

Die Quellencodierung hat die Aufgabe, ein beliebiges (analoges oder zeitdiskretes) Quellsignal in einen binären Datenstrom mit möglichst geringer Rate zu überführen. Wir können uns ohne Beschränkung der Allgemeinheit auf *diskrete Datenquellen* beschränken, d. h. auf Datenquellen, welche eine Quellsymbolsequenz erzeugen, weil jedes bandbegrenzte Signal gemäß dem Abtasttheorem ohne jeglichen Informationsverlust durch eine Sequenz von Abtastwerten repräsentiert werden kann. Die Hauptaufgabe der Quellencodierung besteht im Sinne der Informationstheorie nicht in der Digitalisierung, sondern in einer Reduzierung der Datenmenge, d. h. einer *Datenkompression*.

Die Quellencodierung kann *verlustlos* oder *verlustbehaftet* sein. Man spricht von einer verlustlosen Quellencodierung, wenn die Quellsymbolsequenz aus dem komprimierten Bitstrom verzerrungsfrei rekonstruiert werden kann. Ein verlustloser Quellencodierer bewirkt eine Redundanzreduktion, niemals jedoch auch eine Irrelevanzreduktion. Im Rahmen dieses Kap. 2 werden ausschließlich verlustlose Quellencodierverfahren betrachtet, während die verlustbehaftete Quellencodierung in Kap. 4 im Rahmen der sog. *Rate-Distortion-Theorie* behandelt wird.

2.1 Gedächtnisfreie Quellen

Eine diskrete Datenquelle wird als *gedächtnisfrei* bezeichnet, wenn deren Ausgangssymbole statistisch unabhängig sind. Andernfalls wird sie als *gedächtnisbehaftet* bezeichnet. Ein Beispiel einer gedächtnisfreien Quelle ist das mehrfache Werfen einer Münze: Jeder Münzwurf ist von vorausgegangenen Münzwürfen unabhängig, selbst wenn die Münze nicht fair ist. Beispiele für gedächtnisbehaftete Quellen sind Texte oder Bilder: Benachbarte Buchstaben oder Pixel sind typischerweise korreliert.

Es werden in diesem Abschn. 2.1 zunächst gedächtnisfreie Quellen angenommen. Wir beschränken uns dabei auf Sequenzen von statistisch unabhängigen, identisch verteil-

ten („independent identically distributed (i. i. d.)“) Zufallsvariablen. Gedächtnisbehaftete Quellen werden im anschließenden Abschn. 2.2 untersucht.

2.1.1 Typische Sequenzen und asymptotische Äquipartitionseigenschaft

Es sei X eine Zufallsvariable mit Werten über einem Alphabet \mathcal{X} der Mächtigkeit $L = |\mathcal{X}|$. Die Entropie dieser Zufallsvariablen wird mit $H(X)$ bezeichnet. X_1, X_2, \dots, X_n sei eine Sequenz von statistisch unabhängigen, identisch verteilten Zufallsvariablen. (Man beachte die Verwechslungsmöglichkeit zwischen „identisch verteilten“ Zufallsvariablen und einer „gleichverteilten“ Zufallsvariable: Bei einer Sequenz von „identisch verteilten“ Zufallsvariablen besitzt jede Zufallsvariable die gleiche Statistik, während bei einer „gleichverteilten“ Zufallsvariable jedes Ereignis die gleiche Wahrscheinlichkeit besitzt.) Es gibt folglich L^n mögliche Sequenzen. Eine Realisierung wird mit $\mathbf{x} = [x_1, x_2, \dots, x_n] \in \mathcal{X}^n$ bezeichnet.

Beispiel 2.1.1 Trickmünze

Wir betrachten eine binäre Zufallsvariable ($L = 2$), z. B. das Werfen einer Trickmünze. Die Wahrscheinlichkeit für „Kopf“ sei p , die Wahrscheinlichkeit für „Zahl“ sei $q = 1 - p$, wobei bei einer Trickmünze definitionsgemäß $p \neq 0,5$ gilt. Wir werfen die Trickmünze n mal. Die Erfahrung lehrt uns, dass wir ungefähr $n \cdot p$ mal „Kopf“ würfeln, falls n hinreichend groß ist. \diamond

Man teilt deshalb die Gesamtmenge der L^n möglichen Sequenzen in zwei Teilmengen und lässt n gegen unendlich gehen:

1. Die erste Teilmenge umfasst alle sog. *typischen Sequenzen*. Für das Beispiel der Trickmünze umfasst die Menge der typischen Sequenzen alle n -Tupel, die „ungefähr“ $n \cdot p$ mal „Kopf“ aufweisen.
2. Die zweite Teilmenge umfasst alle restlichen n -Tupel. Dies ist die Menge der *atypischen Sequenzen*.

Bei diesem intuitiven Vorgehen ist die Menge der typischen Sequenzen mathematisch nicht exakt definiert. Um dieses Problem zu lösen, betrachten wir zunächst folgenden Satz:

► **Satz 2.1.1** Für $n \rightarrow \infty$ stimmen die Entropie $H(X)$ und der arithmetische Mittelwert der Eigeninformation, $\overline{I(X = x_i)}$, überein.

► **Beweis 2.1.1** Das *Gesetz der großen Zahlen* (siehe Abschn. 31.5) besagt, dass für eine Sequenz von statistisch unabhängigen, identisch verteilten Zufallsvariablen X_1, X_2, \dots, X_n gilt:

$$E\{f(X)\} = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n f(x_i) = \overline{f(x_i)}. \quad (2.1)$$

Als direkte Konsequenz folgt

$$H(X) = E\{I(X = x)\} = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n I(X = x_i) = \overline{I(X = x_i)}. \quad (2.2)$$

□

Dieser Satz motiviert folgende Definition:

► **Definition 2.1.1** Ein Ereignis $\mathbf{x} = [x_1, x_2, \dots, x_n]$ wird als ϵ -typische Sequenz bezeichnet, wenn sich der arithmetische Mittelwert der Eigeninformation betragsmäßig von der Entropie $H(X)$ maximal um eine beliebige Zahl $\epsilon \in \mathbb{R}^+$ unterscheidet. Die Menge $\mathcal{A}_\epsilon(X)$ der ϵ -typischen Sequenzen lautet somit:

$$\mathcal{A}_\epsilon(X) = \left\{ \mathbf{x} : \left| \underbrace{-\frac{1}{n} \log_b p_X(\mathbf{x})}_{\overline{I(X=\mathbf{x})}} - \underbrace{H(X)}_{E\{I(X=x)\}} \right| \leq \epsilon \right\}. \quad (2.3)$$

Als Konsequenz dieser Definition ergibt sich die sog. *asymptotische Äquipartitionseigenschaft* (AEP):

► **Satz 2.1.2 (Asymptotische Äquipartitionseigenschaft)** Für jedes $\epsilon > 0$ existiert eine ganze Zahl n , so dass $\mathcal{A}_\epsilon(X)$ die folgenden drei Bedingungen erfüllt:

$$\mathbf{x} \in \mathcal{A}_\epsilon(X) \Rightarrow \left| -\frac{1}{n} \log_b p_X(\mathbf{x}) - H(X) \right| \leq \epsilon \quad (2.4)$$

$$1 \geq P(\mathbf{x} \in \mathcal{A}_\epsilon(X)) \geq 1 - \epsilon, \quad \text{d. h.} \quad 0 \leq P(\mathbf{x} \notin \mathcal{A}_\epsilon(X)) < \epsilon \quad (2.5)$$

$$(1 - \epsilon) b^{n(H(X) - \epsilon)} \leq |\mathcal{A}_\epsilon(X)| \leq b^{n(H(X) + \epsilon)}, \quad (2.6)$$

wobei $|\mathcal{A}_\epsilon(X)|$ die Anzahl der ϵ -typischen Sequenzen ist. (Der Begriff asymptotische Äquipartitionseigenschaft stammt ebenso wie die Entropie aus der Thermodynamik.)

► **Beweis 2.1.2**

- Bedingung (2.4) folgt direkt aus der Definition von $\mathcal{A}_\epsilon(X)$.
- Bedingung (2.5) folgt aus der Konvergenz von $-\frac{1}{n} \log_b p_X(\mathbf{x})$ in der Definition von $\mathcal{A}_\epsilon(X)$. Diese Bedingung besagt, dass die Wahrscheinlichkeit, dass ein Zufallsgenerator eine atypische Sequenz generiert, kleiner als ϵ (und folglich beliebig klein) ist.
- Bedingung (2.6) gibt eine Abschätzung über die Anzahl der ϵ -typischen Sequenzen und folgt aus

$$1 = \sum_{\mathbf{x}} p_X(\mathbf{x}) \geq \sum_{\mathcal{A}_\epsilon(X)} p_X(\mathbf{x}) \geq \sum_{\mathcal{A}_\epsilon(X)} b^{-n(H(X) + \epsilon)} = |\mathcal{A}_\epsilon(X)| b^{-n(H(X) + \epsilon)} \quad (2.7)$$

und

$$1 - \epsilon \leq \sum_{\mathcal{A}_\epsilon(\mathbf{X})} p_{\mathbf{X}}(\mathbf{x}) \leq \sum_{\mathcal{A}_\epsilon(\mathbf{X})} b^{-n(H(\mathbf{X})-\epsilon)} = |\mathcal{A}_\epsilon(\mathbf{X})| b^{-n(H(\mathbf{X})-\epsilon)}. \quad (2.8)$$

□

► **Folgesatz 2.1.3** Für alle ϵ -typischen Sequenzen $\mathbf{x} \in \mathcal{A}_\epsilon(\mathbf{X})$ gilt:

$$b^{-n(H(\mathbf{X})+\epsilon)} \leq p_{\mathbf{X}}(\mathbf{x}) \leq b^{-n(H(\mathbf{X})-\epsilon)}. \quad (2.9)$$

► **Beweis 2.1.3** Der Folgesatz ergibt sich durch eine Fallunterscheidung unmittelbar aus der Bedingung (2.4). □

Der Folgesatz definiert die exakte Vorschrift um zu entscheiden, ob eine Sequenz \mathbf{x} mit gegebener Wahrscheinlichkeit $p_{\mathbf{X}}(\mathbf{x})$ ein Kandidat für eine ϵ -typische Sequenz ist oder nicht. (Zusätzlich muss für ϵ -typische Sequenzen die Bedingung 2 der asymptotischen Äquipartitionseigenschaft erfüllt sein.)

Die *Eigenschaften von typischen Sequenzen* lassen sich am einfachsten diskutieren, wenn wir die Basis $b = 2$ annehmen und $\epsilon \rightarrow 0$ gehen lassen:

- Alle ϵ -typischen Sequenzen haben gemäß (2.9) ungefähr die gleiche Wahrscheinlichkeit $p_{\mathbf{X}}(\mathbf{x}) \approx 2^{-nH(\mathbf{X})}$.
- Die Summe der Wahrscheinlichkeiten von ϵ -typischen Sequenzen ist gemäß Bedingung (2.5) nahezu gleich eins.
- Dennoch ist die Anzahl der ϵ -typischen Sequenzen, bezogen auf die Gesamtzahl aller möglichen Sequenzen, gemäß Bedingung (2.6) für $H(\mathbf{X}) < \log_2 L$ sehr klein: $|\mathcal{A}_\epsilon(\mathbf{X})| \approx 2^{nH(\mathbf{X})} \ll L^n$. Die Anzahl der ϵ -typischen Sequenzen ist umso kleiner, je kleiner die Entropie $H(\mathbf{X})$ ist. Obwohl es relativ wenige ϵ -typische Sequenzen gibt, tragen sie fast zur gesamten Wahrscheinlichkeit bei.
- In der Informationstheorie werden deshalb die atypischen Sequenzen vernachlässigt.
- Die ϵ -typischen Sequenzen sind jedoch nicht die wahrscheinlichsten Sequenzen! (Beispiel: Die Sequenz mit n mal „Zahl“ ist für $p < 1/2$ wahrscheinlicher als jede typische Sequenz.)

