64-040 Modul IP7: Rechnerstrukturen

6. Codierung und Informationstheorie

Norman Hendrich

Universität Hamburg
MIN Fakultät, Department Informatik
Vogt-Kölln-Str. 30, D-22527 Hamburg
hendrich@informatik.uni-hamburg.de

WS 2013/2014

Inhalt

Definition: Codierung

Unter **Codierung** versteht man das Umsetzen einer vorliegenden Repräsentation A in eine andere Repräsentation B.

- häufig liegen beide Repräsentationen A und B in der selben Abstraktionsebene

- die Interpretation von B nach A muss eindeutig sein
- eine **Umcodierung** liegt vor, wenn die Interpretation umkehrbar eindeutig ist

Code, Codewörter

- die Wörter der Repräsentation B aus einem Zeichenvorrat Z heißen **Codewörter**

- Die Menge aller Codewörter heißt **Code**
- **Blockcode**: alle Codewörter haben die selbe Länge

- **Binärzeichen**: der Zeichenvorrat z enthält genau zwei Zeichen
- **Binärwörter**: Codewörter aus Binärzeichen
- **Binärcode**: alle Codewörter sind Binärwörter

Gründe für den Einsatz von Codes

- effiziente Darstellung und Verarbeitung von Information
- Datenkompression, -reduktion

- effiziente Übertragung von Information
  - Verkleinerung der zu übertragenden Datenmenge
  - Anpassung an die Technik des Übertragungskanals
  - Fehlererkennende und -korrigierende Codes

- Geheimhaltung von Information (Chiffrierung in der Kryptologie)
- Identifikation, Authentifikation

Wichtige Aspekte

Unterteilung gemäß der Aufgabenstellung

- **Quellencodierung** (Anpassung an Sender/Quelle)
- **Kanalcodierung** (Anpassung an Übertragungsstrecke)
- **Verarbeitungscodierung** (im Rechner)

- sehr unterschiedliche Randbedingungen und Kriterien für diese Teilbereiche. Zum Beispiel sind fehlerkorrigierende Codes bei der Nachrichtenübertragung essentiell, im Rechner wegen der hohen Zuverlässigkeit weniger wichtig.

Darstellung von Codes

- **Wertetabellen**: jede Zeile enthält das Urbild (zu codierende Symbol) und das zugehörige Codewort
- sortiert, um das Auffinden eines Codeworts zu erleichtern
- technische Realisierung durch Ablegen der Wertetabelle im Speicher, Zugriff über Adressierung anhand des Urbilds

- **Codebäume**: Anordnung der Symbole als Baum,
- die zu codierenden Symbole als Blätter
- die Zeichen an den Kanten auf dem Weg von der Wurzel zum Blatt bilden das Codewort

- **Logische Gleichungen**
- **Algebraische Ausdrücke**

Codierung von Text

- siehe letzte Woche
- Text selbst als Reihenfolge von Zeichen
- ASCII, ISO-8859 und Varianten, Unicode

Für geschriebenen (formatierten) Text:

- Trennung des reinen Textes von seiner Formatierung
- Formatierung: Schriftart, Größe, Farbe, usw.
- diverse applikationsspezifische Binärformate
- Markup-Sprachen (SGML, HTML)

Codierungen für Dezimalziffern

	BCD	Gray	Exzess3	Aiken	biquinär	1-aus-10	2-aus-5
0	0000	0000	0011	0000	000001	0000000001	11000
1	0001	0001	0100	0001	000010	0000000010	00011
2	0010	0011	0101	0010	000100	0000000100	00101
3	0011	0010	0110	0011	001000	0000001000	00110
4	0100	0110	0111	0100	010000	0000010000	01001
5	0101	0111	1000	1011	100001	0000100000	01010
6	0110	0101	1001	1100	100010	0001000000	01100
7	0111	0100	1010	1101	100100	0010000000	10001
8	1000	1100	1011	1110	101000	0100000000	10010
9	1001	1101	1100	1111	110000	1000000000	10100

Codierungen für Dezimalziffern

- alle Codes der Tabelle sind Binärcodes
- alle Codes der Tabelle sind Blockcodes
- jede Spalte der Tabelle listet alle Codewörter eines Codes

- jede Wandlung von einem Code der Tabelle in einen anderen Code ist eine Umcodierung
- aus den Codewörtern geht **nicht** hervor, welcher Code vorliegt

- Dezimaldarstellung in Rechnern unüblich, die obigen Codes werden also kaum noch verwendet

Begriffe für Binärcodes

- **Minimalcode**: alle $N = 2^n$ Codewörter bei Wortlänge n werden benutzt
- **Redundanter Code**: nicht alle möglichen Codewörter werden benutzt

- **Gewicht**: Anzahl der Einsen in einem Codewort
- **komplementär**: zu jedem Codewort c existiert ein gültiges Codewort $\overline{c}$
- **einschrittig**: aufeinanderfolgende Codewörter unterscheiden sich nur an einer Stelle
- **zyklisch**: bei n geordneten Codewörtern ist c_0 gleich c_n

