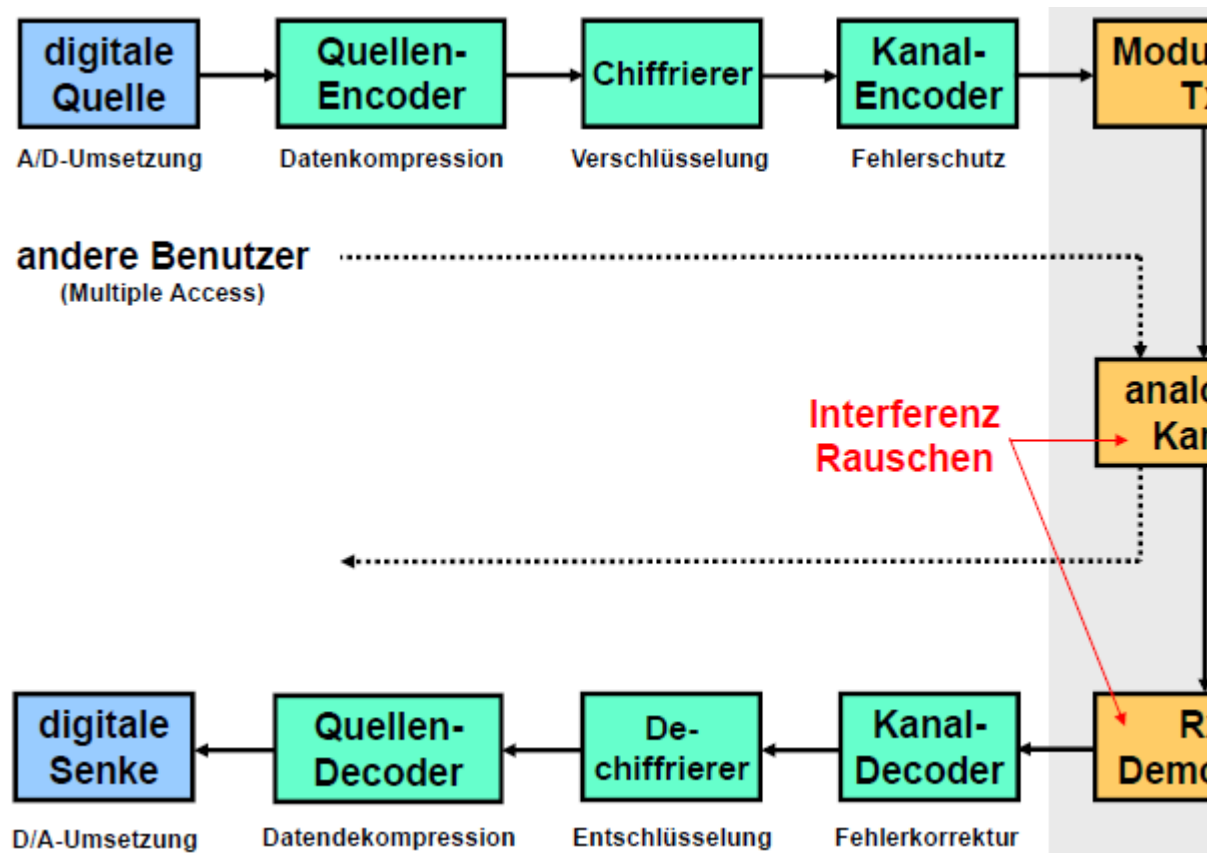


# INT Zusammenfassung

## Übertragungssysteme



## Quellencodierung

- Reduktion der Menger digitaler Daten
- Entfernen überflüssiger Informationen
- Verlustfreie Codierung
- Verlustbehaftete Kompression (Bilder, Video, Audio)

## Diskrete Informationsquellen

### DMS (Discrete Memoryless Source)

$X[n]$  unabhängig, besitzen identische Wahrscheinlichkeitsverteilung.

### BMS (Binary Memoryless Source)

$M = 2$ , d.h.  $P_x(x_1) = p$  und  $P_x(x_2) = 1-p$

### BSS (Binary Symmetric Source)

$P_x(x_1) = P_x(x_2) = 0.5$  "coin-flipping"

## Quellencodierungstheorem (Shannon, 1948)

Die Quelle kann verlustlos codiert werden, solange die Coderate (Durchschnitt. Codelänge)  $R \geq H$ .