Beispiel 2.1.2

Gegeben sei eine binäre Zufallsvariable X ($L = 2$) mit den Ereignissen $x^{(1)} = \text{„Kopf“}$ und $x^{(2)} = \text{„Zahl“}$, wobei $P(X = x^{(1)}) := p = 2/5$ und $P(X = x^{(2)}) = 1 - p := q = 3/5$. Aus der binären Entropiefunktion ergibt sich $H(X) = h(p = 2/5) = 0,971$ bit. Wir wählen zunächst $n = 5$ und $\epsilon = 0,0971$ (10 % von $H(X)$). Aus dem Folgesatz (2.9) folgt, dass für ϵ -typische Sequenzen \mathbf{x} die Ungleichung $0,0247 \leq p_{\mathbf{X}}(\mathbf{x}) \leq 0,0484$ erfüllt sein muss. In Tab. 2.1 sind die $L^n = 2^5 = 32$ möglichen Sequenzen aufgelistet (K: „Kopf“, Z: „Zahl“). Demnach sind von den zweiunddreißig möglichen Sequenzen zehn

n -Tupel Kandidaten für ϵ -typische Sequenzen. (Dies ist ein kombinatorisches Problem, kein statistisches.) Diese Kandidaten sind mit $*$ markiert. Der Anteil von ϵ -typischen Sequenzen an der Gesamtwahrscheinlichkeit, $P(\mathbf{x} \in \mathcal{A}_\epsilon(\mathbf{X}))$, beträgt allerdings (nur) 34,6 %, weil $n = 5$ sehr klein ist. Somit ist die Voraussetzung $P(\mathbf{x} \in \mathcal{A}_\epsilon(\mathbf{X})) \geq 1 - \epsilon$ gemäß der asymptotischen Äquipartitionseigenschaft (2.5) nicht erfüllt. Daher gibt es in diesem einführenden Beispiel keine ϵ -typische Sequenz, die alle drei Voraussetzungen der asymptotischen Äquipartitionseigenschaft erfüllen. Dies liegt insbesondere daran, dass $n = 5$ zu klein gewählt wurde. \diamond

Beispiel 2.1.3

Für das gleiche Szenario wie im letzten Beispiel betrachten wir nun große Sequenzlängen n . Wir wählen $p = 0,11$ (somit ergibt sich $H(X) = h(p) = 0,5$ bit) und $\epsilon = 0,05$ (10 % von $H(X)$). Die Ergebnisse sind in Tab. 2.2 zusammengefasst. Man erkennt, dass ab etwa $n = 2000$ für die gegebenen Parameter ϵ -typische Sequenzen existieren.

Die Anzahl der ϵ -typischen Sequenzen der Länge $n = 10.000$ beträgt nur $2^{5471,45}$. Der 10^{-1361} -te Teil aller Sequenzen hat dennoch einen Beitrag zur Gesamtwahrscheinlichkeit von praktisch 100 %. Die Datensequenz kann um fast 50 % komprimiert werden. \diamond

2.1.2 Simultan Typische Sequenzen

Wir betrachten nun Zufallsvariablen X und Y mit der Verbundwahrscheinlichkeitsfunktion $p_{XY}(x, y)$ und bezeichnen mit $[(X_1, Y_1), (X_2, Y_2), \dots, (X_n, Y_n)]$ eine Sequenz von Paaren (X_i, Y_i) , wobei $i \in \{1, 2, \dots, n\}$ der Laufindex ist.

Beispiel 2.1.4

X_i ist das i -te Kanaleingangssymbol, Y_i ist das i -te Kanalausgangssymbol. \diamond

► **Definition 2.1.2** Die Menge $\mathcal{A}_\epsilon(\mathbf{X}, \mathbf{Y})$ der *simultan ϵ -typischen Sequenzen* $(\mathbf{x}, \mathbf{y}) = [(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)]$ der Länge n wird wie folgt definiert:

$$\mathcal{A}_\epsilon(\mathbf{X}, \mathbf{Y}) := \left\{ (\mathbf{x}, \mathbf{y}) : \begin{aligned} &\left| -\frac{1}{n} \log_b p_X(\mathbf{x}) - H(X) \right| \leq \epsilon, \\ &\left| -\frac{1}{n} \log_b p_Y(\mathbf{y}) - H(Y) \right| \leq \epsilon, \\ &\left| -\frac{1}{n} \log_b p_{XY}(\mathbf{x}, \mathbf{y}) - H(X, Y) \right| \leq \epsilon \end{aligned} \right\}, \quad (2.10)$$

wobei $p_{XY}(\mathbf{x}, \mathbf{y}) = \prod_{i=1}^n p_{XY}(x_i, y_i)$.

Tab. 2.1 Ergebnisse zu Beispiel 2.1.2 ($N = 5$, $p = 2/5$, $\epsilon \approx 0,1$)

Sequenz ($n = 5$)	Wahrscheinlichkeit	ϵ -typ. Sequenz ?	Codierung
K K K K K	$p^5(1-p)^0 = 1,02 \cdot 10^{-2}$		
K K K K Z	$p^4(1-p)^1 = 1,54 \cdot 10^{-2}$		
K K K Z K	$p^4(1-p)^1 = 1,54 \cdot 10^{-2}$		
K K K Z Z	$p^3(1-p)^2 = 2,30 \cdot 10^{-2}$		
K K Z K K	$p^4(1-p)^1 = 1,54 \cdot 10^{-2}$		
K K Z K Z	$p^3(1-p)^2 = 2,30 \cdot 10^{-2}$		
K K Z Z K	$p^3(1-p)^2 = 2,30 \cdot 10^{-2}$		
K K Z Z Z	$p^2(1-p)^3 = 3,46 \cdot 10^{-2}$	*	0000
K Z K K K	$p^4(1-p)^1 = 1,54 \cdot 10^{-2}$		
K Z K K Z	$p^3(1-p)^2 = 2,30 \cdot 10^{-2}$		
K Z K Z K	$p^3(1-p)^2 = 2,30 \cdot 10^{-2}$		
K Z K Z Z	$p^2(1-p)^3 = 3,46 \cdot 10^{-2}$	*	0001
K Z Z K K	$p^3(1-p)^2 = 2,30 \cdot 10^{-2}$		
K Z Z K Z	$p^2(1-p)^3 = 3,46 \cdot 10^{-2}$	*	0010
K Z Z Z K	$p^2(1-p)^3 = 3,46 \cdot 10^{-2}$	*	0011
K Z Z Z Z	$p^1(1-p)^4 = 5,18 \cdot 10^{-2}$		
Z K K K K	$p^4(1-p)^1 = 1,54 \cdot 10^{-2}$		
Z K K K Z	$p^3(1-p)^2 = 2,30 \cdot 10^{-2}$		
Z K K Z K	$p^3(1-p)^2 = 2,30 \cdot 10^{-2}$		
Z K K Z Z	$p^2(1-p)^3 = 3,46 \cdot 10^{-2}$	*	0100
Z K Z K K	$p^3(1-p)^2 = 2,30 \cdot 10^{-2}$		
Z K Z K Z	$p^2(1-p)^3 = 3,46 \cdot 10^{-2}$	*	0101
Z K Z Z K	$p^2(1-p)^3 = 3,46 \cdot 10^{-2}$	*	0110
Z K Z Z Z	$p^1(1-p)^4 = 5,18 \cdot 10^{-2}$		
Z Z K K K	$p^3(1-p)^2 = 2,30 \cdot 10^{-2}$		
Z Z K K Z	$p^2(1-p)^3 = 3,46 \cdot 10^{-2}$	*	0111
Z Z K Z K	$p^2(1-p)^3 = 3,46 \cdot 10^{-2}$	*	1000
Z Z K Z Z	$p^1(1-p)^4 = 5,18 \cdot 10^{-2}$		
Z Z Z K K	$p^2(1-p)^3 = 3,46 \cdot 10^{-2}$	*	1001
Z Z Z K Z	$p^1(1-p)^4 = 5,18 \cdot 10^{-2}$		
Z Z Z Z K	$p^1(1-p)^4 = 5,18 \cdot 10^{-2}$		
Z Z Z Z Z	$p^0(1-p)^5 = 7,78 \cdot 10^{-2}$		
$\Sigma = 1$		$\Sigma = 0,346$	

Tab. 2.2 Ergebnisse zu Beispiel 2.1.3 ($p = 0,11$, $\epsilon = 0,05$)

n	$(1 - \epsilon)2^{n(H(X) - \epsilon)}$	$ \mathcal{A}_\epsilon(X) $	$2^{n(H(X) + \epsilon)}$	$P(\mathbf{x} \in \mathcal{A}_\epsilon(X)) \geq 1 - \epsilon ?$
100	$2^{44,92}$	$2^{50,10}$	$2^{54,99}$	$0,3676 < 0,9500$
200	$2^{89,91}$	$2^{105,38}$	$2^{109,98}$	$0,5711 < 0,9500$
500	$2^{224,88}$	$2^{269,19}$	$2^{274,96}$	$0,7760 < 0,9500$
1000	$2^{449,84}$	$2^{541,87}$	$2^{549,92}$	$0,9049 < 0,9500$
2000	$2^{899,76}$	$2^{1090,53}$	$2^{1099,83}$	$0,9834 > 0,9500$
5000	$2^{2249,51}$	$2^{2731,75}$	$2^{2749,58}$	$0,9998 > 0,9500$
10.000	$2^{4499,09}$	$2^{5471,45}$	$2^{5499,16}$	$1,0000 > 0,9500$
20.000	$2^{8998,25}$	$2^{10951,35}$	$2^{10998,32}$	$1,0000 > 0,9500$
50.000	$2^{22495,72}$	$2^{27389,08}$	$2^{27495,80}$	$1,0000 > 0,9500$
100.000	$2^{44991,52}$	$2^{54787,76}$	$2^{54991,60}$	$1,0000 > 0,9500$

Ein Paar von Sequenzen \mathbf{x} und \mathbf{y} ist demnach simultan ϵ -typisch, wenn \mathbf{x} , \mathbf{y} und (\mathbf{x}, \mathbf{y}) ϵ -typisch sind, vgl. Abb. 2.1.

► **Satz 2.1.4 (Asymptotische Äquipartitionseigenschaft (AEP))** Für jedes $\epsilon > 0$ existiert eine ganze Zahl n , so dass $\mathcal{A}_\epsilon(X, Y)$ die folgenden drei Bedingungen erfüllt:

$$(\mathbf{x}, \mathbf{y}) \in \mathcal{A}_\epsilon(X, Y) \Rightarrow \left| -\frac{1}{n} \log_b p_{XY}(\mathbf{x}, \mathbf{y}) - H(X, Y) \right| \leq \epsilon \quad (2.11)$$

$$1 \geq P((\mathbf{x}, \mathbf{y}) \in \mathcal{A}_\epsilon(X, Y)) \geq 1 - \epsilon, \quad \text{d. h.} \quad 0 \leq P((\mathbf{x}, \mathbf{y}) \notin \mathcal{A}_\epsilon(X, Y)) < \epsilon \quad (2.12)$$

$$(1 - \epsilon)b^{n(H(X, Y) - \epsilon)} \leq |\mathcal{A}_\epsilon(X, Y)| \leq b^{n(H(X, Y) + \epsilon)}, \quad (2.13)$$

wobei $|\mathcal{A}_\epsilon(X, Y)|$ die Anzahl der simultan ϵ -typischen Sequenzen ist.