Dualcode

- der Name für Codierung der Integerzahlen im Stellenwertsystem
- Codewort

$$c = \sum_{i=0}^{n-1} a_i \cdot 2^i, \qquad a_i \in \{0, 1\}$$

- alle Codewörter werden genutzt: Minimalcode
- zu jedem Codewort existiert ein komplementäres Codewort
- bei fester Wortbreite ist c_0 gleich c_n: zyklisch
- nicht einschrittig

Einschrittige Codes

- möglich für Mengen mit Ordnungsrelation
- Elemente der Menge werden durch Binärwörter codiert
- **einschrittiger Code**: die Codewörter für benachbarte Elemente der Menge unterscheiden sich in genau einer Stelle
- **zyklisch einschrittig**: das erste und letzte Wort des Codes unterscheiden sich ebenfalls genau in einer Stelle

- Einschrittige Codes werden benutzt, wenn ein Ablesen der Bits auch beim Wechsel zwischen zwei Codeworten möglich ist (bzw. nicht verhindert werden kann)
- z.B.: Winkelcodierscheiben oder digitale Schieblehre
- viele interessante Varianten möglich: siehe Knuth, AoCP

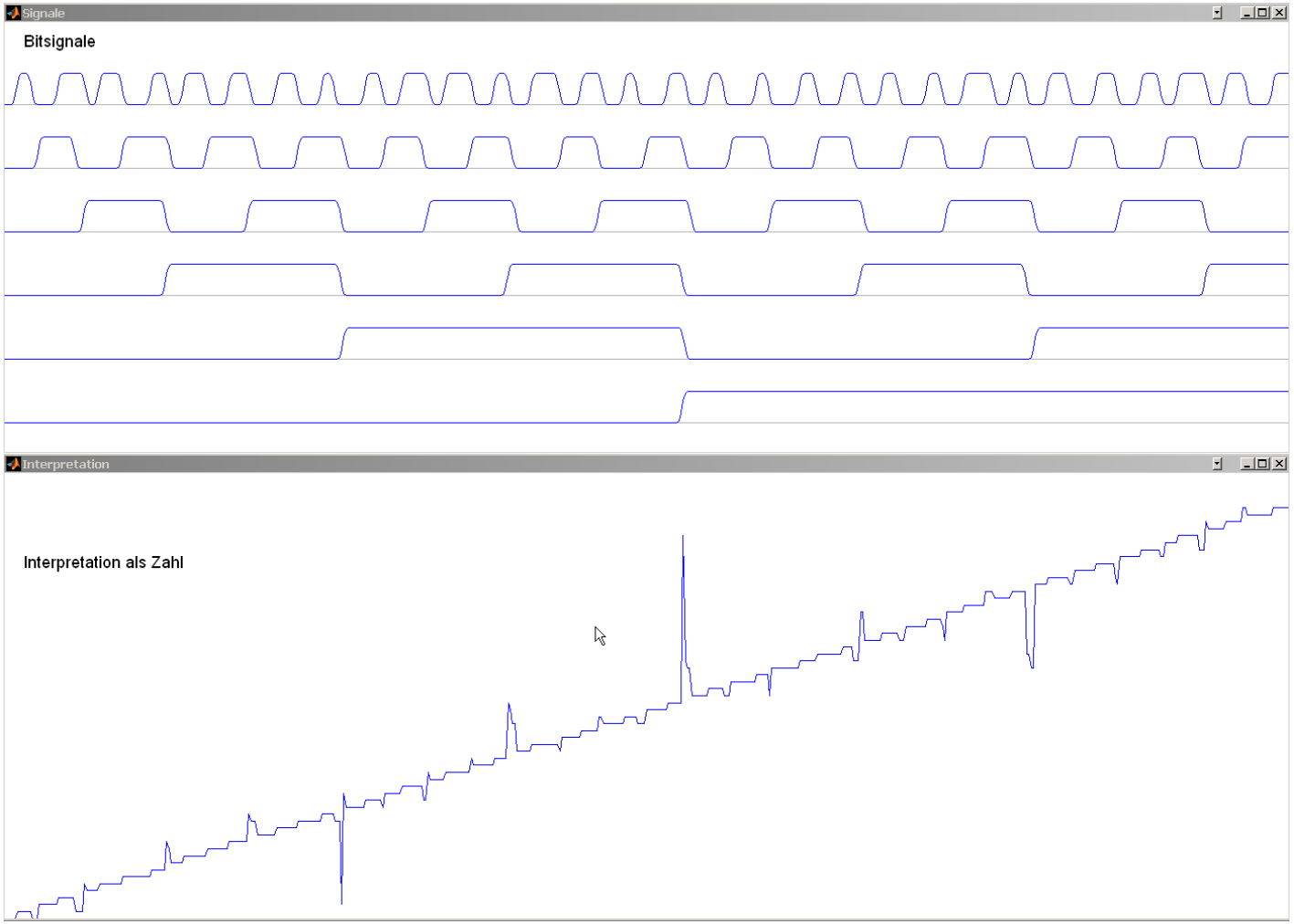
Einschrittige Codes: Matlab-Demo

- T1-Skript, Kapitel 1.4: Ablesen eines Wertes mit leicht gegeneinander verschobenen Übergängen der Bits

