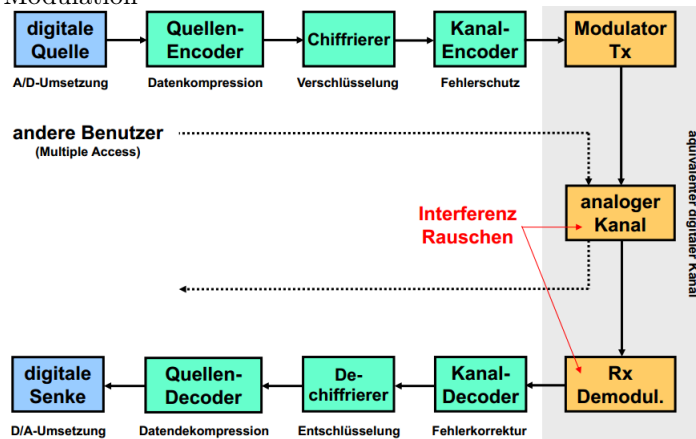


# INFORMATIONSTHEORIE

## Part 1. Kompression

### 1. ELEMENTE IM ÜBERTRAGUNGSSYSTEM

- Quelle/Senke
- Quellencodierung
- Chiffrierung
- Kanalcodierung
- Modulation



## Part 2. Entropie

### 2. DISKRETE INFORMATIONSQUELLEN

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 $X$	$\sum_{m=1}^M P_X(x_m) = 1$

#### 2.0.1. gedächtnislose Quellen.

- DMS (Discrete Memoryless Source), Die Symbole  $X[n]$  sind unabhängig und haben identische Wahrscheinlichkeitsverteilung.
- BMS (Binary Memoryless Source), Die unabhängigen Symbole  $X[n]$  sind 2-wertig, d.h.  $P_X(x_1) = p$  und  $P_X(x_2) = 1 - p$ .
- BSS (Binary Symmetric Source), Die unabhängigen Symbole  $X[n]$  sind 2-wertig und es gilt:  $P_X(x_1) = 0.5$  und  $P_X(x_2) = 0.5$ .

## 3. 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:

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

## 4. REDUNDANZ

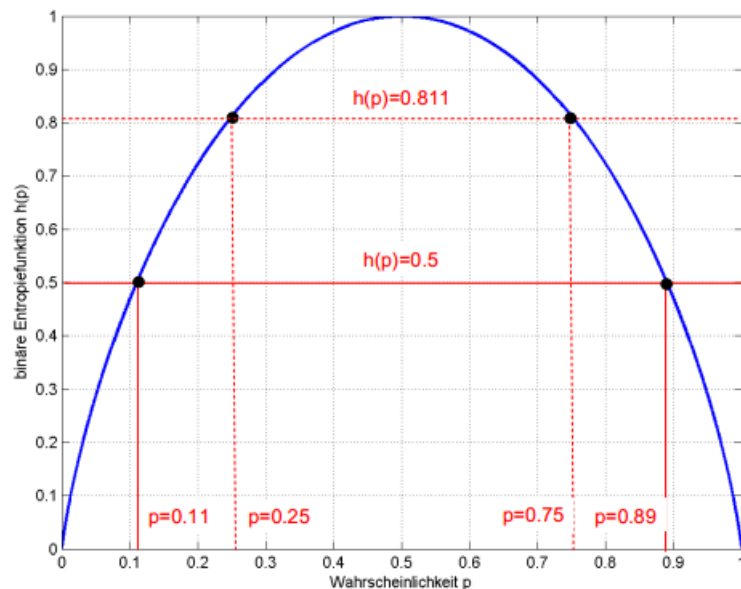
Differenz zwischen maximaler und mittlerer Entropie. Redundanz ( $M$  ist die Anzahl Symbole des Alphabets)  
Entropie ist maximal  $\log_2(M)$ , wenn  $X$ -Werte gleichverteilt.  
Möglichst wenig Redundanz am Ausgang des Quellencoders.

$$R = \log_2(M) - H(x)$$

## 5. ENTROPIE

Datenübertragung: die maximale (verlustlose) Kompression = Entropie

$$H(X) = \sum_{i=1}^M P_x(x_i) \cdot \log_2 \left( \frac{1}{P_X(x_i)} \right) [\text{bit}]$$



## 5.1. Binäre Entropiekurve.

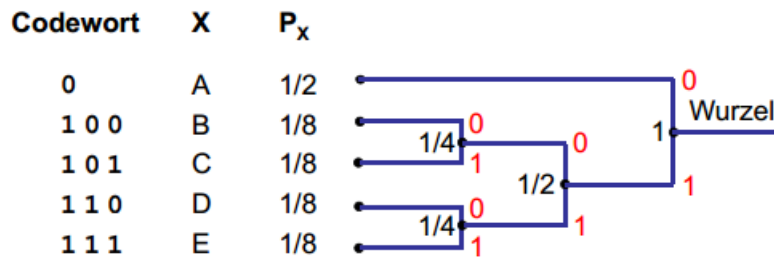
### Part 3. Kompression

#### 6. HUFFMAN CODE

Abhängig von der Quellenstatistik

##### Algorithmus.

- (1) Symbole nach Wahrscheinlichkeiten ordnen und Knoten eines Baums zuweisen
- (2) Zwei Symbole mit kleinster Wahrscheinlichkeit in neuem Symbol zusammenfassen, neuer Knoten hat Summe der Wahrscheinlichkeiten
- (3) Erneutes Reduzieren des Wahrscheinlichkeitsfeldes gem. Schritt 1
- (4) Schritte 2 und 3 wiederholen bis 2 Symbole bzw. Knoten übrig
- (5) Von der Wurzel aus bei jeder Verzweigung nach oben eine „0“ und nach unten eine „1“ eintragen (auch umgekehrt möglich) //Konstruktion Codebuch



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

#### 7. LEMPEL-ZIV-CODIERUNG

Unabhängig von der Quellenstatistik

##### 7.1. Algorithmus.

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

7.2. **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]
...	...	...