Diese Bedingungen erklären sich analog zu denen der ϵ -typischen Sequenzen.

► **Beweis 2.1.4** Die Beweisführung ist identisch wie bei ϵ -typischen Sequenzen, wobei X durch (X, Y) ersetzt wird. \square

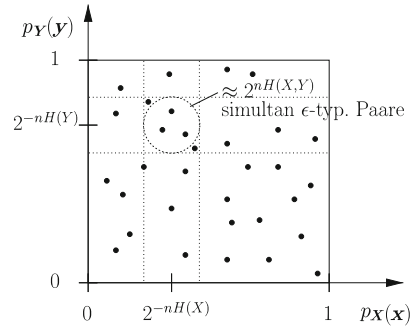
► **Folgesatz 2.1.5** Für alle simultan ϵ -typischen Sequenzen $(\mathbf{x}, \mathbf{y}) \in \mathcal{A}_\epsilon(X, Y)$ gilt:

$$b^{-n(H(X, Y) + \epsilon)} \leq p_{XY}(\mathbf{x}, \mathbf{y}) \leq b^{-n(H(X, Y) - \epsilon)}. \quad (2.14)$$

► **Beweis 2.1.5** Der Beweis folgt unmittelbar aus (2.11). \square

Mit Hilfe des Folgesatzes gelingt es zu entscheiden, ob Sequenzen (\mathbf{x}, \mathbf{y}) simultan ϵ -typisch sind oder nicht. Auf simultan ϵ -typische Sequenzen werden wir zwecks Beweis von Shannons Kanalcodiertheorem zurückkommen.

Abb. 2.1 Simultan ϵ -typische Sequenzen



2.1.3 Shannons Quellencodiertheorem

Wir betrachten nun das Blockschaltbild eines Kommunikationssystems mit Quellen- und Kanalcodierung gemäß Abb. 2.2. Die *diskrete, gedächtnisfreie Datenquelle* in Abb. 2.2 erzeugt eine Sequenz von n Quellensymbolen $\mathbf{q} = [q_1, q_2, \dots, q_n]$, wobei jedes Quellensymbol über einem endlichen Alphabet \mathcal{Q} mit $L = |\mathcal{Q}|$ Elementen definiert ist, dem sog. Quellensymbolalphabet. Die Entropie pro Quellensymbol wird mit $H(Q)$ bezeichnet. Der Quellencodierer generiert aus der Quellensymbolsequenz eine Datensequenz \mathbf{u} , dessen Elemente wir als binär annehmen. (Eine Rechtfertigung dieser Annahme folgt umgehend aus Shannons Quellencodiertheorem.) Die Datenübertragung sei fehlerfrei, d. h. $\hat{\mathbf{u}} = \mathbf{u}$. (Eine Rechtfertigung dieser Annahme folgt später aus Shannons Kanalcodiertheorem.) Dem Quellendecodierer stehe somit die Datensequenz \mathbf{u} zur Verfügung.

Eine fundamentale Frage ist: Wie groß ist die minimale Anzahl an Bits pro Quellensymbol, die der Quellencodierer im Mittel bereitstellen muss, damit aus der quellenkodierten Datensequenz \mathbf{u} die Quellensymbolsequenz \mathbf{q} beliebig genau rekonstruiert werden kann? Diese fundamentale Frage wird durch *Shannons Quellencodiertheorem* wie folgt beantwortet [Sha48]:

► **Satz 2.1.6 (Shannons Quellencodiertheorem)** Gegeben sei eine diskrete, gedächtnisfreie Quelle. Für eine verlustlose Quellencodierung ist es notwendig und hinreichend, wenn für die Codierung im Mittel $H(Q)$ bit/Quellensymbol verwendet werden, vorausgesetzt, die Länge n der Quellensymbolsequenz geht gegen unendlich. Die Umkehrung besagt: Weniger als $H(Q)$ bit/Quellensymbol reichen für eine verlustlose Quellencodierung nicht aus.

Verlustlose Verfahren zur Quellencodierung werden deshalb auch als *Entropiecodierung* bezeichnet (obwohl die mittlere Codewortlänge in der Praxis nur in Spezialfällen die Entropie $H(Q)$ erreicht, meist aber größer ist).

Gemäß der asymptotischen Äquipartitionseigenschaft kann die Codierung beispielsweise derart erfolgen, dass jede ϵ -typische Sequenz $\mathbf{q} \in \mathcal{A}_\epsilon(Q)$ mit $\lceil n(H(Q) + \epsilon) \rceil$ Binärzeichen fortlaufend adressiert wird. Dies ist in der rechten Spalte in Tab. 2.1 illustriert. Bei

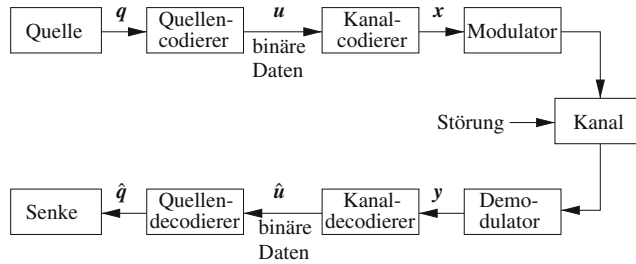


Abb. 2.2 Blockschaltbild eines Kommunikationssystems mit Quellen- und Kanalcodierung

dieser Art der Codierung wird immer eine binäre Datensequenz erzeugt. Die atypischen Sequenzen werden ignoriert. Weil atypische Sequenzen vernachlässigbar selten auftreten, spricht man dennoch von einer verlustlosen Quellencodierung.

► **Beweis 2.1.6** Für beliebige $\epsilon > 0$ gibt es ein hinreichend großes n , so dass eine Quellensymbolsequenz $\mathbf{q} = [q_1, q_2, \dots, q_n]$ mit $n(H(Q) + \epsilon)$ bit auf eine eindeutige Weise codiert werden kann, abgesehen von einer Menge atypischer Quellensymbolsequenzen, deren Gesamtwahrscheinlichkeit insgesamt kleiner als ϵ ist. Das folgt aus der Existenz einer Menge $\mathcal{A}_\epsilon(\mathbf{Q})$ von Sequenzen \mathbf{q} , welche die ersten beiden Bedingungen der AEP in Satz 2.1.2 erfüllen. Die rechte Ungleichung der dritten Bedingung ergibt, dass $n(H(Q) + \epsilon)$ bit hinreichend sind, während die zweite Bedingung garantiert, dass $P(\mathbf{q} \notin \mathcal{A}_\epsilon(\mathbf{Q})) < \epsilon$ ist.

Wenn wir im Gegensatz dazu nur $n(H(Q) - 2\epsilon)$ bit verwenden, so reichen die Codeworte nur für einen verschwindend geringen Teil der typischen Sequenzen, was aus der linken Ungleichung der dritten Bedingung folgt. Deshalb sind $H(Q)$ bit/Quellensymbol nicht nur hinreichend, sondern für lange Sequenzen auch notwendig, wenn die Wahrscheinlichkeit für eine perfekte Rekonstruktion gegen eins gehen soll. \square

2.1.4 Präfixfreie Quellencodierung

Gegeben sei eine diskrete Quelle. Die diskrete Quelle erzeuge eine Sequenz von n Zufallsvariablen Q_1, Q_2, \dots, Q_n . Die Sequenz der n Quellensymbole werde mit q_1, q_2, \dots, q_n bezeichnet, die entsprechenden Entropien mit $H(Q_1), H(Q_2), \dots, H(Q_n)$. Wir nehmen an, dass alle Quellensymbole über dem gleichen endlichen Alphabet $\mathcal{Q} = \{q^{(1)}, q^{(2)}, \dots, q^{(L)}\}$ der Mächtigkeit L definiert sind. Ferner nehmen wir an, dass die Quellensymbole statistisch unabhängig seien, d. h. die Quelle sei gedächtnisfrei. Deshalb ist es völlig ausreichend, wenn wir uns auf ein einziges Quellensymbol beschränken. Wir lassen den Laufindex weg und bezeichnen das betrachtete Quellensymbol mit q und dessen Entropie mit $H(Q)$.

Der Quelle werde gemäß Abb. 2.3 ein Quellencodierer nachgeschaltet. Der Quellencodierer sei verlustlos und erzeugt pro Quellensymbol q eine codierte Symbolsequenz \mathbf{u} , im Folgenden *Codewort* genannt. Dem i -ten Quellensymbol $q^{(i)}$ sei das Codewort $\mathbf{u}^{(i)}$

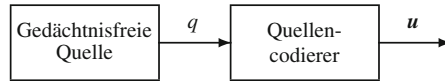


Abb. 2.3 Diskrete gedächtnisfreie Quelle mit nachgeschalteter Quellencodierung

zugeordnet, $i \in \{1, \dots, L\}$. Wir beschränken uns im Weiteren auf binäre Codewörter. Die Codewörter haben eine variable Länge. Sinnvollerweise werden häufig auftretenden Quellsymbolen kürzere Codewörter als unwahrscheinlichen Quellsymbolen zugeordnet. Alternativ ist es auch möglich, mehrere Quellsymbole auf Codewörter fester Länge abzubilden, d. h. eine variable Quellenwortlänge wird einer festen Codewortlänge zugeordnet. Informationstheoretisch betrachtet ergeben sich keine Unterschiede hinsichtlich der Effizienz. Deshalb wird die zweite Variante nicht weiter betrachtet.

Die Quellencodierung muss folgende Eigenschaften erfüllen:

1. Die Abbildung zwischen einem Quellsymbol $q^{(i)}$ und dem zugehörigen Codewort $\mathbf{u}^{(i)}$ muss eineindeutig sein, d. h. keine zwei Codewörter dürfen identisch sein: $\mathbf{u}^{(i)} \neq \mathbf{u}^{(j)}$ für alle $i \neq j$. Hieraus folgt, dass $H(Q) = H(U)$.
2. Kein Codewort darf *Präfix* eines längeren Codewortes sein. Somit kann ein Codewort erkannt werden, sobald dessen letztes Symbol empfangen wurde, auch bei einem kontinuierlich arbeitenden Quellencodierer.

Codes, die beide Bedingungen erfüllen, nennt man *präfixfrei*.

Beispiel 2.1.5 Präfixfreier Code

Tabelle 2.3 zeigt zwei Codes mit jeweils drei Codewörtern. Der Code auf der linken Seite ist präfixfrei, während der Code auf der rechten Seite nicht präfixfrei ist, weil $\mathbf{u}^{(1)}$ Präfix von $\mathbf{u}^{(2)}$ ist. \diamond

Präfixfreie Codes ermöglichen eine *kommafreie Übertragung*, wenn keine Übertragungsfehler auftreten. Bereits ein einziger Übertragungsfehler kann jedoch bewirken, dass eine verzerrungsfreie Rekonstruktion der Quellsymbole nicht mehr möglich ist. Folglich sind präfixfreie Codes anfällig gegenüber Übertragungsfehlern. Diesbezüglich hilfreich ist die Kanalcodierung, denn gemäß Shannons Kanalcodiertheorem kann die Fehlerwahrscheinlichkeit durch Kanalcodierung beliebig klein gemacht werden, wie wir später beweisen werden.