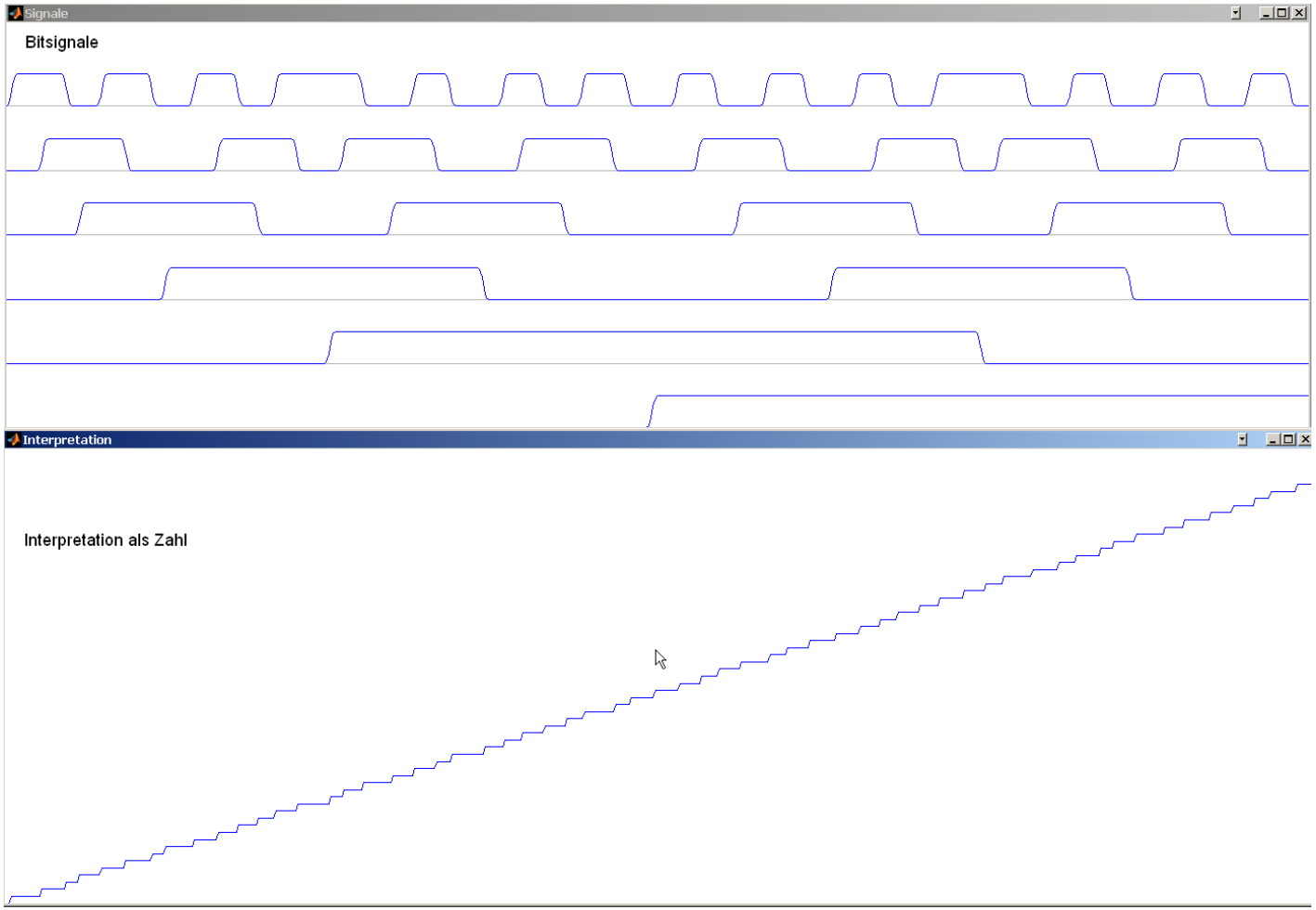
- `demoeinschritt(0:59)` normaler Dualcode
- `demoeinschritt(einschritt(60))` einschrittiger Code
- maximaler Ablesefehler ist 1 beim einschrittigen Code, aber 2^{n-1} beim Dualcode

- Konstruktion eines einschrittigen Codes rekursiv
- oder als ununterbrochenen Pfad im KV-Diagramm (s.u.)

