Rightarrow$

$$x \in W_n.$$

(2.4) und (2.5) $\Rightarrow$

$$\begin{aligned} K(x) &\leq n + \tilde{c} \\ &= K'(x) + 1 + \tilde{c} \\ &= K'(x) + c, \end{aligned}$$

wobei $c = 1 + \tilde{c}$. ∎

Ziel:
Formulierung und Beweis eines "Eindeutigkeits-Satzes" für die Kolmogorov-Komplexitätsfunktion.

Satz 2.5
Sei $K' : \{0,1\}^+ \to \mathbb{N}$ eine Funktion, die folgende Axiome erfüllt:

(A1) Für jede berechenbare (partielle) Funktion f existiert eine Konstante c_f, so dass

$$K'(f(x)) \leq K'(x) + c_f$$

für alle $x \in \{0,1\}^+$, für die $f(x)$ definiert ist.

(A2) Die Funktion K' ist von oben rekursiv aufzählbar.

(A3) Es existieren Konstanten c und C, so dass

$$c2^n \leq |\{x \in \{0,1\}^+ \mid K'(x) < n\}| \leq C2^n$$

für alle $n \in \mathbb{N}$.

Dann unterscheidet sich K' von der Kolmogorov-Komplexitätsfunktion K höchstens um einen konstanten additiven Term.

Bemerkung:
Dass die Kolmogorov-Komplexitätsfunktion K die Axiome erfüllt, haben wir bereits bewiesen.

Beweis:

Axiome (A2), (A3) und Satz 2.4 $\Rightarrow$

$\exists$ Konstante $\check{c}$, so dass

$$K(x) \leq K'(x) + \check{c} \quad \forall x \in \{0,1\}^+.$$

2.2. $\exists$ Konstante $\bar{c}$, so dass

$$K'(x) \leq K(x) + \bar{c} \quad \forall x \in \{0,1\}^+ \qquad (2.6)$$

Zunächst werden wir beweisen, dass eine Konstante $\hat{c}$ existiert, so dass

$$K'(x) \leq |x| + \hat{c} \quad \forall x \in \{0,1\}^+ \qquad (2.7)$$

Danach werden wir (2.7) verwenden um (2.6) zu beweisen.

K' von oben rekursiv aufzählbar $\Rightarrow$

Wir können, wie im Beweis von Satz 2.4 beschrieben, Strings $x \in \{0,1\}^+$ mit $K'(x) < n$ generieren.

Axiom (A3) $\Rightarrow$

Es gibt mindestens $c2^n$ solche Strings. Betrachten wir $d \in \mathbb{N}$ minimal, so dass

$$2^{n-d} \leq c2^n \quad \forall n \in \mathbb{N}.$$

Für $n = 1, 2, 3, \ldots$ generieren wir sukzessive Strings $x \in \{0,1\}^+$ mit $K'(x) < n$ und beenden die Generierung bezüglich des aktuellen n, sobald 2^{n-d} solche Strings generiert sind.

Für $n \in \mathbb{N}$ bezeichne S_n die Menge der Strings, die bezüglich n generiert worden sind.

Bemerkung:
Für $n < d$ gilt $S_n = \emptyset$.

Da wir bei der Generierung von S_{n+1} stets genau S_n zu S_{n+1} hinzunehmen können und dies auch tun, gilt

$$S_d \subset S_{d+1} \subset S_{d+2} \subset \ldots$$

Seien

$$T_{d+1} := S_{d+1} \quad \text{und für } i > d+1$$

$$T_i := S_{i+1} \setminus S_i.$$

Dann gilt

$$|T_{d+1}| = |S_{d+1}| = 2^{(d+1)-d}$$

und für $i > d+1$

$$\begin{aligned} |T_i| &= |S_{i+1}| - |S_i| \\ &= 2^{(i+1)-d} - 2^{i-d} \\ &= 2^{i-d} \end{aligned}$$

Ferner sind alle T_i, $i \geq d+1$ paarweise disjunkt.

Wir betrachten eine beliebige berechenbare surjektive Funktion $f: \{0,1\}^+ \to \{0,1\}^+$, die für $n = d+1, d+2, \ldots$ die Strings in T_n auf Strings der Länge $n-d$ abbildet.

f surjektiv und $|T_n| = 2^{n-d}$ $\Rightarrow$

Jeder String der Länge $n-d$ ist Bild eines Elements $x \in T_n$.