Wir bezeichnen die Länge des i -ten Codeworts $\mathbf{u}^{(i)}$ mit $w^{(i)}$, $i \in \{1, \dots, L\}$. Hieraus ergibt sich die *mittlere Codewortlänge pro Quellsymbol* zu

$$E\{W\} = \sum_{i=1}^L p_Q(q^{(i)}) w^{(i)} \quad [\text{bit/Quellsymbol}]. \quad (2.15)$$

Je kürzer die mittlere Codewortlänge, umso effizienter ist die Codierung.

Tab. 2.3 Präfixfreier Code (links) und nicht präfixfreier Code (rechts)

q	u	q	u
$q^{(1)}$	$u^{(1)} = [00]$	$q^{(1)}$	$u^{(1)} = [0]$
$q^{(2)}$	$u^{(2)} = [01]$	$q^{(2)}$	$u^{(2)} = [01]$
$q^{(3)}$	$u^{(3)} = [10]$	$q^{(3)}$	$u^{(3)} = [10]$

► **Definition 2.1.3 (Optimaler Code)** Ein binärer präfixfreier Code wird optimal genannt, wenn es keinen anderen binären Code mit einer kleineren mittleren Codewortlänge $E\{W\}$ pro Quellensymbol gibt.

► **Satz 2.1.7** Für die mittlere Codewortlänge eines optimalen binären präfixfreien Codes gilt:

$$H(Q) \leq E\{W\} < H(Q) + 1. \quad (2.16)$$

► **Beweis 2.1.7** Die vordere Ungleichung entspricht der Kernaussage von Shannons Quellencodiertheorem. Die hintere Ungleichung folgt aus

$$E\{W\} = \sum_{i=1}^L w^{(i)} p_Q(q^{(i)}) < \sum_{i=1}^L \left(\log_b \frac{1}{p_Q(q^{(i)})} + 1 \right) p_Q(q^{(i)}) = H(Q) + 1. \quad (2.17)$$

□

► **Bemerkung 2.1.1** Die vordere Ungleichung motiviert, dass die *Effizienz eines Quellencodierers* gemäß

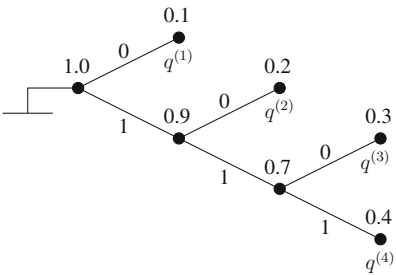
$$\eta := 100 \frac{H(Q)}{E\{W\}} \% \quad (2.18)$$

definiert wird, wobei $0 \leq \eta \leq 100 \%$. Erstrebenswert ist eine möglichst hohe Effizienz.

► **Bemerkung 2.1.2** Die hintere Ungleichung ist für optimale Codes notwendig, aber nicht hinreichend. Nicht jeder Code, für den $E\{W\} < H(Q) + 1$ gilt, ist ein optimaler Code. Ein Beispiel folgt unmittelbar.

In der Quellencodierung können Codewörter durch einen sog. *Codebaum* (kurz: Baum) illustriert werden. Ein Baum besteht aus Knoten, von denen Zweige ausgehen. Der Anfangsknoten wird Wurzel genannt. Jedem Endknoten wird genau ein diskretes Quellensymbol zugeordnet. Jeder Zweig entspricht einem Codesymbol. In einem binären Baum verlassen (bis zu) zwei Zweige einen Knoten. Adressiert man ohne Beschränkung der Allgemeinheit den oberen Zweig mit dem Codesymbol 0 und den unteren Zweig mit dem Codesymbol 1, so ist ein binärer präfixfreier Code garantiert.

Abb. 2.4 Codebaum für den in Beispiel 2.1.6 gegebenen Code



Beispiel 2.1.6 Codebaum I
Gegeben sei der Quellencode

q	$p_Q(q)$	u
$q^{(1)}$	0,1	[0]
$q^{(2)}$	0,2	[10]
$q^{(3)}$	0,3	[110]
$q^{(4)}$	0,4	[111]

Dieser Code kann durch den in Abb. 2.4 gezeigten binären Baum dargestellt werden. Die Entropie pro Quellensymbol berechnet sich zu

$$H(Q) = - \sum_{i=1}^4 p_Q(q^{(i)}) \log_2 p_Q(q^{(i)}) = 1,846 \text{ bit/Quellensymbol} \tag{2.19}$$

und die mittlere Codewortlänge zu

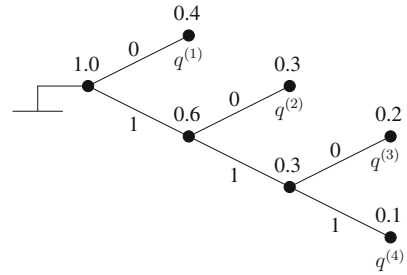
$$E\{W\} = \sum_{i=1}^4 p_Q(q^{(i)}) w^{(i)} = 2,6 \text{ bit/Quellensymbol} . \tag{2.20}$$

Somit ergibt sich eine Effizienz η von nur etwa 71 %. Man beachte, dass die Ungleichung $H(Q) \leq E\{W\} < H(Q) + 1$ erfüllt ist, obwohl der betrachtete Code offenbar nicht optimal sein kann, weil unwahrscheinlichen Quellensymbolen (wie $q^{(1)}$ und $q^{(2)}$) kurze Codewörter zugeordnet werden. \diamond

Beispiel 2.1.7 Codebaum II
Deshalb betrachten wir nun den Quellencode

q	$p_Q(q)$	u
$q^{(1)}$	0,4	[0]
$q^{(2)}$	0,3	[10]
$q^{(3)}$	0,2	[110]
$q^{(4)}$	0,1	[111]

Abb. 2.5 Codebaum für den in Beispiel 2.1.7 gegebenen Code



Dieser Code kann durch den in Abb. 2.5 gezeigten Baum dargestellt werden. Es gilt nach wie vor

$$H(Q) = - \sum_{i=1}^4 p_Q(q^{(i)}) \log_2 p_Q(q^{(i)}) = 1,846 \text{ bit/Quellensymbol}, \quad (2.21)$$

nun aber

$$E\{W\} = \sum_{i=1}^4 p_Q(q^{(i)}) w^{(i)} = 1,9 \text{ bit/Quellensymbol}. \quad (2.22)$$

Somit ergibt sich eine Effizienz η von immerhin etwa 97 %. Dieser Quellencode weist eine wesentlich höhere Effizienz im Vergleich zum Code aus Beispiel 2.1.6 auf, da den wahrscheinlichsten Quellensymbolen kurze Codewörter zugeordnet sind. \diamond

► **Satz 2.1.8 (Massey's Pfadlängen-Satz)** (ohne Beweis) In einem Codebaum ist die mittlere Codewortlänge gleich der Summe der Wahrscheinlichkeiten der inneren Knoten inklusive der Wurzel.

Beispiel 2.1.8

Für den Code aus Beispiel 2.1.6 (Beispiel 2.1.7) erhielten wir auf konventionellem Wege $E\{W\} = \sum_{i=1}^4 p_Q(q^{(i)}) w^{(i)} = 2,6 \text{ bit/Quellensymbol}$ ($1,9 \text{ bit/Quellensymbol}$). Mit Hilfe von Massey's Pfadlängen-Satz kann die mittlere Codewortlänge wesentlich einfacher berechnet werden: $E\{W\} = 1,0 + 0,9 + 0,7 = 2,6 \text{ bit/Quellensymbol}$ ($1,0 + 0,6 + 0,3 = 1,9 \text{ bit/Quellensymbol}$). \diamond

2.1.5 Huffman-Algorithmus

Bislang sind wir weder in der Lage zu entscheiden, ob ein gegebener präfixfreier Quellencode optimal ist oder nicht, noch können wir einen (unter gewissen Randbedingungen) optimalen binären präfixfreien Quellencode systematisch entwerfen. Eine Antwort auf diese offenen Herausforderungen liefert der sog. *Huffman-Algorithmus*. Wir wollen den Huffman-Algorithmus zunächst kennenlernen, bevor wir dessen Optimalität beweisen.

Der Huffman-Algorithmus erzeugt einen optimalen binären präfixfreien Code im Sinne der Entropiecodierung. Dabei wird vorausgesetzt, dass (i) die Wahrscheinlichkeit der Quellsymbole dem Quellencodierer bekannt ist, (ii) die Quellsymbole statistisch unabhängig sind und (iii) die Codierung symbolweise erfolgt. Der Huffman-Algorithmus bildet eine feste Quellenwortlänge auf eine variable Codewortlänge ab, d. h. er erzeugt binäre Codeworte variabler Länge. Übertragungsfehler sind aufgrund der Kommafreiheit nicht erlaubt. Wir werden sehen, dass mit Hilfe des Huffman-Algorithmus die Entropie exakt erreicht werden kann, wenn die Wahrscheinlichkeiten der Quellsymbole Kehrwerte von Zweierpotenzen sind.

Der *Huffman-Algorithmus* besteht aus den folgenden drei Schritten:

- **Schritt 0** (Initialisierung): Jedem der L diskreten Quellsymbole $q^{(i)}$ mit den zugehörigen Wahrscheinlichkeiten $p_Q(q^{(i)})$, $i \in \{1, 2, \dots, L\}$, wird ein Endknoten zugeordnet. Alle Endknoten werden als „aktiv“ erklärt.
- **Schritt 1**: Die beiden unwahrscheinlichsten „aktiven“ Knoten werden vereinigt. Diese beiden Knoten werden als „passiv“ erklärt, der neu geschaffene Knoten wird als „aktiv“ erklärt. Dem neuen „aktiven“ Knoten wird die Summe der Wahrscheinlichkeiten der beiden verbundenen Knoten zugewiesen.
- **Schritt 2** (Abbruchkriterium): Man stoppt, falls nur noch ein „aktiver“ Knoten übrig bleibt. In diesem Fall ist die Wurzel erreicht, d. h. die Summe der Wahrscheinlichkeiten ist gleich eins. Andernfalls fährt man mit Schritt 1 fort.

Beispiel 2.1.9 Huffman-Algorithmus I

Als Anwendung für eine Huffman-Codierung betrachten wir zunächst das in Beispiel 1.3.4 eingeführte Pferderennen. Die acht Pferde haben folgende Gewinnwahrscheinlichkeiten:

q	$q^{(1)}$	$q^{(2)}$	$q^{(3)}$	$q^{(4)}$	$q^{(5)}$	$q^{(6)}$	$q^{(7)}$	$q^{(8)}$
$p_Q(q)$	1/2	1/4	1/8	1/16	1/64	1/64	1/64	1/64

Gesucht ist ein optimaler binärer präfixfreier Code.