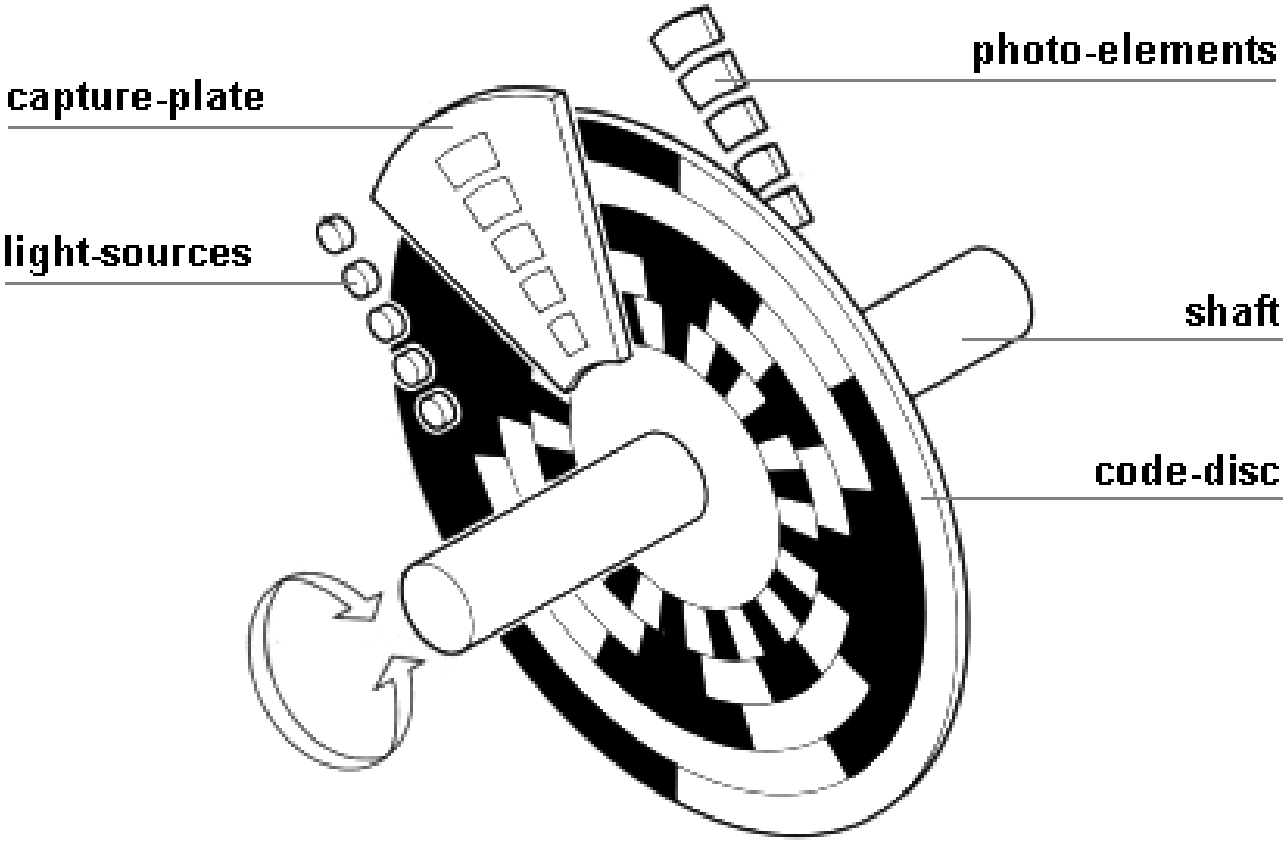
Ablesen des Wertes aus einem Dualcode



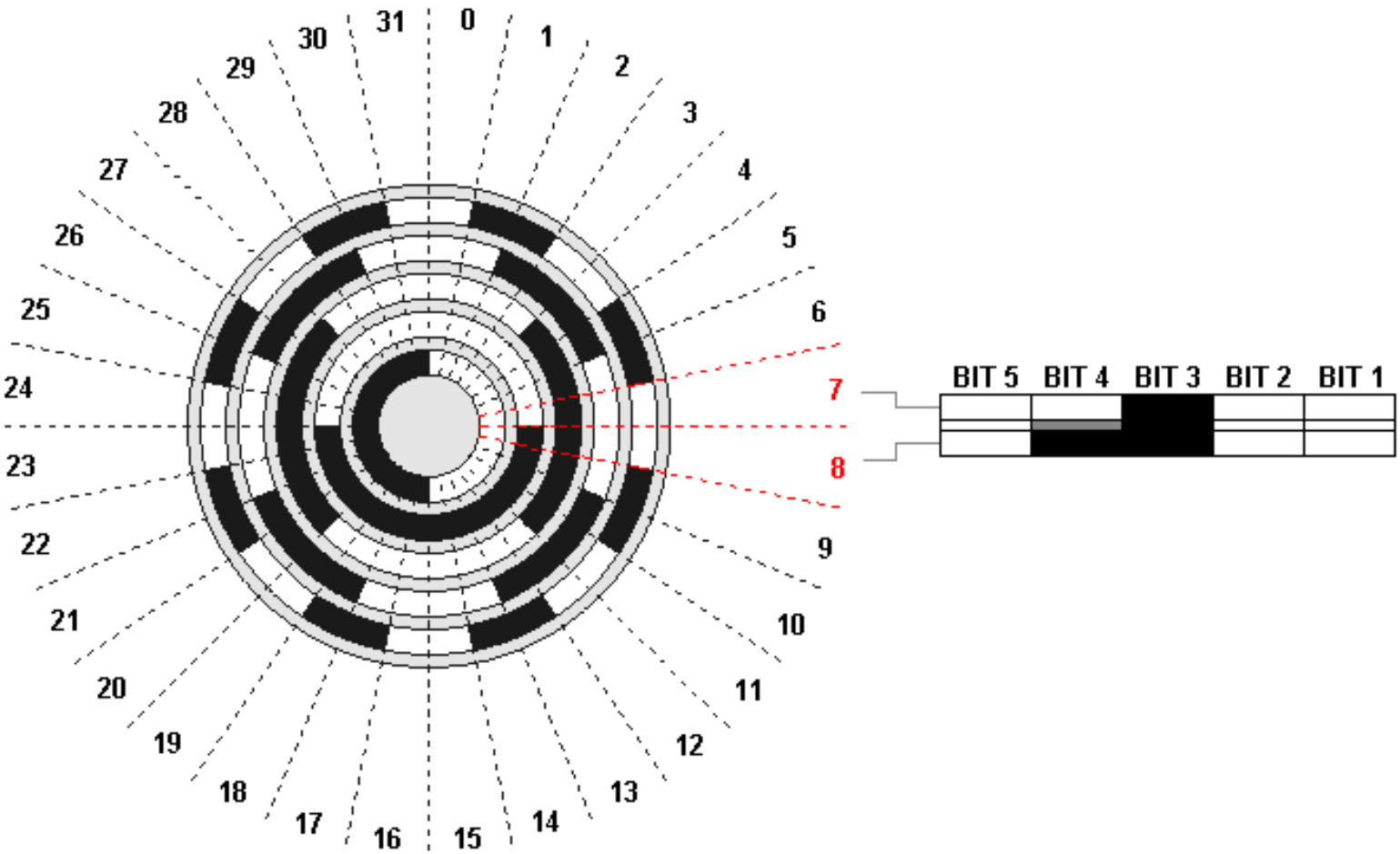
Ablesen des Wertes aus einschrittigem Code