Axiom (A1) $\Rightarrow$

$$K'(f(x)) \leq K'(x) + c_f \quad \forall x \in T_n.$$

Konstruktion von T_n $\Rightarrow$

$$K'(x) < n \quad \forall x \in T_n.$$

Also gilt für jedes $y \in \{0,1\}^+$ mit $|y| = n-d$

$$K'(y) \leq n + c_f = |y| + d + c_f.$$

Für $\hat{c} := d + c_f$ erhalten wir somit $\forall x \in \{0,1\}^+$

$$K'(x) \leq |x| + \hat{c}. \qquad (2.7)$$

Ziel:

Unter Verwendung von (2.7) Konstruktion einer Konstanten $\bar{c}$, so dass $\forall x \in \{0,1\}^+$

$$K'(x) \leq K(x) + \bar{c}.$$

Sei U der asymptotisch optimale Dekompressionsalgorithmus, der bei der Definition der Kolmogorov-Komplexitätsfunktion verwendet wird. Sei h diejenige partielle Funktion, die U berechnet.

Sei $x \in \{0,1\}^+$ beliebig, aber fest.

Betrachten wir $p \in \{0,1\}^+$ mit

$$U(p) = h(p) = x \quad \text{und} \quad |p| = K(x).$$

Anwendung von Axiom (A_1) ergibt

$$K'(x) = K'(h(p)) \leq K'(p) + c_h,$$

für eine Konstante c_h.

Sei $\bar{c} := \hat{c} + c_h$. Dann ergibt die Anwendung von (2.7) auf $K'(p)$

$$K'(x) \leq |p| + \hat{c} + c_h$$

$$= K(x) + \bar{c},$$

womit der Satz bewiesen ist. ∎

Übung:

a) Sei $f: \mathbb{N} \to \mathbb{N}$ eine berechenbare bijektive Funktion. Zeigen Sie, dass eine Konstante $\check{c}$ existiert, so dass $K(x) \leq K(f(x)) + \check{c}$. Dabei sind x bzw $f(x)$ die Binärdarstellungen der Zahlen x bzw $f(x)$ ohne führende Nullen.

b) Gilt obige Aussage auch, wenn wir bijektiv durch injektiv ersetzen?

c) Gilt obige Aussage auch, wenn wir bijektiv durch surjektiv ersetzen?

2.2 Bedingte Kolmogorov-Komplexität

Annahme:

Dekompressionsalgorithmus ψ erhält zusätzlich zur Eingabe p weitere Information über x.

Effekt:

Diese Information muss nicht "irgendwie" in p enthalten sein.

$\Rightarrow$

p kann eventuell kürzer sein, als es ohne diese Information möglich wäre

$\rightsquigarrow$

Interesse für durch zusätzliche Information bedingte Kolmogorov-Komplexität.

weiter auf S. 77

Da wir bedingte Kolmogorov-Komplexität mit Hilfe der Kolmogorov-Komplexität von Paaren von Strings definieren werden, definieren und analysieren wir zunächst diese.

Sei $g : \Sigma^+ \times \Sigma^+ \to \Sigma^+$ eine beliebige berechenbare injektive Funktion, die Paare von Strings auf Strings abbildet

g injektiv $\Rightarrow$

$$g(x,y) \neq g(x',y') \quad \text{falls } x \neq x' \text{ oder } y \neq y'.$$

Eine derartige Funktion heißt <u>Paarungs-funktion</u>.

Die <u>Kolmogorov-Komplexität</u> $K(x,y)$ eines Paares $(x,y) \in \Sigma^+ \times \Sigma^+$ ist definiert durch $K(g(x,y))$.

<u>Frage:</u>
Inwieweit ändert sich die Kolmogorov-Komplexität $K(x,y)$, wenn wir anstatt g eine andere Paarungsfunktion h nehmen würden?

<u>Lemma 2.2</u>

a) Seien g_1 und g_2 zwei beliebige Paarungsfunktionen. Dann existiert eine Konstante c, so dass $\forall (x,y) \in \Sigma^+ \times \Sigma^+$

$$K(g_1(x,y)) \leq K(g_2(x,y)) + c.$$

b) Es existieren Konstanten c_1 und c_2, so dass $\forall (x,y) \in \Sigma^+ \times \Sigma^+$

$$K(x,y) \geq K(x) - c_1 \quad \text{und}$$
$$K(x,y) \geq K(y) - c_2.$$

c) Es existieren Konstanten c_1 und c_2, so dass $\forall (x,y) \in \Sigma^+ \times \Sigma^+$

$$|K(x,y) - K(y,x)| \leq c_1 \quad \text{und}$$
$$|K(x,x) - K(x)| \leq c_2.$$

Beweis:

Zum Beweis von a) genügt es zu zeigen, dass ein Algorithmus existiert, der gegeben $g_2(x,y)$ den Funktionswert $g_1(x,y)$ berechnet. Dieser hat endliche Länge und kann der Eingabe p mit

$$U(p) = g_2(x,y) \quad \text{und} \quad |p| = K(g_2(x,y))$$

hinzugefügt werden.