Lösung: Wendet man den Huffman-Algorithmus schrittweise an, so erhält man den in Abb. 2.6 dargestellten Baum, wobei vereinbarungsgemäß nach oben abgehende Zweige mit 0 und nach unten abgehende Zweige mit 1 adressiert sind. Die Entropie pro Quellsymbol berechnet sich zu $H(Q) = 2$ bit, die mittlere Codewortlänge ist ebenfalls gleich $E\{W\} = 2$ bit/Quellsymbol. Folglich beträgt in diesem Beispiel die Effizienz $\eta = 100\%$. Dies erklärt sich durch die spezielle Wahl der angenommenen Gewinnwahrscheinlichkeiten: Diese sind im vorliegenden Beispiel Kehrwerte von Zweierpotenzen.

◇

Obwohl der Huffman-Algorithmus einen optimalen präfixfreien Code generiert, ist die Effizienz meist kleiner als $\eta = 100\%$. Dies sei an folgendem Beispiel illustriert:

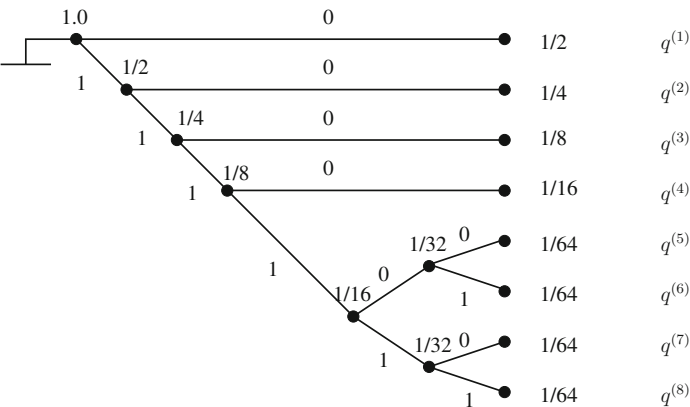


Abb. 2.6 Huffman-Code für Beispiel 2.1.9

Beispiel 2.1.10 Huffman-Algorithmus II

Entwerfe einen Huffman-Code für vier Quellsymbole mit folgenden Wahrscheinlichkeiten:

q	$q^{(1)}$	$q^{(2)}$	$q^{(3)}$	$q^{(4)}$
$p_Q(q)$	0,4	0,3	0,2	0,1

Lösung: Die schrittweise Vorgehensweise ist in Abb. 2.7 von rechts nach links dargestellt. Der optimale präfixfreie Code und somit die Lösung lautet:

q	$q^{(1)}$	$q^{(2)}$	$q^{(3)}$	$q^{(4)}$
u	[0]	[10]	[110]	[111]

Es stellt sich heraus, dass der Huffman-Code mit dem Code aus Beispiel 2.1.7 identisch ist. Die mittlere Codewortlänge beträgt $E\{W\} = 1,9$ bit/Quellsymbol, die Effizienz $\eta = 97\%$. ◇

Ist die Effizienz kleiner als 100 %, so kann diese normalerweise verbessert werden, indem mehrere Quellsymbole zu einem Supersymbol zusammengefasst werden.

Beispiel 2.1.11 Huffman-Algorithmus III

Fasst man zwei Quellsymbole zu einem Supersymbol zusammen, so ergibt sich für die in Beispiel 2.1.10 gegebenen Parameter der in Abb. 2.8 gezeigte Huffman-Code. Man beachte, dass die mittlere Codewortlänge nun von $E\{W\} = 1,9$ auf $E\{W\} = 1,865$ gesunken ist. Dies entspricht einer Effizienz von immerhin $\eta = 99\%$. Fasst man noch mehr Quellsymbole zusammen, so kann asymptotisch eine Effizienz von $\eta = 100\%$ erreicht werden. ◇

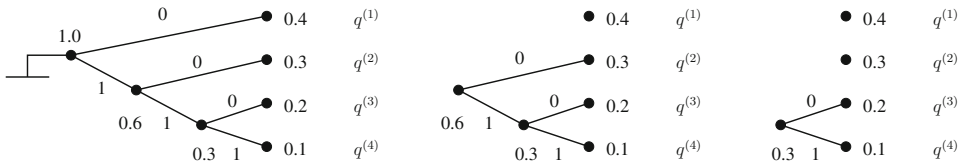


Abb. 2.7 Huffman-Code zu Beispiel 2.1.10

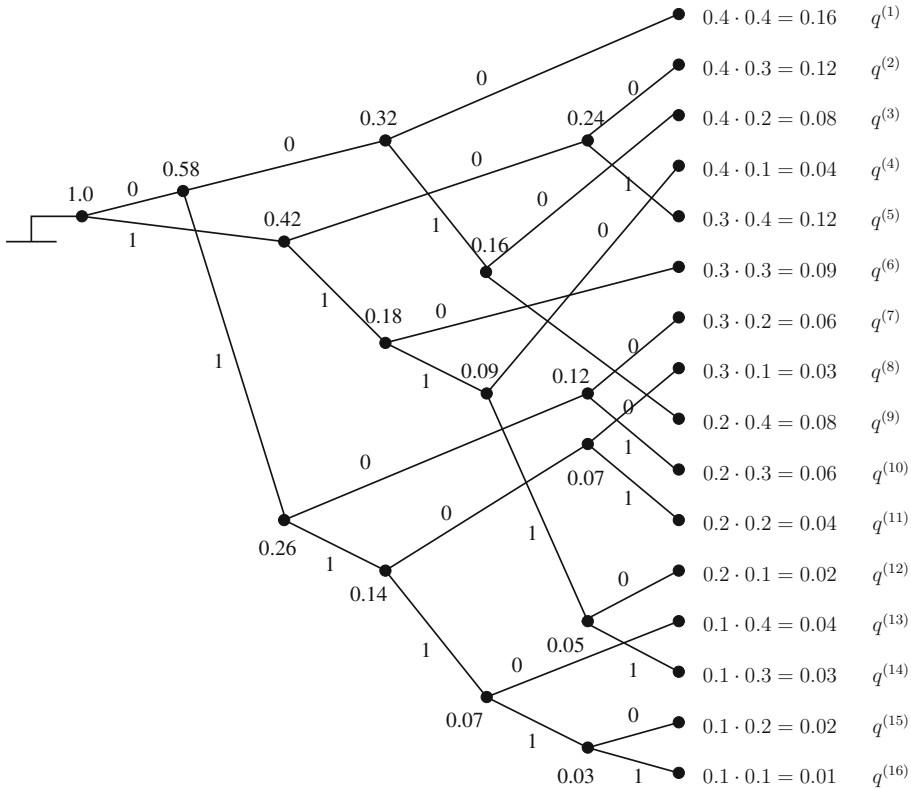


Abb. 2.8 Huffman-Code zu Beispiel 2.1.11

Bislang wurden statistisch unabhängige Quellsymbole angenommen. Sind die Quellsymbole statistisch abhängig (wie z. B. in Texten), so können durch Verwendung von Supersymbolen die statistischen Bindungen berücksichtigt werden. Wie wir in Abschn. 2.2.1 lernen werden, ist bei gedächtnisbehafteten Quellen nicht nur die Entropie erreichbar, sondern die sog. Entropierate, die typischerweise kleiner als die Entropie ist. Allerdings wächst bei Verwendung des Huffman-Algorithmus die Komplexität erheblich und es muss die Wahrscheinlichkeit aller Supersymbole bekannt sein. Alternativen werden in Abschn. 2.2.1 aufgezeigt.

Für den interessierten Leser wollen wir nun die Optimalität der Huffman-Codierung beweisen. Hierzu wird der Begriff des *abgeleiteten Codes* benötigt:

► **Definition 2.1.4 (Abgeleiteter Code)** Sei $\{u\}$ ein binärer präfixfreier Code. Man nennt den Code $\{u'\}$ mit $u'^{(1)} = u^{(1)}, u'^{(2)} = u^{(2)}, \dots, u'^{(L-2)} = u^{(L-2)}, u'^{(L-1)}$ einen abgeleiteten Code, wenn $u^{(L-1)} = u'^{(L-1)}\|0$ und $u^{(L)} = u'^{(L-1)}\|1$ sowie

$$p_{Q'}(q'^{(i)}) = \begin{cases} p_Q(q^{(i)}) & \text{für } i = 1, 2, \dots, L-2 \\ p_Q(q^{(L-1)}) + p_Q(q^{(L)}) & \text{für } i = L-1. \end{cases} \quad (2.23)$$

Hierbei bezeichnet „ $\|$ “ eine Verkettung von Zeichenketten. Durch Anwendung von Masseys Pfadlängen-Satz erhält man:

$$E\{W\} = E\{W'\} + p_Q(q^{(L-1)}) + p_Q(q^{(L)}). \quad (2.24)$$

Somit ist $E\{W\}$ genau dann minimal, wenn auch $E\{W'\}$ minimal ist. Wir haben folglich bewiesen, dass der binäre präfixfreie Code mit $u^{(L-1)} = u'^{(L-1)}\|0$ und $u^{(L)} = u'^{(L-1)}\|1$ genau dann optimal ist, wenn der abgeleitete Code optimal ist. Der Huffman-Algorithmus folgt durch Induktion.

Sind alle Wahrscheinlichkeiten der Quellsymbole Kehrwerte von Zweierpotenzen, wie in Beispiel 2.1.9, so gilt (bei abfallender Sortierung) $p_Q(q^{(L-1)}) = p_Q(q^{(L)})$. Folglich ist $p_Q(q^{(L-1)}) + p_Q(q^{(L)})$ ebenfalls eine Zweierpotenz. Entsprechendes gilt für alle anderen Baumtiefen. Für alle anderen Wahrscheinlichkeitsverteilungen trifft letzteres nicht zu. Folglich entspricht die mittlere Codewortlänge $E\{W\}$ nur in diesem Spezialfall exakt der Entropie $H(Q)$.

Der Huffman-Algorithmus erfordert, wie erwähnt, die Kenntnis der Quellenstatistik. Wird diese vor der Codierung geschätzt, spricht man von einem *adaptiven Huffman-Algorithmus*. Es bezeichne $p_Q(q^{(i)})$ die wahre Wahrscheinlichkeitsfunktion und $q_Q(q^{(i)})$ die im Codierer angenommene Wahrscheinlichkeitsfunktion, $i \in \{1, \dots, L\}$. Stimmt die angenommene Quellenstatistik $q_Q(q^{(i)})$ nicht mit der wahren Quellenstatistik $p_Q(q^{(i)})$ überein, so ist der Huffman-Algorithmus naturgemäß nicht mehr optimal. Gemäß Beispiel 1.3.5 werden im Mittel mindestens $H(Q) + D(p \parallel q)$ bit/Quellsymbol für eine verlustlose Quellencodierung benötigt, wobei es sich bei $D(p \parallel q)$ um die Kullback-Leibler-Distanz handelt.

2.1.6 Arithmetische Quellencodierung

Obwohl der Huffman-Algorithmus optimal im Sinne einer präfixfreien Quellencodierung ist, gibt es Situationen, in denen dessen Effizienz gering ist. Ein einfaches Beispiel ist wie folgt:

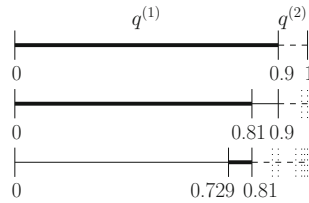


Abb. 2.9 Prinzipielle Funktionsweise der arithmetischen Quellencodierung am Beispiel einer binären Quelle ($L = 2$, $p_Q(q^{(1)}) = 0,9$, $p_Q(q^{(2)}) = 0,1$). Die hervorgehobenen Intervalle repräsentieren die Quellensymbolsequenz $[q^{(1)}, q^{(1)}, q^{(2)}]$

Beispiel 2.1.12 Trickmünze

Gegeben sei eine binäre Zufallsvariable Q mit $p_Q(q^{(1)}) = 0,9$ und $p_Q(q^{(2)}) = 0,1$. Wendet man den Huffman-Algorithmus an, so beträgt die Codewortlänge für beide Ereignisse $W = 1$, folglich gilt auch $E\{W\} = 1$. Da $H(Q) < 0,5$ (vgl. Abb. 1.3), ist die Effizienz $\eta < 50\%$.