Gray-Code: Prinzip eines Winkeldrehgebers

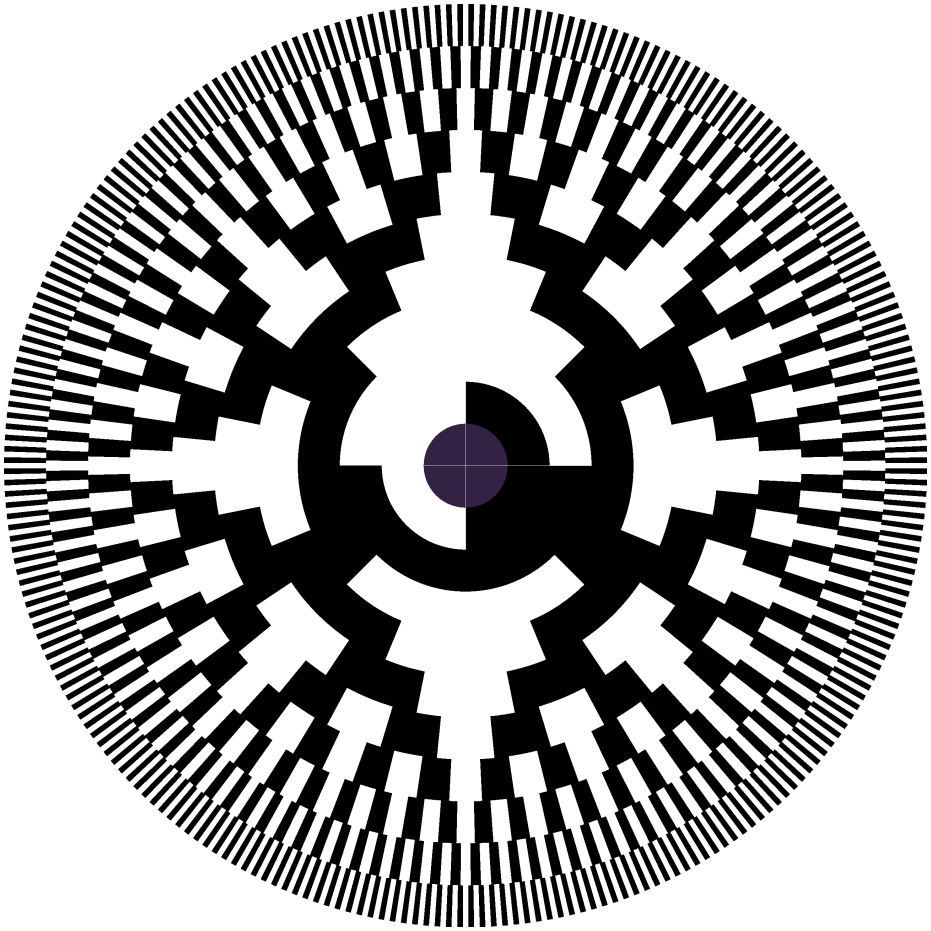


Gray-Code: 5-bit Codierscheibe



(Demo: java GreycodeWheel 200 10 5 10)

Gray-Code: 10-bit Codierscheibe



(Demo: java GreycodeWheel 600 5 5 10)