Folgender Algorithmus leistet dies.

(1) Bis das Paar (x,y) gefunden ist, zähle die Paare $(x',y') \in \Sigma^+ \times \Sigma^+$ auf, berechne $g_2(x',y')$ und vergleiche $g_2(x',y')$ mit $g_2(x,y)$.

Da g_2 injektiv ist, gilt

$$(x',y') = (x,y) \iff g_2(x',y') = g_2(x,y).$$

(2) Berechne $g_1(x,y)$.

Die Behauptungen b) und c) können ähnlich bewiesen werden. ∎

Übung:
Beweisen Sie Lemma 2.2. b) und Lemma 2.2 c).

Verallgemeinerung der Kolmogorov-Komplexität für Paare auf Tripel, Quadrupel etc.:

- $K(x,y,z) := K(g(x,g(y,z)))$
- $K(x,y,z,t) := K(g(x,g(y,g(z,t))))$

etc.

Satz 2.6

Es existiert eine Konstante $\tilde{c}$, so dass für alle Paare $(x,y) \in \Sigma^+ \times \Sigma^+$

$$K(x,y) \leq K(x) + K(y) + 2\log K(x) + \tilde{c}.$$

Beweis:
Sei U der asymptotisch optimale Dekompressionsalgorithmus, der bei der Definition

der Kolmogorov-Komplexitätsfunktion ver-
wendet wird.

Seien p bzw. q Eingaben für U mit

$$U(p) = x \text{ und } |p| = K(x) \quad \text{bzw.}$$
$$U(q) = y \text{ und } |q| = K(y).$$

Idee:

Konstruktion eines Strings z mit

- $|z| = K(x) + K(y) + 2 \log K(x) + 2$
- z enthält p und auch q als Teilstring
- p und auch q können aus z eindeutig extrahiert werden.

Hat man p und q, dann können $U(p)$ und $U(q)$ berechnet werden. Unter Verwendung eines Algorithmus zur Berechnung der Paarungsfunktion g berechnen wir dann $g(x,y)$.

Durchführung:

Betrachte

$$z := \overline{bin(|p|)}\,01\,pq,$$

wobei wir $\overline{bin(|p|)}$ aus der Binärdarstellung $bin(|p|)$ der Länge von p mittels Verdopplung der einzelnen Bits erhalten

Bsp.:

$\overline{10110111} = 1100111100111111$

Es gilt dann

$$|z| = K(x) + K(y) + 2 \cdot \log K(x) + 2.$$

Wir betrachten z von links nach rechts und ermitteln die erste Eins, für die gilt:

- Der unmittelbar links von der Eins stehende Nullblock enthält eine ungerade Anzahl von Nullen.

Dies ist die Eins, die unmittelbar vor p steht

$\Rightarrow$

Wir kennen den Anfang von p in z.

Da wir auch die Länge von p kennen, können wir das Ende von p in z berechnen.

$\Rightarrow$

Wir kennen auch den Anfang von q in z.

Also können p und q eindeutig aus z extrahiert werden. ∎

Ziel:

Formale Definition der bedingten Kolmogorov-Komplexität

Seien $D : \Sigma^+ \times \Sigma^+ \longrightarrow \Sigma^+$ eine berechenbare Funktion und σ ein Algorithmus, der D berechnet. Die <u>bedingte algorithmische Komplexität</u> $K_\sigma(x|y)$ von x, wenn y bekannt ist, bezüglich σ ist definiert durch

$$K_\sigma(x|y) := \min\{|p| \mid \sigma(p,y) = x\},$$

falls p mit $D(p,y) = x$ existiert. Andernfalls vereinbaren wir $K_\sigma(x|y) = \infty$.

Die Funktion D heißt <u>bedingte Dekompressionsfunktion</u>. Jeder universelle Algorithmus kann auch einen Algorithmus, der die Funktion D berechnet, simulieren, wenn er ein entsprechendes Simulationsprogramm als Eingabe erhält.

<u>Satz 2.7</u> (Invarianztheorem)
Seien U ein universeller Algorithmus und $(x,y) \in \Sigma^+ \times \Sigma^+$. Dann gilt für jeden anderen Algorithmus σ

$$K_U(x|y) \leq K_\sigma(x|y) + c_\sigma,$$

wobei c_σ eine Konstante ist, die von U und σ, nicht jedoch von x und y abhängt.