Der ungünstigste Fall tritt auf, wenn ein Ereignis fast immer auftritt, das andere Ereignis fast nie. Obwohl die Information der Quelle dann gegen null geht, muss bei der Huffman-Codierung für $L = 2$ ein Codebit pro Quellensymbol generiert werden. \diamond

Bei der arithmetischen Quellencodierung wird die Randbedingung, dass ein Quellensymbol in eine ganzzahlige Anzahl an Codebits abgebildet wird, aufgegeben. Dadurch kann die Codierung für fast alle gedächtnisfreien Quellen effizienter durchgeführt werden, allerdings auf Kosten der Präfixfreiheit.

Die prinzipielle Funktionsweise der arithmetischen Quellencodierung ist in Abb. 2.9 für obiges Beispiel der Trickmünze skizziert. Man unterteilt das Intervall $[0, 1]$ in L Teilintervalle, wobei das i -te Teilintervall eine Länge $p_Q(q^{(i)})$ aufweist und dem Quellensymbol $q^{(i)}$ zugeordnet wird, $i \in \{1, 2, \dots, L\}$. Man selektiert das Teilintervall, welches dem ersten Quellensymbol zugeordnet ist. Dieses Teilintervall (im vorliegenden Beispiel das erste) wird anschließend erneut in L Teilintervalle aufgeteilt, wobei deren Längen proportional zu $p_Q(q^{(i)})$ gewählt werden. Nun selektiert man das Teilintervall, welches dem zweiten Quellensymbol zugeordnet ist. Dieses Teilintervall (im vorliegenden Beispiel wiederum das erste) wird anschließend erneut in L Teilintervalle aufgeteilt, wobei deren Längen proportional zu $p_Q(q^{(i)})$ gewählt werden. Nun selektiert man das Teilintervall, welches dem dritten Quellensymbol zugeordnet ist. Im vorliegenden Beispiel ist es das zweite Intervall. Nachdem alle Quellensymbole verarbeitet wurden, wird das zuletzt selektierte Intervall binär codiert. Hierzu wird die kürzeste Binärzahl gesucht, die innerhalb der Intervallgrenzen (hier $[0,729, 0,81]$) liegt. Im betrachteten Beispiel ist dies die Binärzahl $.11$, denn $0,5 + 0,25 = 0,75$ liegt im ausgewählten Intervall. Der Quellencodierer gibt folglich die Bitsequenz $[11]$ aus. Dabei wird die in Tab. 2.4 definierte Arithmetik verwendet:

Tab. 2.4 Binärdarstellung von Zweierpotenzen von $1/2$

Wert	Binärdarstellung
0,5	.1
0,25	.01
0,125	.001
0,0625	.0001
...	...

Würde dreimal hintereinander das Quellensymbol $q^{(1)}$ gesendet, so würde in allen drei Schritten das erste Intervall ausgewählt werden. Dieses kann durch ein einziges Codebit 0 adressiert werden. Somit könnten drei Quellensymbole durch ein einziges Codebit repräsentiert werden. Dies zeigt einerseits den Hauptvorteil gegenüber dem Huffman-Algorithmus auf, weist andererseits auf ein Problem hin: Man erkennt aus dem Codewort nicht, wie viele Quellensymbole verarbeitet wurden. Es hätten beispielsweise auch vier oder fünf Quellensymbole sein können. Dieses Problem kann in der Praxis gelöst werden, indem man hinter dem letzten Quellensymbol ein Stoppsymbol anfügt, welches gesondert codiert wird. Hierdurch sinkt allerdings die Effizienz der Quellencodierung. Ein numerisches Problem besteht darin, dass die Intervalle mit zunehmender Anzahl an Quellensymbolen immer kürzer werden. Dieser Herausforderung kann man in der Praxis durch eine ganzzahlige Arithmetik begegnen. Interessanterweise können zeitvariante Quellen nahezu trivial berücksichtigt werden, indem in jedem Schritt die Anzahl der Intervalle und deren Längen an die Wahrscheinlichkeiten der jeweiligen Quellensymbole angepasst werden.

2.2 Gedächtnisbehaftete Quellen

2.2.1 Markoff-Quellen

Bislang wurden statistisch unabhängige Quellensymbole (d. h. gedächtnisfreie Quellen) vorausgesetzt. In der Praxis hat man es aber meist mit gedächtnisbehafteten Quellen zu tun. Bei statistisch abhängigen Quellensymbolen kann eine Quellencodierung wesentlich effizienter sein.

Wir betrachten nun das in Abb. 2.10 gezeigte Szenario. Die diskrete Quelle erzeuge, wie in Abschn. 2.1.4, eine Sequenz von n Zufallsvariablen Q_1, Q_2, \dots, Q_n . Die Sequenz der n Quellensymbole werde mit q_1, q_2, \dots, q_n bezeichnet, die entsprechenden Entropien mit $H(Q_1), H(Q_2), \dots, H(Q_n)$. Wir nehmen an, dass alle Quellensymbole über dem gleichen endlichen Alphabet $\mathcal{Q} = \{q^{(1)}, q^{(2)}, \dots, q^{(L)}\}$ der Mächtigkeit L definiert sind. Im Unterschied zu Abschn. 2.1.4 können wir uns nicht mehr auf ein einziges Quellensymbol beschränken. Der Quellencodierer sei erneut verlustlos. Er erzeuge pro Quellensymbol q_k ein Codewort $u_k, k \in \{1, 2, \dots, n\}$. Wir beschränken uns weiterhin auf binäre präfixfreie Codewörter variabler Länge.

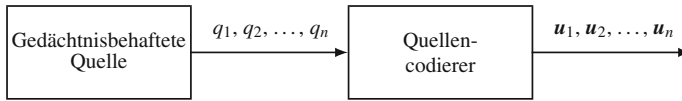


Abb. 2.10 Diskrete gedächtnisbehaftete Quelle mit nachgeschalteter Quellencodierung

C. E. Shannon hat in seiner bahnbrechenden Arbeit aus dem Jahre 1948 ein didaktisch hervorragendes Experiment veröffentlicht, um die statistischen Bindungen zwischen Buchstaben in englischsprachigen Texten zu verdeutlichen. Dazu hat er die 26 Großbuchstaben und das Leerzeichen, also ein Alphabet bestehend aus $L = 27$ Quellensymbolen (Zeichen) betrachtet.

Im ersten Schritt hat er die Häufigkeitsverteilung der 27 betrachteten Quellensymbole durch eine Analyse englischsprachiger Texte ermittelt. Auf Basis dieser Statistik generierte ein Zufallsgenerator die folgende Quellensymbolsequenz:

OCRO HLI RGWR NMIELWIS EU LL NBNESEBYA
TH EEI ALHENHTTPA OOBTTVA NAH BRL

Man beachte, dass in diesem ersten Experiment die Quellensymbole statistisch unabhängig generiert wurden.

In einem zweiten Schritt hat Shannon jeweils zwei Quellensymbole zu einem Supersymbol zusammengefasst und die Häufigkeitsverteilung der 27^2 möglichen Supersymbole durch eine Textanalyse ermittelt. Auf Basis dieser Statistik generierte ein Zufallsgenerator die folgende Quellensymbolsequenz:

ON IE ANTSOUTINYS ARE T INCTORE ST BE S
DEAMY ACHIN D ILONASIVE TUCOOWE AT
TEASONARE FUSO TIZIN ANDY TOBE SEACE CTISBE

In einem dritten Schritt schließlich wurden jeweils drei Quellensymbole zu einem Supersymbol zusammengefasst, und die Häufigkeitsverteilung der 27^3 möglichen Supersymbole durch eine Textanalyse ermittelt. Auf Basis dieser Statistik generierte ein Zufallsgenerator die folgende Quellensymbolsequenz:

IN NO IST LAT WHEY CRATICT FROURE BIRS
GROCID PONDEOME OF DEMONSTRURES OF THE
REPTAGIN IS REGOACTIONA OF CRE

Die Ähnlichkeit mit englischen Texten ist verblüffend, auch wenn der Zufallsprozess keine inhaltlich Bedeutung besitzt. Für andere Sprachen sind vergleichbare Ergebnisse zu erzielen, ebenso für Wörter anstelle von Buchstaben.

Diskrete Zufallsprozesse kann man beispielsweise durch eine *Markoff-Quelle* modellieren. Eine Markoff-Quelle entsteht aus einer Markoff-Kette, indem jedem Zustand ein Quellensymbol zugeordnet wird.

► **Definition 2.2.1 (Markoff-Kette und Markoff-Quelle)** Eine *Markoff-Kette* 1. Ordnung ist gekennzeichnet durch eine endliche Zustandsmenge $\mathcal{Z} = \{z^{(1)}, z^{(2)}, \dots, z^{(L_z)}\}$ der Mächtigkeit $L_z = |\mathcal{Z}|$ sowie einer Übergangsmatrix $\mathbf{P} = [p_{i,j}]$. Die Elemente der Übergangsmatrix, $p_{i,j} := P(Z_k = z^{(j)} | Z_{k-1} = z^{(i)}) \in [0, 1]$, beschreiben die Wahrscheinlichkeit für einen Übergang von einem Zustand i (zum Zeitpunkt $k-1$) zu einem Folgezustand j (zum Zeitpunkt k), wobei $i, j \in \{1, 2, \dots, L_z\}$. Definitionsgemäß muss die Summe aller Wahrscheinlichkeiten, die von einem Zustand i ausgehen, eins ergeben: $\sum_j p_{i,j} = 1 \forall i$. Falls $p_{i,j} = 0$, so existiert der entsprechende Übergang nicht. Es bezeichne $[Z_1, Z_2, \dots, Z_k, \dots, Z_n]$ eine Sequenz von Zuständen, wobei $Z_k \in \mathcal{Z}$, $k \in \{1, 2, \dots, n\}$. Per Definition hängt bei einer Markoff-Kette 1. Ordnung die Wahrscheinlichkeit für einen Zustand Z_k nur vom vorherigen Zustand Z_{k-1} ab:

$$P(Z_k | Z_1, Z_2, \dots, Z_{k-1}) = P(Z_k | Z_{k-1}). \quad (2.25)$$

Eine *Markoff-Quelle* ist eine Sequenz $[Q_1, Q_2, \dots, Q_n]$ von Zufallsvariablen, die dadurch gekennzeichnet ist, dass jedem Zustand $z^{(1)}, z^{(2)}, \dots, z^{(L_z)}$ ein Quellensymbol $q^{(1)}, q^{(2)}, \dots, q^{(L_z)}$ zugeordnet wird. Jede Zufallsvariable Q_k nimmt Werte aus einer endlichen Menge \mathcal{Q} der Mächtigkeit $L = |\mathcal{Q}|$ an, wobei $L \leq L_z$. Die Menge \mathcal{Q} nennt man Symbolalphabet.