Umgekehrt, wenn  $R < H$ , kann die Quelle auf keinen Fall verlustlos codiert werden.

### Optimale Codierung

Mittlere Codewortlänge = Entropie

## BSC - Binary Symmetric Channel

**Kapazität**  $C = \max_p(x)[H(Y) - H(Y|X)]$ ,  $C_{BSC} = 1 - h(\varepsilon)$  (*Bit/Kanalbenützung*)

### Kanaleigenschaften

- Rauschen im Kanal beschränkt nicht die Zuverlässigkeit der Übertragung, nur die Übertragungsrate
- Verschiedene Kanäle mit einer Zahl vergleichbar
- Je grösser die Blocklänge  $N$ , desto komplexer der decoder

## AWGN-Kanal

(Additive White Gaussian Noise)

Coderate

$$R = \frac{\#Infobits}{\#Codebits}$$

Kapazität:

$$C_{AWGN} = B \bullet \log_2 \left( 1 + \frac{\text{Signalleistung}}{\text{Bandbreite}[Hz] \bullet \text{Rauschleistungsdichte}[W/Hz]} \right)$$

## Entropie

Symboldauer	$T$
Symbolrate	$R = 1/T$
Quellensymbol (Zufallsvariable)	$X[n]$
Alphabet	$A = (x_1, x_2, \dots, x_M)$
Wahrscheinlichkeit	$P(X = x_m) = P_X(x_m), m = 1, \dots, M$
Wahrscheinlichkeitsverteilung von $X$	$\sum_{m=1}^M P_X(x_m) = 1$

## Informationsgehalt

Der Informationsgehalt eines Ereignisses  $X = x_m$  ist wie folgt definiert:

$$I_x(x_m) = \log_2 \left( \frac{1}{P_X(x_m)} \right) [\text{bit}]$$

Für Ereignisse von 2 (oder mehreren) Zufallsvariablen  $X$  und  $Y$  gilt sinngemäss:

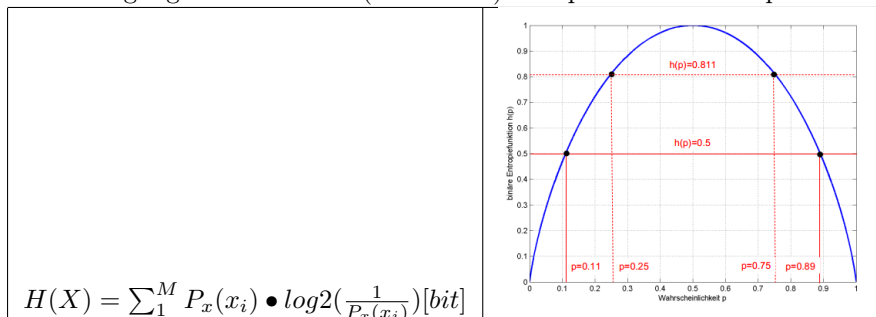
$$I_x(x_m) = \log_2 \left( \frac{1}{P_{XY}(x_i, y_k)} \right) [\text{bit}]$$

Für 2 unabhängige Symbole  $X$  und  $Y$  gilt:

$$I_{XY}(x_i, y_k) = I_X(x_i) + I_Y(y_k)$$

## Entropie

Datenübertragung: die maximale (verlustlose) Kompression = Entropie



## Huffman Code

### Eigenschaften

- Prefixfreier Code

- Huffman-Codes sind optimal
- Minimale mittlere Codewortlänge

### Nachteile

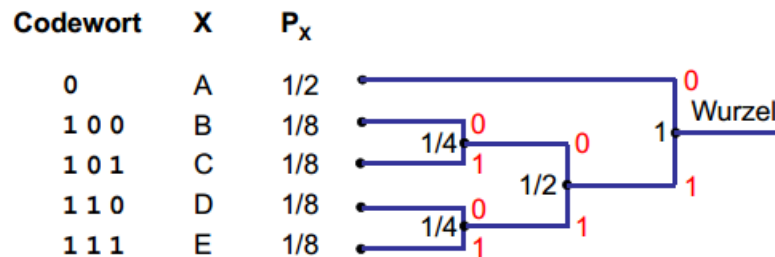
- Huffman-Codes hängen stark von der Quellenstatistik ab
- Quellenstatistik muss im voraus bekannt sein (ev. zuerst „messen“)
- Komplexität wächst exponentiell mit der Blocklänge  $n$