<u>Beweis</u>:

Übung

∎

Die bedingte algorithmische Komplexität $K_U(x|y)$ bzgl. eines universellen Algorithmus U heißt bedingte Kolmogorov-Komplexität von x, wenn y bekannt ist.

Frage:
Wie sehr kann die Kenntnis von y helfen?

Satz 2.8

a) Es existiert eine Konstante c, so dass für alle $(x,y) \in \Sigma^+ \times \Sigma^+$ gilt

$$K(x|y) \leq K(x) + c.$$

b) Für jedes festes $y \in \Sigma^+$ existiert eine Konstante c_y, so dass $\forall x \in \Sigma^+$

$$|K(x) - K(x|y)| \leq c_y.$$

Beweis:

a) folgt direkt aus der Beobachtung, dass jeder nichtbedingte Dekompressionsalgorithmus als bedingter, der y ignoriert, angesehen werden kann.

b) Sei U ein universeller bedingter Dekompressionsalgorithmus. Für beliebiges festes y definieren wir den nichtbedingte Dekompressionsalgorithmus $U_y(p)$ durch

$$U_y(p) := U(p,y).$$

$\Rightarrow$

$$K_{U_y}(x) = K_U(x|y) \quad \forall x \in \Sigma^+$$

$\Rightarrow$

Satz 2.1

$$K(x) \leq K_{U_y}(x) + c',$$

wobei c' eine Konstante ist.

Sei c die Konstante aus Teil a) des Satzes
Für

$$c_y := \max\{c, c'\}$$

folgt dann die Behauptung. ∎

2.3 Nichtkomprimierbare Strings

Ein String x der Länge n heißt <u>nichtkomprimierbar</u>, falls $K(x|n) \geq n$. Sei c eine Konstante. x heißt <u>c-nichtkomprimierbar</u>, falls $K(x|n) \geq n - c$.

<u>Satz 2.9</u>
Für alle $n \in \mathbb{N}$ und alle Konstanten c ist unter allen Strings der Länge n der Anteil der c-nichtkomprimierbaren Strings größer als $1 - 2^{-c}$.

<u>Beweis</u>:
Für alle $i \in \mathbb{N}$ ist die Anzahl der Binärstrings der Länge i exakt 2^i. Also gilt für die Anzahl

$A^{<n-c}$ der Binärstrings der Länge $< n-c$:

$$A^{<n-c} = \sum_{i=1}^{n-c-1} 2^i < 2^{n-c}.$$

Da $A^{<n-c}$ sicher eine obere Schranke für die Anzahl der Strings x der Länge n mit $K(x|n) < n-c$ ist, ist der Anteil der c-nichtkomprimierbaren Strings

$$> 1 - \frac{2^{n-c}}{2^n} = 1 - 2^{-c}.$$

■

Eine unendliche Folge $x := x_1 x_2 x_3 \ldots$ von Nullen und Einsen heißt genau dann <u>berechenbar</u>, wenn es eine Turingmaschine M gibt, die x ausgibt. D.h., für jedes $n \in \mathbb{N}$ existiert ein $T(n) \in \mathbb{N}$, so dass die Ausgabe von M nach $T(n)$ Schritten gerade $x_1 x_2 \ldots x_n$ ist.

<u>Satz 2.10</u>

Eine unendliche Folge $x := x_1 x_2 x_3 \ldots$ von Nullen und Einsen ist genau dann berechenbar, wenn eine Konstante c_x mit

$$K(x_1 x_2 \ldots x_n | n) \leq c_x \quad \forall n \in \mathbb{N}$$

existiert.

Beweis:

„⇒"

Da x berechenbar ist, gibt es eine Turingmaschine M, die x ausgibt. Diese hat eine Beschreibung b_M endlicher Länge c_M.

Sei U die universelle Turingmaschine, die wir bei der Definition der Kolmogorov-Komplexität verwenden. U erhält als Eingabe

1. die Beschreibung b_M, die Länge n des Präfixes von x und
2. ein Simulationsprogramm, das M unter Verwendung von b_M simuliert, die Länge der bisherigen Ausgabe sich merkt und $x_1 x_2 \dots x_n$ ausgibt, sobald M das n-te Symbol x_n ausgegeben hat. Danach hält das Simulationsprogramm an.