Einschrittiger Code: rekursive Konstruktion

- Starte mit zwei Codewörtern: 0 und 1

- Gegeben: Einschrittiger Code C mit n Codewörtern
- Rekursion: Erzeuge Code C' mit (bis zu) $2n$ Codewörtern:
  1. hänge eine führende 0 vor alle vorhandenen n Codewörter
  2. hänge eine führende 1 vor die in umgekehrter Reihenfolge notierten Codewörter

$\{0, 1\}$

$\{00, 01, \quad 11, 10\}$

$\{000, 001, 011, 010, \quad 110, 111, 101, 100\}$

$\ldots$

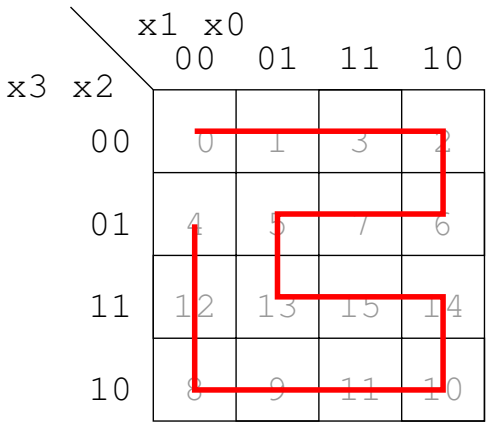
Karnaugh-Veitch Diagramm

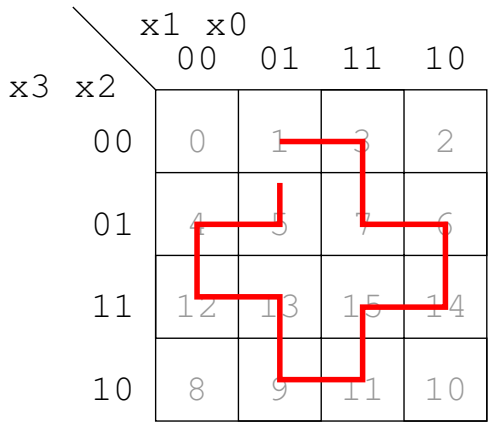
x3 x2 \ x1 x0	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

x3 x2 \ x1 x0	00	01	11	10
00	0000	0001	0011	0010
01	0100	0101	0111	0110
11	1100	1101	1111	1110
10	1000	1001	1011	1010

- 2D-Diagramm mit $2^n = 2^{n_y} \times 2^{n_x}$ Feldern
- gängige Größen sind 2x2 2x4 4x4
- darüber hinaus: mehrere Diagramme der Größe 4x4
- Anordnung der Indizes ist im einschrittigen-Code
- benachbarte Felder unterscheiden sich gerade um 1 Bit

Einschrittiger Code: KV-Diagramm

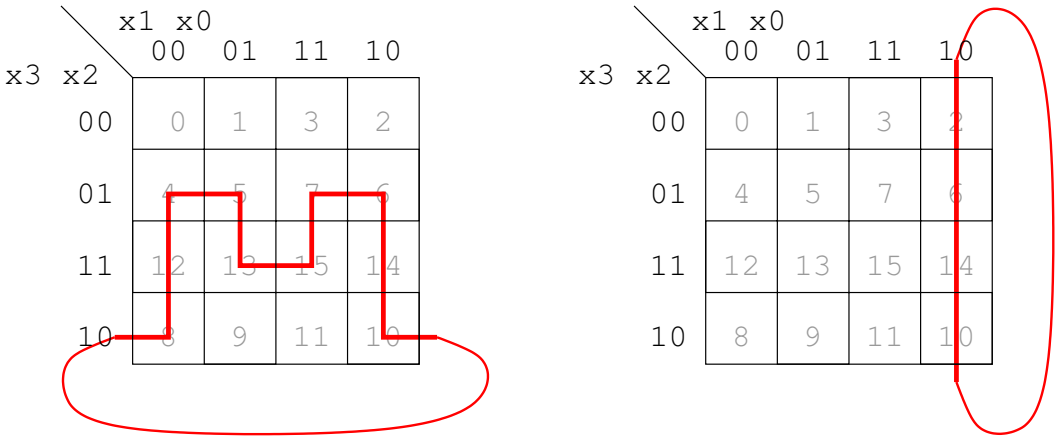




- jeder Pfad entspricht einem einschrittigen Code
- geschlossener Pfad: zyklisch einschrittiger Code

- links: `0,1,3,2, 6,7,5, 13,15,14, 10,11,9,8, 12,4`
- rechts: `1,3,7,6,14,15,11,9,13,12,4,5`

Einschrittiger Code: KV-Diagramm (2)



- linke und rechte Spalte unterscheiden sich in 1 Bit
- obere und untere Zeile unterscheiden sich in 1 Bit
- KV-Diagramm als „außen zusammengeklebt“ denken
- (geschlossener) Pfad: (zyklisch) einschrittiger Code
- Pfade können auch „außenherum“ geführt werden
- links: `4,5,13,15,7,6,14,10,8,12` rechts: `2,6,14,10`