Codierung von  $n$  Symbolen eine DMS:

$$H(X) \leq R \leq H(X) + \frac{1}{n}$$

### Algorithmus

1. nach Wahrscheinlichkeiten ordnen
2. Zwei Symbole mit kleinster Wahrscheinlichkeit zusammenfassen, neuer Knoten hat Summe der Wahrscheinlichkeiten
3. -> Loop Schritt 1
4. Von der Wurzel aus bei jeder Verzweigung nach oben eine „0“ und nach unten eine „1“ eintragen (auch umgekehrt möglich) //Konstruktion Codebuch



mittlere Codewortlänge  $E[L]$ , respektive Rate  $R$

$R = \text{Wahrscheinlichkeit} \cdot \text{Codelänge}$  (bsp:  $1 \cdot 1/8 + (3 \cdot 1/8) \cdot 4 = 2$ )

$R = E[L] = \sum_{m=1}^M P_X(x_m) \cdot L(x_m)$  mit der Wahrscheinlichkeit  $P_X(x_m)$  und der Länge  $L(x_m)$  für ein Zeichen

## Lempel-Ziv

### Vorteile

- Unabhängig von der Quellenstatistik
- Universelle Anwendung
- Asymptotisch optimal, d.h. Codewortlänge  $R \rightarrow H(X)$  (von oben)

### Nachteile

- Anzahl Strings / Grösse des Wörterbuchs beschränkt
- Schlechte Kompression bei kleinen Eingangsbitfolgen

### Algorithmus

1. Eindeutige Unterteilung der Symbolfolge Strings variabler Länge, Unterscheidung nur in 1 Bit
2. Encoding eines Strings: [Position des Präfix, neues Bit]

### Beispiel

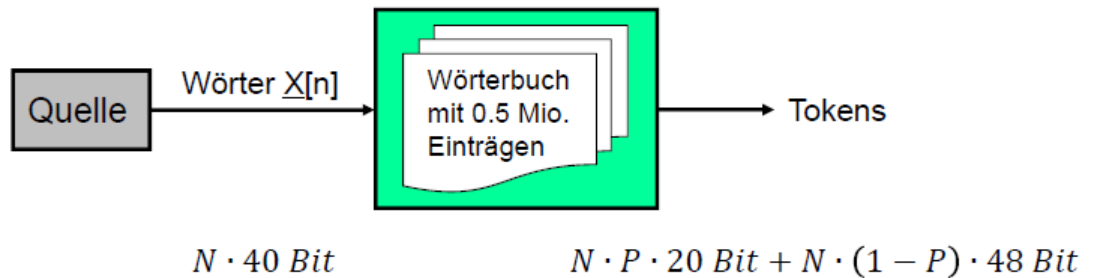
Data: 0 1 0 0 0 1 1 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 ...

Wörterbuch-Nr.	Input	Output
1	0 -> neuer String; neues Bit = 0	[0000 0]
2	1 -> neuer String; neues Bit = 1	[0000 1]
3	00 -> 0 gleich wie 1. String; neues Bit = 0	[0001 0]
4	001 -> 00 gleich wie 3. String; neues Bit = 1	[0011 1]
5	10 -> 1 gleich wie 2. String; neues Bit = 0	[0010 0]
6	000 -> 00 gleich wie 3. String; neues Bit = 0	[0011 0]
7	101 -> 10 gleich wie 5. String; neues Bit = 1	[0101 1]
8	0000 -> 000 gleich wie 6. String; neues Bit = 0	[0110 0]
9	01 -> 0 gleich wie 1. String; neues Bit = 1	[0001 1]
10	010 -> 01 gleich wie 9. String; neues Bit = 0	[1001 0]
...	...	...

## Statisches Wörterbuch

Wie oft muss ein Wort im Wörterbuch sein, damit Kompressionsrate  $R < 1$ ?