Man nennt eine Markoff-Kette *homogen*, wenn die Übergangswahrscheinlichkeiten $p_{i,j} = P(Z_k = z^{(j)} | Z_{k-1} = z^{(i)})$ unabhängig vom Zeitindex k sind, $i, j \in \{1, 2, \dots, L_z\}$. Man nennt sie *stationär*, wenn die Zustandswahrscheinlichkeiten $p^{(i)} := P(Z_k = z^{(i)})$ unabhängig vom Zeitindex k sind. Die stationäre Zustandsverteilung $\mathbf{p} := [p^{(1)}, p^{(2)}, \dots, p^{(L_z)}]$ ergibt sich aus der gegebenen Übergangsmatrix \mathbf{P} durch Lösung des Gleichungssystems $\mathbf{p} = \mathbf{p} \cdot \mathbf{P}$ in Verbindung mit der Randbedingung $\sum_i p^{(i)} = 1$. Kann man von jedem Zustand nach endlich vielen Zwischenschritten jeden anderen Zustand erreichen, so spricht man von einer *irreduziblen* Markoff-Kette. Irreduzible homogene Markoff-Quellen sind *ergodisch*.

Beispiel 2.2.1 Markoff-Kette und Markoff-Quelle

Gegeben sei die in Abb. 2.11 skizzierte Markoff-Kette mit der Übergangsmatrix \mathbf{P} . Die Markoff-Kette besteht aus den $L_z = 4$ Zuständen $z^{(1)}, z^{(2)}, z^{(3)}, z^{(4)}$. Durch die Zuordnung von $L \leq 4$ Quellensymbolen erhält man eine Markoff-Quelle, beispielsweise gelte $q^{(1)} = A$, $q^{(2)} = B$, $q^{(3)} = C$ und $q^{(4)} = \text{Leerzeichen}$. Startet man von einem beliebigen Anfangszustand gemäß der stationären Zustandsverteilung, so wird durch ein Durchlaufen der Zustände eine stationäre, ergodische Zufallssequenz generiert. ◇

Die entscheidende Frage lautet nun: Wie viele Bits pro Quellensymbol sind mindestens notwendig, um eine gedächtnisbehaftete, stationäre Quellensymbolsequenz zu codieren? Wir wissen, dass die Verbundentropie $H(\mathbf{Q}) := H(Q_1, Q_2, \dots, Q_n)$ durch die Kettenregel der Entropie wie folgt exakt beschrieben werden kann:

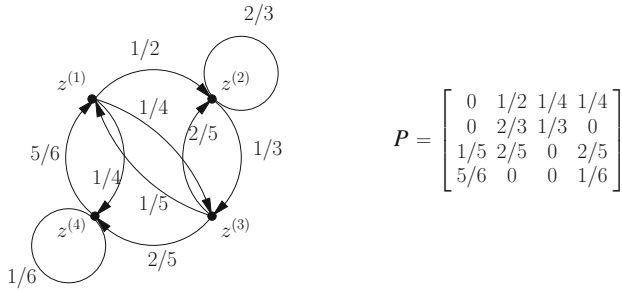


Abb. 2.11 Zustandsdiagramm einer Markoff-Kette

$$H(Q_1, Q_2, \dots, Q_n) = H(Q_1) + H(Q_2|Q_1) + H(Q_3|Q_1, Q_2) + \dots + H(Q_n|Q_1, Q_2, \dots, Q_{n-1}). \quad (2.26)$$

$H(Q)$ ist bekanntlich die Unsicherheit der gesamten Quellensymbolsequenz Q , unabhängig ob die Quelle gedächtnisfrei oder gedächtnisbehaftet ist. Bei gedächtnisbehafteten Quellen spielt es keine Rolle, ob eine Markoff-Quelle vorliegt oder nicht. Die Unsicherheit ist gemäß Shannons Quellencodierungstheorem eine untere Schranke für die minimale Anzahl an Bits, die benötigt werden, um die Quellensymbolsequenz verlustlos zu repräsentieren.

Es sei

$$H_n(Q) := H(Q_n|Q_1, Q_2, \dots, Q_{n-1}). \quad (2.27)$$

$H_n(Q)$ ist die Entropie für das Quellensymbol Q_n gegeben dessen Vergangenheit. Wie in (1.46) gezeigt, gilt

$$H_n(Q) \leq H(Q_n). \quad (2.28)$$

Beispiel 2.2.2

Wir betrachten erneut einen englischsprachigen Text bestehend aus 26 Großbuchstaben und dem Leerzeichen, d.h. $L = 27$. Gegeben sei die Quellensymbolsequenz $[Q_1, Q_2, \dots, Q_7] = \text{„INFORMA“}$. Gesucht ist Q_8 . Offenbar kommen für Q_8 nur wenige Symbole in Betracht wie z.B. „T“, „L“ oder „N“. Kennt man die Quellensymbolsequenz $[Q_1, Q_2, \dots, Q_7]$ hingegen nicht, so ist die Unsicherheit über Q_8 wesentlich größer, denn es kommen alle L Zeichen in Betracht. In diesem Beispiel gilt also offenbar $H(Q_8|Q_1, Q_2, \dots, Q_7) \ll H(Q_8)$. \diamond

Betrachtet man unendlich lange Sequenzen, so erhält man aus (2.27) die sog. *Entropierate*:

► **Definition 2.2.2 (Entropierate)** Die Entropierate der Sequenz $[Q_1, \dots, Q_n]$ ist gemäß

$$H_\infty(Q) := \lim_{n \rightarrow \infty} H(Q_n|Q_1, Q_2, \dots, Q_{n-1}) = \lim_{n \rightarrow \infty} H_n(Q) \quad (2.29)$$

definiert.

Die Entropierate ist nichtnegativ. Sie repräsentiert die Unsicherheit eines Symbols gegeben dessen vollständige Vergangenheit. Man kann für stationäre Quellen zeigen, dass

$$H_\infty(Q) = \lim_{n \rightarrow \infty} \frac{1}{n} H(Q_1, Q_2, \dots, Q_n) = \lim_{n \rightarrow \infty} \frac{1}{n} H(Q). \quad (2.30)$$

Verlustlose Verfahren zur Quellencodierung können im günstigsten Fall die Entropierate erreichen. Der Ausdruck $H(Q_1, Q_2, \dots, Q_n)/n$ wird *Nachrichtenrate* genannt. Für stationäre gedächtnisfreie Sequenzen ist die Nachrichtenrate mit der Entropie pro Quellensymbol, $H(Q)$, identisch.

Ein weiterer Spezialfall ist die sog. *Alphabetrate*:

► **Definition 2.2.3 (Alphabetrate)** Die Alphabetrate einer Zufallsvariable Q ist gleich dem Logarithmus der Mächtigkeit des Symbolalphabets:

$$H_0(Q) := \log |\mathcal{Q}| = \log L. \quad (2.31)$$

Die Alphabetrate entspricht der maximal möglichen Unsicherheit pro Quellensymbol. Um die Alphabetrate zu erreichen, ist keine Datenkompression notwendig. Insgesamt folgt aus (2.27), (2.29) und (2.31):

$$\begin{aligned} H_0(Q) &= \log L \\ H_1(Q) &= H(Q_1) \\ H_2(Q) &= H(Q_2|Q_1) \\ H_3(Q) &= H(Q_3|Q_1, Q_2) \\ &\dots \\ H_{n-1}(Q) &= H(Q_{n-1}|Q_1, Q_2, \dots, Q_{n-2}) \\ H_n(Q) &= H(Q_n|Q_1, Q_2, \dots, Q_{n-1}) \\ H_\infty(Q) &= \lim_{n \rightarrow \infty} H(Q_n|Q_1, Q_2, \dots, Q_{n-1}) \end{aligned} \quad (2.32)$$

Aus (1.47) folgt unmittelbar

$$H(Q_n|Q_1, Q_2, \dots, Q_{n-1}) \leq H(Q_n|Q_2, Q_3, \dots, Q_{n-1}). \quad (2.33)$$

Für stationäre Quellen kann die rechte Seite auch wie folgt geschrieben werden:

$$H(Q_n|Q_2, Q_3, \dots, Q_{n-1}) = H(Q_{n-1}|Q_1, Q_2, \dots, Q_{n-2}), \quad (2.34)$$

d. h. insgesamt

$$H(Q_n|Q_1, Q_2, \dots, Q_{n-1}) \leq H(Q_{n-1}|Q_1, Q_2, \dots, Q_{n-2}), \quad (2.35)$$

oder kürzer

$$H_n(Q) \leq H_{n-1}(Q). \quad (2.36)$$

Dies beweist folgenden wichtigen Satz:

► **Satz 2.2.1** Für stationäre Quellen gilt

$$H_\infty(Q) \leq H_n(Q) \leq H_{n-1}(Q) \leq \dots \leq H_1(Q) \leq H_0(Q). \quad (2.37)$$

Bei gedächtnisfreien Quellen gilt (2.37) mit Gleichheit, bei gedächtnisbehafteten Quellen mit Ungleichheit.

Dieser Satz besagt, dass sich bei gedächtnisbehafteten Quellen mit zunehmender Berücksichtigung der Vergangenheit die Unsicherheit verringert. Da gemäß Shannons Quellencodiertheorem die mittlere Wortlänge bei verlustloser Quellencodierung durch die Entropie nach unten begrenzt ist, können bei gedächtnisbehafteten Quellen kürzere Wortlängen erreicht werden.

Beispiel 2.2.3 Entropie englischsprachiger Texte

Bei englischsprachigen Texten mit $L = 27$ Zeichen (26 Großbuchstaben plus Leerzeichen) gilt: $H_0(Q) = \log_2(27)$ bit/Zeichen $\approx 4,75$ bit/Zeichen, $H(Q) \approx 4,1$ bit/Zeichen und $H_\infty(Q) \approx 1,2$ bit/Zeichen. (Die Zahlenwerte für $H(Q)$ und $H_\infty(Q)$ hängen vom Textmaterial ab und variieren somit leicht in verschiedenen Publikationen. Der Zahlenwert von $H(Q) \approx 4,1$ bit/Zeichen wurde 1951 von Shannon angegeben. Der Zahlenwert von $H_\infty(Q) \approx 1,2$ bit/Zeichen kann aus einer von ihm angegebenen unteren Schranke abgelesen werden [Sha51].)

Dieses Beispiel verdeutlicht, dass durch eine Huffman-Codierung Texte (nur) um den Faktor $H_0(Q)/H(Q) \approx 1,16$ reduziert werden können, da die Huffman-Codierung statistische Bindungen ignoriert. Wenn die statistischen Bindungen vollständig genutzt würden, so wäre eine Reduktion (immerhin) um den Faktor $H_0(Q)/H_\infty(Q) \approx 3,95$ möglich. Bei englischsprachigen Texten (ohne Bilder) kann die Datenmenge somit theoretisch ohne Informationsverlust auf etwa ein Viertel komprimiert werden.