Das Simulationsprogramm hat endliche Länge c_S. Also gilt mit $c_x := c_M + c_S$ $\forall n \in \mathbb{N}$

$$K(x_1 x_2 \dots x_n \mid n) \leq c_x.$$

„⇐"

Annahme:

$\exists$ Konstante c_x, so dass $\forall n \in \mathbb{N}$

$$K(x_1 x_2 \dots x_n \mid n) \leq c_x.$$

2.2. Es gibt eine Turingmaschine M, die x ausgibt.

Church'sche These $\Rightarrow$

Es genügt zu zeigen, dass es einen Algorithmus gibt, der x ausgibt.

Ziel:
Konstruktion eines Algorithmus σ_x, der x ausgibt.

Ein String z der Länge n heißt einfach, falls $K(z|n) \leq c_x$.

Da die Anzahl der Binärstrings der Länge $\leq c_x$ kleiner als 2^{c_x+1} ist, gilt $\forall n \in \mathbb{N}$:

- Die Anzahl der einfachen Strings der Länge n ist $< 2^{c_x+1}$.

Annahme $\Rightarrow$

Jeder Präfix des unendlichen Strings $x = x_1 x_2 x_3 \ldots$ ist einfach.

Ein String y heißt gut, wenn jeder Präfix von y einfach ist.

Bemerkung:
Falls y endlich ist, dann impliziert obige Definition, dass y selbst einfach ist. Jeder Präfix eines guten Strings ist selbst gut.

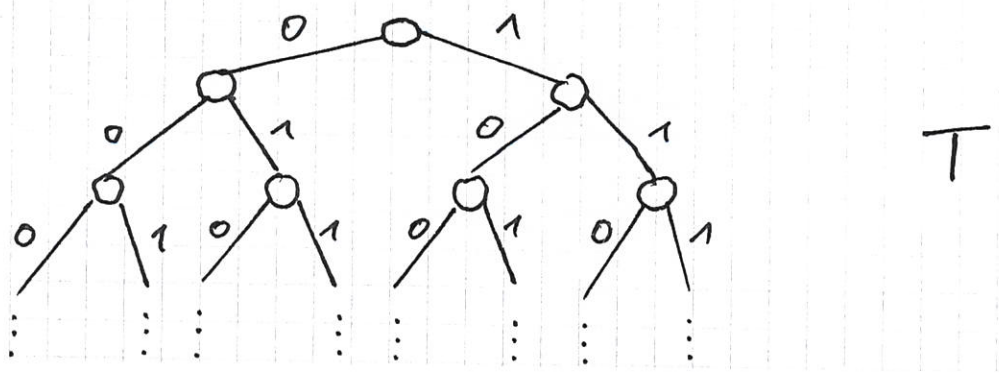
Beobachtung:

- Da für jedes $n \in \mathbb{N}$ die Anzahl der einfachen Strings $< 2^{c \cdot n+1}$ ist, ist auch für jedes $n \in \mathbb{N}$ die Anzahl der guten Strings $< 2^{c \cdot n+1}$. Da jeder Präfix eines unendlichen guten Strings selbst ein guter String ist, folgt hieraus, dass die Anzahl der unendlichen guten Strings auch $< 2^{c \cdot n+1}$ ist.

- Indem wir für alle $n \in \mathbb{N}$ alle kurze Beschreibungen parallel verschränkt testen, können wir die Menge der einfachen Strings rekursiv aufzählen. Den Aufzählungsalgorithmus für die einfachen Strings können wir für die Konstruktion eines Aufzählungsalgorithmus für die Menge der guten Strings verwenden.

Übung:
Konstruieren Sie Aufzählungsalgorithmen für die Menge der einfachen Strings und für die Menge der guten Strings.

- Wir können die Menge aller unendlichen binären Strings als unendlichen Trie T interpretieren.



Sei T' derjenige Teilbaum von T, der alle gute Strings enthält.

Übung:
Nehmen wir an, wir könnten stets für gegebenes m und gegebenen String y entscheiden, ob $K(y) \leq m$ oder nicht. Zeigen Sie, dass dann die Kolmogorov-Komplexitätsfunktion berechenbar ist.

Aus obiger Übung folgt, dass T' nicht bzgl. jeder Konstante c_x berechenbar ist. Jedoch kann ein Teilbaum T'' von T', der alle unendliche Zweige von T' enthält, berechnet werden.

Idee:
Entwickle zunächst einen Algorithmus zur Berechnung von T'' und erweitere dann diesen derart, dass x ausgegeben wird.

Durchführung:
Für $n \in \mathbb{N}$ berechne $g(n)$ die Anzahl der guten Strings der Länge n. Betrachten wir

$$g := \limsup_{n \to \infty} g(n)$$

Da die Folge $g(n)$ beschränkt ist, existiert g.
Da g ein Häufungspunkt ist, finden wir in jeder ε-Umgebung von g unendlich viele Folgeglieder.