- Anzahl der Wörter =  $N$
- $P$  sei die Wahrscheinlichkeit, dass ein Wort im Buch gefunden wird
- Mittlere Wortgrösse: 5 Bytes bzw. 40 Bit



## LZ77

1. Erstes Symbol des Vorschau-Buffers im Such-Buffer suchen
  - (a) rückwärts von rechts nach links
2. Token der längsten (letzten) Übereinstimmung ausgeben
  - (a) Token = (Offset, Anzahl Zeichen, "Nächstes Zeichen")
  - (b) Token-Länge:  $\log_2(S + 1) + \log_2(L + 1) + 8$  typisch :  $11 + 5 + 8 = 24 \text{ Bit}$
  - (c) wenn keine Übereinstimmung: (0,0, nächstes Symbol)
3. 3. Schiebefenster um Länge +1 nach rechts verschieben

## LZ78

## LZW

Initialisierung  $I = []$

1. neues Symbol x zu String I hinzufügen =>  $I = Ix$  setzen
  - (a)  $Ix$  im Wörterbuch verzeichnet? Wenn ja, dann zu step 1. sonst zu step 3.
2. -
  - (a) Output = Wörterbuch-Pointer von I
  - (b) Neuer Wörterbucheintrag mit Phrase  $Ix$
  - (c)  $I = "x"$  setzen

## Beispiel Encoding

Text: ABBABABAC

Anfangswörterbuch: 1 : A, 2 : B, 3 : C

Momentane Buchstaben	String I	verzeichnet	WB-Eintrag	Output
A	A	✓		
A	AB	×	4 : AB	1
B	B	✓		
B	BB	×	5 : BB	2
B	B	✓		
B	BA	×	6 : BA	2
A	A	✓		
AB	AB	✓		
AB	ABA	×	7 : ABA	4
A	A	✓		
AB	AB	✓		
ABA	ABA	✓		
ABA	ABAC	×	8 : ABAC	7
C	C	✓		
C	C,eof			3

## Bsp Decoding (Lösung ist in String J)

68	68	82	99	77	65	82	256	82
D	E	R		M	A	R		R
Input	String I	String J	WB					
68	D	D						
69	D	E	256: DE					
82	E	R	257: ER					
95	R	_	258: R_					
77	_	M	259: _M					
65	M	A	260: MA					
82	A	R	261: AR					
256	R	DE	262: RD					
82	DE	R	263: DER					

## RLE (Run-Length-Encoding)

- 33333333  $\rightarrow$  (@,9,3)
- @  $\rightarrow$  (@,1,@)

Wird nur gebraucht wenn Token nicht länger als Orginaltext.

## PN-Sequenzen

Pseudo Noise Sequenzen

### LSFR

- Periode  $P \leq 2^n - 1$
- Pseudozufall
- Feedbackpolynom  $\rightarrow$  prim. Polynom zB: (3,1,0)

Für (Pseudo-)Randomgenerator

$$a_0 = (a_{18} + a_5 + a_2 + a_1) \text{ modulo } 2$$

### Zufallseigenschaften der m-Sequenzen

- m-Sequenzen sind fast ausgeglichen in der Anzahl "0" und "1"
- Häufigkeit von runs der Länge  $k$  beträgt  $(1/2)^k$  für  $k \leq (n-1)$  und  $12^{(k-1)}$  für  $k = n$ . Ein „run“ ist das Aufeinanderfolgen mehrere Nullen oder Einsen.
  - So weist zum Beispiel die Bitfolge ... 00110101000111001101... folgende runs auf: Run 1 kommt 6x vor, run 2 kommt 4x vor und run 3 kommt 2x vor.
- Die m-Sequenz der Länge  $P$  und die zyklisch verschobene Kopie haben fast 50 % übereinstimmende Bits und 50 % verschiedene Bits
- Maximallängensequenz  $\rightarrow$  PN-Sequenz
- PN-Sequenz  $\nrightarrow$  Maximallängensequenz

## Primitive Polynome

### BlockCodes

- $N$ : Anzahl Bits in einem Wort nach dem Encoding i.e [1,1,1,0,1,1]  $\rightarrow 6$
- $K$ :  $2^K =$  Anzahl Infoworte, i.e {[1,1,1],[1,0,1]}  $\rightarrow 2$  oder Anzahl Bits in einem Wort vor dem Encoding
- Coderate  $R = \frac{K}{N}$
- Minimum Distance Encoding
  - wenige Fehler sind wahrscheinlicher als viele
  - Zuweisung and "nächstgelegenes" Codewort