Gray-Code: Umwandlung in/von Dualcode

Unwandlung: Dual- in Graywort

- MSB des Dualworts wird MSB des Grayworts
- von links nach rechts: bei jedem Koeffizientenwechsel im Dualwort wird das entsprechende Bit im Graywort 1, sonst 0.
  0011 $\rightarrow$ 0010, 1110 $\rightarrow$ 1001, usw.
- `gray(x) = x ^ (x >>> 1)`

Umwandlung: Gray- in Dualwort

- MSB wird übernommen
- von links nach rechts: wenn das Graywort eine Eins aufweist, wird das vorhergehende Bit des Dualworts invertiert in die entsprechende Stelle geschrieben, sonst wird das Zeichen der vorhergehenden Stelle direkt übernommen
  0010 $\rightarrow$ 0011, 1001 $\rightarrow$ 1110, usw.

Optimalcodes: Codes variabler Länge

- Einsatz zur Quellencodierung
- Minimierung der Datenmenge durch Anpassung an die Symbolhäufigkeiten
- häufige Symbole bekommen kurze Codewörter, seltene Symbole längere Codewörter

- anders als bei Blockcodes ist die Trennung zwischen Codewörtern nicht durch Abzählen möglich
- Einhalten der **Fano-Bedingung** notwendig
- oder Einführen von Markern zwischen den Codewörtern

Fano-Bedingung

eindeutige Decodierung eines Codes mit variabler Wortlänge?

- **Fano-Bedingung**: kein Wort aus einem Code bildet den Anfang eines anderen Codewortes
- die sogenannte **Präfix-Eigenschaft**
- nach R. M. Fano (1961)

- ein **Präfix-Code** ist eindeutig decodierbar
- Blockcodes sind Präfix-Codes

Fano-Bedingung: Beispiele

- Telefonnummern: das Vorwahlsystem gewährleistet die Fano-Bedingung

110, 112:	Notrufnummern
42883 2399:	Ortsnetz (keine führende Null)
040 42883 2399:	nationales Netz
0049 40 42883 2399:	internationale Rufnummer

- Morse-Code: Fano-Bedingung verletzt

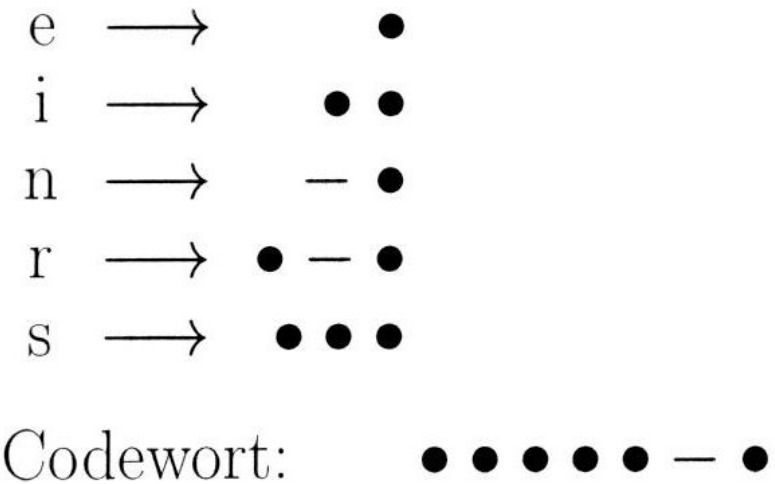
Morse-Code

Codetabelle des Morse-Codes

a	● –	o	– – –	4	● ● ●● –
ä	● – ● –	ö	– – – ●	5	● ● ● ● ●
å	● – – ● –	p	● – – ●	6	– ● ● ●●
b	– ● ● ●	q	– – ● –	7	– – ● ● ●
c	– ● – ●	r	● – ●	8	– – – ●●
ch	– – – –	s	● ● ●	9	– – – – ●
d	– ●●	t	–		
e	●	u	●● –	.	● – ● – ● –
é	●● – ●●	ü	●● – –	,	– – ●● – –
f	●● – ●	v	● ● ● –	:	– – – ● ● ●
g	– – ●	w	● – –	-	– ● ● ●● –
h	● ● ●●	x	– ●● –	'	● – – – – ●
i	●●	y	– ● – –	(	– ● – – ● –
j	● – – –	z	– – ●●	?	●● – – ●●
k	– ● –			“	● – ●● – ●
l	● – ●●	0	– – – – –	Notruf	● ● ● – – – ● ● ●
m	– –	1	● – – – –	SP	●●
n	– ●	2	●● – – –	Anfang	– ● – ● –
ñ	– – ● – –	3	● ● ● – –	Ende	● ● ● – ● –

Morse-Code: Präfix-Bedingung verletzt

- Punkt steht für kurzen Ton, Strich für langen Ton



- bestimmte Morse-Sequenzen sind mehrdeutig
- Pause zwischen den Symbolen notwendig