Für $\varepsilon = \frac{1}{2}$ folgt somit

(*) Es gibt unendlich viele n mit $g(n) = g$.

Da g größter Häufungspunkt ist, gilt.

(**) Es gibt nur endlich viele n mit $g(n) > g$

Andernfalls gäbe es einen Häufungspunkt, der größer als g ist.

(**) $\Rightarrow$

(***) $\exists\, n_0 \in \mathbb{N}$ mit $g(n) \leq g \;\forall\, n \geq n_0$.

Ein Level $n \geq n_0$ mit $g(n) = g$ heißt <u>vollständig</u>.

(*) $\Rightarrow$

Es gibt unendlich viele vollständige Levels.

<u>Ziel</u>:
Konstruktion eines Algorithmus O_7, der n_0 und g als Eingabe erhält und T'' berechnet.

O_7 zählt alle guten Strips auf bis das erste vollständige Level $n \geq n_0$ berechnet ist.
Dann werden alle Pfade von der Wurzel zu einem der g berechneten Knoten auf Level n zu T'' hinzugefügt. Die Berechnung der Knoten auf Level n setzt die Berechnung dieser Pfade voraus.

Beobachtung:

- Alle noch laufenden Berechnungen bzgl. Knoten auf Levels $< n$ können abgebrochen werden, da diese zu guten Strings der Länge $< n$ führen oder sowieso nicht terminieren. Ansonsten wäre $g(n) > g$, was wegen $n \geq n_0$ nicht sein kann.

Daher bricht der Algorithmus O_1 alle noch laufenden Berechnungen des Aufzählungsalgorithmus ab.

- Der bisher konstruierte Teilbaum von T'' korrespondiert zu g guten Strings $y_1, y_2, \ldots, y_g$.

Der Algorithmus O_1 startet nun mit dem zuletzt konstruierten vollständigen Level n und den korrespondierenden guten Strings $y_1, y_2, \ldots, y_g$ der Länge n und zählt, bis das nächste vollständige Level $> n$ berechnet ist, alle guten Strings, deren Präfixe aus $\{y_1, y_2, \ldots, y_g\}$ sind, auf. Alle Pfade von einem Knoten auf Level n zu einem Knoten auf dem neuen vollständigen Level werden hinzugefügt. Alle laufenden Berechnungen des Aufzählungsalgorithmus werden abgebrochen.

u.s.w.

Da der Algorithmus O_1 aus T' nur Pfade ent-

lent, die zu guten Strings endliche Länge korrespondieren, enthält T'' alle unendliche gute Strings.

Ziel:
Erweiterung von OT zu einem Algorithmus OT_x, der x ausgibt.

Sei $h \leq g$ die Anzahl der unendlichen guten Strings.

Beobachtung:
Zwei unterschiedliche unendliche gute Strings starten gemeinsam in der Wurzel von T'', haben einen Teilpfad, der mindestens die Wurzel enthält, gemeinsam, trennen sich in einem Knoten und treffen danach niemals wieder aufeinander.

$\Rightarrow$
$\exists$ ein erstes Level n_1, auf dem alle h unendliche gute Strings unterschiedliche Knoten durchlaufen

Sei q das erste vollständige Level mit

$$q \geq \max\{n_0, n_1\}.$$

Der Algorithmus OT_x erhält als Eingabe

- Präfix $x_1 x_2 \dots x_q$ von x, n_0 und g.

O_T berechnet sukzessive den unendlichen String x, indem er sukzessive den zu $x_1x_2x_3\ldots$ korrespondierende Zweig in T'' konstruiert. Dabei liegt der letzte Knoten v_N des bisher berechneten Teils des Zweiges stets in einem vollständigen Level. Im nachfolgenden Algorithmus O_T bezeichnet

- N stets den Index des Levels dieses letzten Knotens des bisher berechneten Teils des Zweiges und
- n stets den Index des zuletzt berechneten vollständigen Levels.

O_T berechnet die Ausgabe x wie folgt:

(1) Berechne T'' bis Level q;
Gib $x_1x_2\ldots x_q$ aus und streiche alle Knoten aus T'', die nicht auf dem zu $x_1x_2\ldots x_q$ korrespondierenden Pfad liegen;
Sei v_q derjenige Knoten auf Level q, der in T'' verbleiben ist;
$N := q$;

(2) Berechne T'' bis zum nächsten vollständigen Level n;

(3) Falls auf einem der vollständigen Level p mit $N < p \leq n$ nur ein Knoten v_p existiert, dann

- gib die Beschriftung des Pfades von v_N nach v_p aus und
- $N := p$.