K. Küpfmüller hat drei Jahre später die Entropie deutschsprachiger Texte mit einer anderen Methode untersucht [Kue54]. Für das gleiche Symbolalphabet bestehend aus 27 Großbuchstaben einschließlich des Leerzeichens hat er eine Entropierate von $H_\infty(Q) \approx 1,3$ angegeben. Dieser Wert ist nur unwesentlich höher als der von Shannon ermittelte Wert. ◇

► **Definition 2.2.4 (Redundanz)** Gegeben sei eine gedächtnisbehaftete Quellensymbolsequenz \mathbf{Q} der Länge n . Die Differenz zwischen der Alphabetrate, $H_0(Q)$, und der Nachrichtenrate, $H(\mathbf{Q})/n$, wird *Redundanz* genannt:

$$D := H_0(Q) - H(\mathbf{Q})/n. \quad (2.38)$$

Die Redundanz gibt an, in welchem Maße eine gedächtnisbehaftete Quellensymbolsequenz der Länge n (im Mittel) ohne Informationsverlust komprimiert werden kann. Für gedächtnisfreie stationäre Quellensymbolsequenzen gilt folglich $D = H_0(Q) - H(Q)$. Für den Grenzübergang $n \rightarrow \infty$ definiert man entsprechend

$$D_\infty := H_0(Q) - H_\infty(Q). \quad (2.39)$$

Eine lange gedächtnisbehaftete Quellensymbolsequenz kann somit (im Mittel) maximal um den Wert D_∞ ohne Informationsverlust komprimiert werden.

► **Bemerkung 2.2.1** Häufig wird die Redundanz in normierter Darstellung angegeben. Analog zu (2.38) und (2.39) definiert man

$$\tilde{D} := (H_0(Q) - H(Q)/n)/H_0(Q) = 1 - H(Q)/nH_0(Q) \quad (2.40)$$

und

$$\tilde{D}_\infty := (H_0(Q) - H_\infty(Q))/H_0(Q) = 1 - H_\infty(Q)/H_0(Q). \quad (2.41)$$

Beispiel 2.2.4 Entropie englischsprachiger Texte, Fortsetzung

Für die in Beispiel 2.2.3 genannten Zahlenwerte ergibt sich $D_\infty \approx 4,75 - 1,2 = 3,55$ bit/Zeichen. Die Redundanz natürlicher Sprachen ist somit beträchtlich. ◇

Ein weiteres Beispiel möge diese Aussage kräftigen:

Beispiel 2.2.5 Redundanz in deutschsprachigen Texten

Solange der erste und der letzte Buchstabe eines Wortes an der richtigen Stelle sind, kann man einen Text lesen, auch wenn die anderen Buchstaben vertauscht sind:

Sgonlae der estre und der lzette Butsahcbe enies Wteors an der rechtiugn Sletle snid, knan man eenin Txet leesn, auch wnen die aendern Basbuhcten vsuchratet snid.

Ähnliches gilt, wenn einige Vokale gestrichen werden. ◇

2.2.2 Willems-Algorithmus

Beim Huffman-Algorithmus muss die Häufigkeitsverteilung der Quellensymbole bekannt sein oder geschätzt werden. Wird eine falsche Quellenstatistik angenommen, so führt die Huffman-Codierung zu einer geringeren Effizienz. Die Huffman-Codierung ist sensitiv gegenüber der angenommenen Verteilung, außerdem ist sie für gedächtnisfreie Quellen konzipiert. Gleiches gilt für die arithmetische Codierung.

► **Definition 2.2.5 (Universelle Quellencodierung)** Wir nennen einen Algorithmus zur Quellencodierung universell, falls die Quellenstatistik vor der Codierung nicht bekannt sein muss.

Eine wichtige Frage ist nun: Können universelle Quellencodierverfahren die Entropie $H(Q)$ oder sogar die Entropierate $H_\infty(Q)$ erreichen? Als Beispiel eines universellen Quellencodierverfahrens wollen wir nun den Willems-Algorithmus studieren. Dieser bildet eine feste Quellenwortlänge auf eine variable Codewortlänge ab.

Der Willems-Algorithmus kann auf beliebig lange Quellensymbolsequenzen mit endlichem Alphabet angewendet werden. Die Quellensymbolsequenz wird in Teilsequenzen konstanter Länge N unterteilt. Für jede Teilsequenz wird die Zeit bestimmt, wann die Teilsequenz zuletzt aufgetreten war. Die *Wiederholungszeit* t_r ist ein Maß für die Auftrittshäufigkeit der Teilsequenz. Das wesentliche Prinzip des Willems-Algorithmus besteht darin, kleinen Wiederholungszeiten t_r eine kurze Codewortlänge zuzuordnen. Die wesentlichen Schritte sind wie folgt:

- **Schritt 0** (Initialisierung): Es werden, wie nachfolgend beschrieben, eine Codierungstabelle und ein Puffer der Länge $2^N - 1$ angelegt. Der Puffer wird mit einer beliebigen Quellensymbolsequenz initialisiert, vorzugsweise mit einer wahrscheinlichen Quellensymbolsequenz. Der initiale Pufferinhalt muss dem Quellendecodierer bekannt sein.
- **Schritt 1**: Die Quellensymbolsequenz q der Länge n wird in $m = n/N$ Teilsequenzen q_1, q_2, \dots, q_m der Länge N unterteilt, wobei $q = [q_1, q_2, \dots, q_m]$.
- **Schritt 2**: Diese Teilsequenzen werden nacheinander von rechts in den Puffer der Länge $2^N - 1$ geschoben. Der Codierer bestimmt die Wiederholungszeit t_r der aktuellen Teilsequenz. Ist diese kleiner gleich der Pufferlänge (d. h. die Teilsequenz ist im Puffer enthalten), so liest man aus der Codierungstabelle ein Codewort aus, welches die Wiederholungszeit repräsentiert. Andernfalls wird die Teilsequenz mit einem Präfix versehen und zusammen mit diesem Präfix ausgegeben.
- **Schritt 3** (Abbruchkriterium): Der Pufferinhalt wird um N Schritte nach links verschoben. Man hört auf, wenn das Sequenzende erreicht ist. Sonst fährt man mit Schritt 2 fort.

Ein geeigneter Entwurfparameter ist $N = 2^p - 1$, wobei $p \geq 2$.

Die *Codierungstabelle* wird für ein gegebenes N wie folgt generiert: Es seien i und j zwei ganze Zahlen aus dem Bereich $0 \leq i, j \leq N$. Zu jeder Wiederholungszeit $t_r \in \{0, 1, \dots, 2^N - 1\}$ wird genau ein Indexpaar (i, j) bestimmt, wobei

$$\begin{aligned} i &= \lfloor \log_2(t_r) \rfloor \\ j &= t_r - 2^i. \end{aligned} \tag{2.42}$$

(Die Schreibweise $\lfloor \cdot \rfloor$ steht für die nächst kleinere ganze Zahl.) Für Wiederholungszeiten $t_r \geq 2^N$ setzt man $i = N$, während j in diesem Fall entfällt.

Die Indizes i und j werden binär codiert. Der Index i bestimmt den ersten Teil des Codeworts („*Präfix*“), der Index j den zweiten Teil des Codeworts („*Suffix*“). Das Präfix wird mit $\lfloor \log_2(N + 1) \rfloor$ Bits codiert. Falls $i < N$, so wird das Suffix mit i Bits codiert. Wenn $i = N$, dann wird die aktuelle Teilsequenz q_k (in binärer Form) als Suffix genommen. Für $N = 3$ beispielsweise ergibt sich die in Tab. 2.5 dargestellte Codierungstabelle.

Tab. 2.5 Codierungstabelle für den Willems-Algorithmus mit Teilblöcken der Länge $N = 3$

t_r	i	j	Präfix	Suffix	Codewortlänge
1	0	0	[00]	–	2
2	1	0	[01]	[0]	3
3	1	1	[01]	[1]	3
4	2	0	[10]	[00]	4
5	2	1	[10]	[01]	4
6	2	2	[10]	[10]	4
7	2	3	[10]	[11]	4
≥ 8	3	–	[11]	q_k	2 + Suffixlänge

Beispiel 2.2.6 Willems-Algorithmus

Gegeben sei folgendes Bild bestehend aus $3 \cdot 5 = 15$ Pixeln:

A	A	B	A	A
A	A	A	A	A
C	C	A	C	C

Jedes Pixel kann 4 mögliche Graustufen $A = [00]$, $B = [01]$, $C = [10]$ und $D = [11]$ annehmen. Ohne Quellencodierung würden demnach $15 \cdot 2 = 30$ Bits zur verlustlosen Repräsentation aller Pixel benötigt. Wir nehmen an, dass das Bild zeilenweise gelesen wird und erhalten die Quellensymbolsequenz $q = [AAB \text{ } AAA \text{ } AAA \text{ } ACC \text{ } ACC]$ der Länge $n = 15$. Wir wählen $N = 3$ als Länge der Teilsequenzen und erhalten somit $m = n/N = 5$ gleichlange Teilsequenzen. Der Inhalt des Puffers der Länge $2^N - 1 = 7$ sei mit [AAAAAAA] initialisiert. Die Verarbeitungsschritte des Willems-Algorithmus sind wie folgt:

1. Teilsequenz: [AAAAAAA]AAB $\Rightarrow t_r \geq 8 \Rightarrow u_1 = [11\ 000001]$
2. Teilsequenz: [AAAAAAB]AAA $\Rightarrow t_r = 4 \Rightarrow u_2 = [10\ 00]$
3. Teilsequenz: [AABABAA]AAA $\Rightarrow t_r = 1 \Rightarrow u_3 = [00]$
4. Teilsequenz: [BAAAAAA]ACC $\Rightarrow t_r \geq 8 \Rightarrow u_4 = [11\ 001010]$
5. Teilsequenz: [AAAAACC]ACC $\Rightarrow t_r = 3 \Rightarrow u_5 = [01\ 1]$

Aufgrund der Willems-Codierung werden (nur) 25 Bits zur verlustlosen Quellencodierung benötigt. Durch eine Vergrößerung des Puffers kann die Effizienz weiter gesteigert werden. \diamond

Wir erkennen, dass die Quellensymbolsequenz maximal um den Faktor $N/[\log_2(N + 1)]$ komprimiert wird (für $t_r = 1$) und höchstens um den Faktor $1 + [\log_2(N + 1)]/N$ expandiert wird (für $t_r \geq 2^N$). Man kann für stationäre ergodische Sequenzen $q = [q_1, q_2, \dots, q_m]$ zeigen, dass

$$\lim_{N \rightarrow \infty} E\{W\}/N = H_\infty(Q) \quad (2.43)$$

gilt. Der Willems-Algorithmus ist bei unendlich großer Pufferlänge $2^N - 1$ folglich optimal. Man kann also mit universellen Quellencodierv Verfahren die Entropierate erreichen. Insofern könnte man von einer *Entropierate*ncodierung sprechen, auch wenn diese Bezeichnung unüblich ist.

Eine Alternative zum Willems-Algorithmus stellt der Lempel-Ziv-Algorithmus dar, der in mehreren Ausführungsformen (LZ77, LZW, ...) auf vielen Computern zur Datenkompression eingesetzt wird. Beim Lempel-Ziv-Algorithmus wird eine variable Quellenwortlänge auf eine feste Codewortlänge abgebildet. Mit Hilfe des Lempel-Ziv-Algorithmus kann die Entropierate ebenfalls asymptotisch erreicht werden. Auf eine detaillierte Darstellung wird hier bewusst verzichtet, da der Lempel-Ziv-Algorithmus didaktisch weniger lehrreich als der Willems-Algorithmus ist.

Grundlagen der digitalen Informationsübertragung

Von der Theorie zu Mobilfunkanwendungen

Höher, P.A.

2013, X, 804 S. 455 Abb. Mit 234 Beispielen., Softcover

ISBN: 978-3-8348-1784-6