#### 8. 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, Länge, nächstes Symbol)
  - (b) Token-Länge:  $\log_2(S+1) + \log_2(L+1) + 8$  typisch :  $11 + 5 + 8 = 24$  Bit
  - (c) wenn keine Übereinstimmung: (0,0, nächstes Symbol)
- (3) 3. Schiebefenster um Länge +1 nach rechts verschieben

9. LZ78

10. 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		
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

## 11. JPG

statt Redundanzreduktion vor allem Irrelevanzreduktion (Qualitätsverlust, häufig jedoch nicht bemerkbar)

**RLE.**

11.1. **PN-Sequenzen.** Pseudo Noise Sequenzen

*LSFR.* Für (Pseudo-)Randomgenerator

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

11.1.1. *Zufallseigenschaften der m-Sequenzen.*

- m-Sequenzen sind fast ausgeglichen in Bezug auf die Anzahl Einsen und Nullen
- Die relative Häufigkeit von runs der Länge  $k$  beträgt  $(1/2)^k$  für  $k \leq (n-1)$  und  $1/2^{(k-1)}$  für  $k = n$ . Ein „run“ ist das Aufeinanderfolgen mehrerer 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

Beispiel.

06 May 2014 09:58

primitives Polynom:  $(4, 1, 0) \rightarrow$  Feedback-Polynom:  $x^4 + x^1 + 1$

A diagram of a 4-bit shift register. It consists of four square cells connected in a row. Above the cells are inputs  $x^4, x^3, x^2, x^1$  with arrows pointing down to the cells. The cell under  $x^4$  has an output arrow pointing left labeled "output". The cell under  $x^1$  has an output arrow pointing right to a circle containing a plus sign (+). A feedback line connects the output of the  $x^1$  cell back to the input of the  $x^4$  cell.

$x^6 \ x^3 \ x^2 \ x^1$   
 Vorher.  $\oplus$  Vorher. = neues  
 $x^6 \ x^3 \ x^2 \ x^1$

seed: 1111

output	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1	1	1	1	1	0	1	$\oplus$	1	=	0									
1	1	1	0	1	0	1	$\oplus$	0	=	1									
1	1	0	1	0	1	0	$\oplus$	1	=	0									
1	0	1	0	1	0	1	$\oplus$	0	=	1									
0	1	0	1	1	0	0	$\oplus$	1	=	1									
1	0	1	1	0	0	1	$\oplus$	1	=	0									
0	1	1	0	0	0	1	$\oplus$	0	=	0									
1	1	0	0	1	0	1	$\oplus$	0	=	1									

output  $\rightarrow$

1	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1	0	0	1	0	1	0	$\oplus$	1	=	0									
0	0	1	0	0	0	0	$\oplus$	0	=	0									
0	1	0	0	0	0	0	$\oplus$	0	=	0									
1	0	0	0	1	1	0	$\oplus$	0	=	1									
0	0	0	1	1	0	0	$\oplus$	1	=	1									
0	0	1	1	1	0	0	$\oplus$	1	=	1									
0	1	1	1	1	0	0	$\oplus$	1	=	1									

$\Rightarrow$  Periode von 15

Überprüfen der Zufallseigenschaften (des Outputs):

2. Runlänge:  $k$   
 Anzahl Runs der Länge  $k$ :  $\#$   
 Formel:  $P = \left(\frac{1}{2}\right)^k$  für  $k \leq (n-1)$  und  $P = \left(\frac{1}{2}\right)^{k-1}$  für  $k = n$   
 Test:  $\sum \# = 4 + 2 + 1 + 1 = 8$ ; muss gelten:  $P * 8 = \#$

Theorie Seite 1

Test:  $\sum \# = 4 + 2 + 1 + 1 = 8$ ; muss gelten:  $P * 8 = \#$

k	#	P	Test	
1	4	$(1/2)^1 = 1/2$	$8 \cdot 1/2 = 4$	✓
2	2	$(1/2)^2 = 1/4$	$8 \cdot 1/4 = 2$	✓
3	1	$(1/2)^3 = 1/8$	$8 \cdot 1/8 = 1$	✓
4	1	$(1/2)^{4-1} = 1/8$	$8 \cdot 1/8 = 1$	✓

3.

Output	111101011001000
Output um 1 nach rechts geschiftet	011110101100100
XOR der zwei Sequenzen soll die gleiche Sequenz wieder ergeben aber geschiftet	100011110101100



## 11.2. Binäre Block-Codes.

11.2.1. *Hamming-Gewicht*  $w_H(x)$ . entspricht der Anzahl "1" im Codewort  $x$

11.2.2. *Hamming-Distanz*  $d_H(x_i, x_j)$ . entspricht der Anzahl unterschiedlicher Positionen in  $x_i$  und  $x_j$

11.2.3. *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

11.2.4. *Beispiel: (3,2)-Blockcode*. Anzahl Informationsbits =  $K = 2$ , Länge eines Codewortes =  $N = 3$

$2^K = 4$  Infoworte

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

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

even Parity

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