(4) Berechne T'' bis zum nächsten vollständigen Level n;
Streiche aus T'' alle Knoten, die nur zu Knoten, die zu abgebrochenen Berechnungen korrespondieren, führen;
Gehe zu (3).

Bemerkung:
Obwohl es vorher nicht der Fall war, kann nach Durchführung von Schritt (4), aufgrund der darin erfolgten Streichungen bereits konstruierter Knoten, ein Level p mit $N < p < n$ existieren der nur Knoten v_p enthält.

Wegen $q \geq n_1$ haben alle zwischenzeitlich konstruierte Zweige in T'', die nicht zum String $x_1 x_2 \ldots$ korrespondieren, endliche Länge
$\Rightarrow$
Diese werden irgendwann in Schritt (4) wieder entfernt.

Der zu $x = x_1 x_2 x_3 \ldots$ korrespondierende Zweig hat unendliche Länge und wird daher niemals entfernt.

Mittels vollständiger Induktion kann gezeigt werden, dass für alle von $O_T{}_x$ berechneten vollständigen Level p irgendwann gilt:

Level p enthält exakt einen Knoten v_p.

$\Rightarrow$

$O_T{}_x$ gibt in der Tat den unendlichen String x aus. ∎

Übung
Arbeiten Sie den Algorithmus $O_T{}_x$ und den Korrektheitsbeweis aus.

Sei Ω die Menge aller unendlichen Folgen von Nullen und Einsen. Für $x \in \{0,1\}^*$ sei

$$\Omega_x := \{ z \in \Omega \mid x \text{ ist Präfix von } z \}.$$

Satz 2.10 besagt, dass alle berechenbare Elemente in Ω komprimierbar sind.

Frage:
Gibt es ein $x \in \Omega$ und eine Konstante c, so dass jeder endliche Präfix von x c-nichtkomprimierbar ist?

Folgender Satz beantwortet diese Frage.

Satz 2.11

Es existiert eine Konstante c, so dass für jede unendliche Folge $x = x_1 x_2 x_3 \ldots \in \Omega$ für unend-

lich viele n

$$K(x_1 x_2 \dots x_n) \leq n - \log n + c$$

erfüllt ist.

Beweis:

Die Idee der Konstruktion basiert darauf, dass die harmonische Reihe $\sum_{i=1}^{\infty} \frac{1}{i}$ divergiert.

Idee:

Konstruktion einer unendlichen Folge $A_1, A_2, A_3, \dots$ von Mengen, die folgende Eigenschaften besitzt:

1) Jedes A_i, $i \in \mathbb{N}$ besteht aus Strings der Länge i.
2) $|A_i| \leq \frac{2^i}{i} \quad \forall i \in \mathbb{N}$.
3) $\forall x = x_1 x_2 x_3 \dots x_i \dots \in \Omega$ existieren unendlich viele i, so dass $x_1 x_2 \dots x_i \in A_i$.

Durchführung:

Wir konstruieren die Folge $A_1, A_2, \dots$ derart, dass diese von links nach rechts in Blöcke $A_\ell, A_{\ell+1}, \dots, A_{\ell+k}$ unterteilt werden können, so dass jeder dieser Blöcke Ω ganz überdeckt. D.h.,

$$\Omega = \bigcup_{i=\ell}^{\ell+k} \bigcup_{x_j \in A_i} \Omega_{x_j}.$$

Annahme:

Die Konstruktion des vorangegangenen Blocks endet mit $A_{\ell-1}$, so dass wir die Konstruktion des aktuellen Blocks mit A_ℓ starten.

Konstruktion von A_ℓ:

- Betrachte die Strings aus $\{0,1\}^\ell$ in ihrer lexikographischen Ordnung und wähle die ersten $\frac{2^\ell}{\ell}$ Strings in dieser Ordnung als Elemente von A_ℓ.

Effekt:

A_ℓ überdeckt exakt einen Anteil von $\frac{1}{\ell}$ von Ω

Für die darauf folgenden Mengen $A_{\ell+j}$ müssen wir darauf achten, dass wir keine Strings für $A_{\ell+j}$ auswählen, die bereits überdeckt sind.

$\Rightarrow$

Wir starten in der lexikographischen Ordnung von $\{0,1\}^{\ell+j}$ mit dem ersten String, für den kein Präfix bereits in

$$\bigcup_{i=\ell}^{\ell+j-1} A_i$$

ist. Wir wählen die nächsten $\frac{2^{\ell+j}}{\ell+j}$ als Elemente von $A_{\ell+j}$ aus.