**Hamming-Gewicht**  $w_H(x)$

entspricht der Anzahl "1" im Codewort  $x$

**Hamming-Distanz**  $d_H(x_i, x_j)$

entspricht der Anzahl unterschiedlicher Positionen in  $x_i$  und  $x_j$

**Minimaldistanz**  $d_{min}$

$d_{min} = \min_{i,j} d_H(x_i, x_j) = \min_{i,j} w_H(x_i + x_j) = \min_k w_H(x_k) = w_{min}(i \neq j)$   
Für linearen (N,K) Block-Codes

**Beispiel: (3,2)-Blockcode**

Anzahl Informationsbits (Infowort  $u$ ) =  $K = 2$ , Länge eines Codewortes ( $x$ ) =  $N = 3$

$2^K = 4$  Infoworte

Coderate =  $R = \frac{K}{N}$

$A = \{[00], [01], [10], [11]\}$

even Parity

$C = \{[000], [101], [110], [011]\}$  (vorderstes Bit ist hier Paritybit)

**Begriff ,systematischer Block-Code '**

Infowort "enblock" in Codewort.  $Cw = \text{Parity} + Iw$

**Begriff ,linearer (N,K) Block-Code C'**

Falls die modulo-2 Summe zweier Codewörter wieder ein Codewort ergibt, dann ist der Block Code linear.

**Begriff ,linearer, zyklischer (N,K) Block-Code C'**

Falls die zyklische Verschiebung eines Codeworts wieder ein Codewort ergibt, ist der Code ausser- dem zyklisch. Aufgrund dieser Eigenschaft sind die verschiedenen Codeworte sehr einfach mit Hilfe eines LFSR (Linear Feedback Shift Register) realisierbar.

**Generator-Matrix**

Für jeden linearen  $(N, K)$  Code gibt es eine  $K \times N$  Generator-Matrix  $G$

$$[x_0, \dots, x_{N-1}] = [u_0, \dots, u_{K-1}] \cdot G$$

Die Generator-Matrix hat die Form  $G = [PI_K]$ ,  $I_K: K \times K$ -Einheitsmatrix

### Parity-Check-Matrix

Jeder lineare  $(N, K)$  Code hat eine  $(N - K) \times N$  Parity-Check-Matrix  $H$

$$[x_0, \dots, x_{N-1}] \cdot H^T = [0, \dots, 0]$$

Wenn  $G = [PI_K]$  in systematischer Form, dann  $H = [I_{N-K}P^T]$

### Syndrom

$$\vec{s} = [s_0, \dots, s_{N-K-1}] = \vec{y} \cdot H^T = (\vec{x} + \vec{e}) \cdot H^T = \vec{e} \cdot H^T$$

Wobei  $\vec{e}$  der Fehler ist und  $\vec{y}$  das neue Codewort mit dem Fehler addiert (also ein potenziell falsches Wort, falls  $\vec{e} \neq \vec{0}$ )

Das Syndrom ist nur vom Fehler abhängig. Falls keine Fehler übertragen wurden ist  $\vec{s} = \vec{0}$ .

### Fehlererkennung

alle Muster mit  $\leq (d_{min} - 1)$  Fehler sind erkennbar

### Fehlerkorrektur

$$t \leq \lfloor (d_{min} - 1)/2 \rfloor$$

**(N,K,t)BC = NK-BC der t Fehler korrigieren Kann**

$$P = \sum_0^t \binom{N}{i} \bullet \varepsilon^i \bullet (1-\varepsilon)^{N-i} = \sum_0^t \left( \frac{N!}{i! \bullet (N-i)!} \right) \bullet \varepsilon^i \bullet (1-\varepsilon)^{N-i}$$

Wahrscheinlichkeit für i Fehler pro Code Wort

$$\binom{N}{i} \bullet \varepsilon^i \bullet (1-\varepsilon)^{N-i} = \frac{N!}{i! \bullet (N-i)!} \bullet \varepsilon^i \bullet (1-\varepsilon)^{N-i}$$