Effekt:
Falls $\frac{2^{\ell+j}}{\ell+j}$ viele Elemente gewählt werden konnten, dann überdeckt $A_{\ell+j}$ einen Anteil von $\frac{1}{\ell+j}$ von Ω.

Die Konstruktion eines Blockes $A_\ell, A_{\ell+1}, \dots, A_{\ell+k}$ terminiert, sobald die Anzahl der Elemente in $\{0,1\}^{\ell+k}$, für die kein Präfix bereits in $\bigcup_{i=\ell}^{\ell+k-1} A_i$ hinzugenommen wurde, höchstens $\frac{2^{\ell+k}}{\ell+k}$ ist.

$\Rightarrow$

$$k = \min\left\{ j \in \mathbb{N} \mid \sum_{i=\ell}^{\ell+j} \frac{1}{i} \geq 1 \right\}.$$

Die harmonische Reihe $\sum_{i=1}^{\infty} \frac{1}{i}$ divergiert

$\Rightarrow$

Wir konstruieren auf diese Art und Weise unendlich viel Blöcke.

(Würden wir nur endlich viele Blöcke konstruieren, dann wäre die harmonische Reihe beschränkt. Da die zur harmonischen Reihe korrespondierende Folge $H_n := \sum_{i=1}^{n} \frac{1}{i}$ streng monoton wächst, würde die harmonische Reihe konvergieren, was ja nicht der Fall ist.)

Beobachtung:

Wenn wir die Strings in $A_1, A_2, A_3, \ldots$ mit Eins beginnend in der Reihenfolge, in der sie zur Folge dazugenommen werden, nummerieren, dann genügt die Nummer eines Strings, um diesen zu rekonstruieren. Gegeben $num(y)$ eines Strings $y \in \bigcup_i A_i$ genügt es, obiges Konstruktionsverfahren durchzuführen, dabei mitzuzählen und zu terminieren, wenn der String mit Nummer $num(y)$ zur Folge hinzugefügt wird. Dies ist dann der gesuchte String.

$\Rightarrow$

$$K(x_1 x_2 \ldots x_n) \leq |num(x_1 x_2 \ldots x_n)| + c$$

$$\forall\, x_1 x_2 \ldots x_n \in \bigcup_i A_i ,$$

wobei c eine Konstante ist.

Es genügt somit zu zeigen, dass eine Konstante d existiert, so dass $\forall\, i \in \mathbb{N}$

$$\sum_{j=1}^{i} |A_j| \leq \sum_{j=1}^{i} \frac{2^j}{j} \leq d \cdot \frac{2^i}{i} .$$

Da in obiger Summe jeder Summand in etwa zweimal so groß ist wie sein Vorgänger und wir wissen, dass $\sum_{j=0}^{\infty} \frac{1}{2^j} = 2$, haben wir

die Intuition, dass dies wohl stimmen muss. Um uns davon zu überzeugen, dass unsere Intuition auch richtig ist, klammern wir in obiger Summe $\frac{2^i}{i}$ aus. Wir erhalten dann

$$\sum_{j=1}^{i} \frac{2^j}{j} = \frac{2^i}{i}\left(1 + \frac{i}{2^1(i-1)} + \frac{i}{2^2(i-2)} + \ldots + \frac{i}{2^{i-1}\cdot 1}\right)$$

Da in

$$\frac{i}{2^1(i-1)} + \frac{i}{2^2(i-2)} + \frac{i}{2^3(i-3)} + \ldots + \frac{i}{2^{i-1}\cdot 1}$$

die erste Hälfte der Summe offensichtlich größer als die zweite ist, gilt

$$\sum_{j=1}^{i} \frac{2^j}{j} \leq \frac{2^i}{i}\left(1 + 2\left(\frac{i}{2^1(i-1)} + \frac{i}{2^2(i-2)} + \ldots + \frac{i}{2^{i/2}\cdot i/2}\right)\right)$$

Wegen $2 \geq \frac{i}{i-r}$ für $r \leq \frac{i}{2}$ erhalten wir somit

$$\sum_{j=1}^{i} \frac{2^j}{j} \leq \frac{2^i}{i}\left(1 + 4\left(\frac{1}{2} + \frac{1}{4} + \ldots + \frac{1}{2^{i/2}}\right)\right)$$

$$\leq 5\frac{2^i}{i},$$

womit der Satz bewiesen ist.

■