Morse-Code: Umschlüsselung

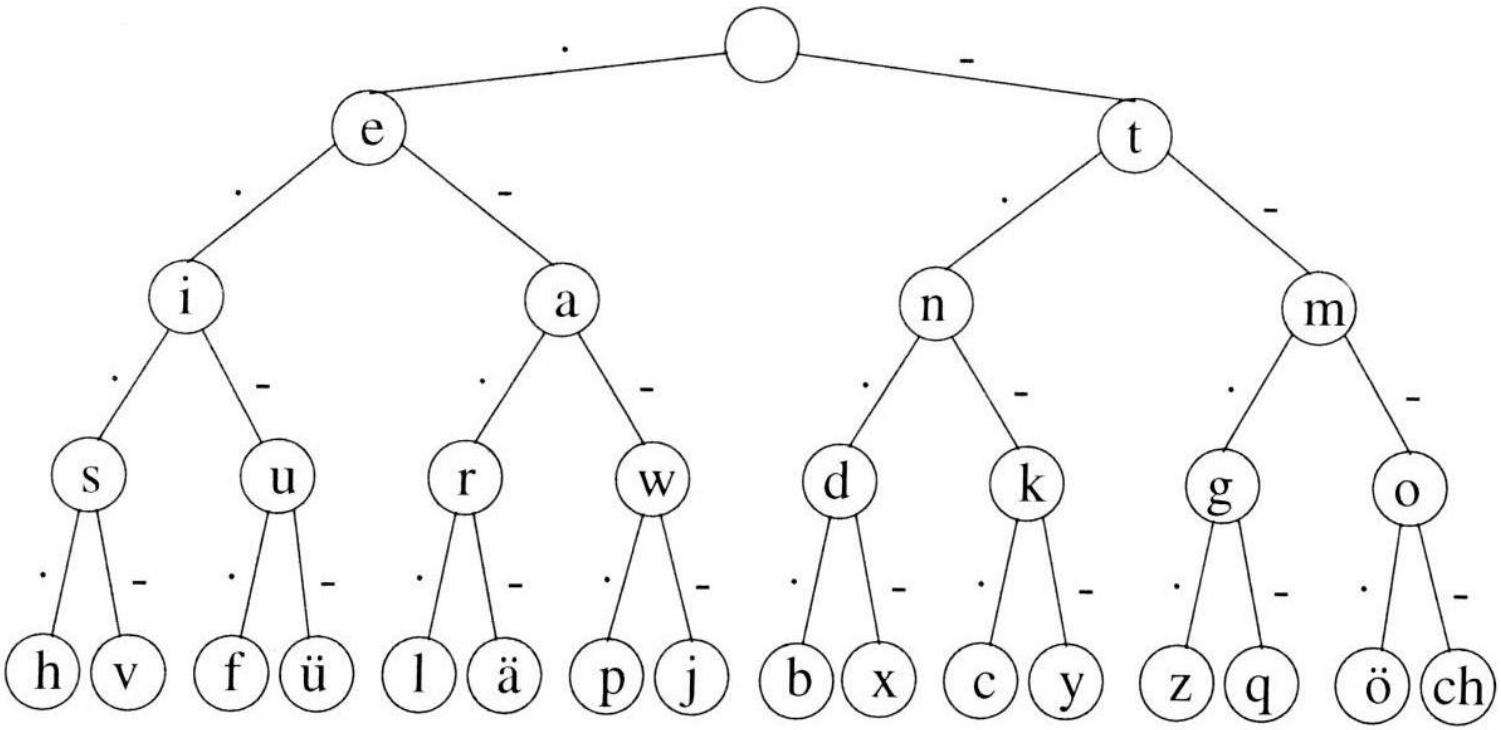
Einfache Umschlüsselung des Morse-Codes für binäre Nachrichtenübertragung:

- `110` als Umschlüsselung des langen Tons –
- `10` als Umschlüsselung des kurzen Tons .
- `0` als Trennzeichen zwischen Morse-Codewörtern

- der neue Code erfüllt die Fano-Bedingung
- jetzt eindeutig decodierbar: 101010011011011001010100 (SOS)

- viele andere Umschlüsselungen möglich, z.B. `1` für kurzen Ton, `01` für langen Ton, `00` für Trennzeichen

Morse-Code: Codebaum (Ausschnitt)



- Symbole als Knoten (!) oder Blätter
- Codewort am Pfad von Wurzel zum Blatt ablesen

Codierung nach Fano

- Ordnung der Urwörter anhand ihrer Wahrscheinlichkeiten $p(a_1) \geq p(a_2) \geq \cdots \geq p(a_n)$
- Einteilung der geordeneten Urwörter in zwei Gruppen mit möglichst gleicher Gesamtwahrscheinlichkeit. Die erste Gruppe bekommt als erste Codewortstelle eine 0, die zweite Gruppe eine 1 (oder umgekehrt)
- Die Teilgruppen werden wiederum entsprechend geteilt, und den oberen Hälften wieder eine 0, den unteren eine 1 als nächste Codewortstelle zugeordnet.
- Aufteilung wird wiederholt, bis jede Teilgruppe nur noch ein Element enthält.
- vorteilhafter, je größer die Anzahl der Urwörter

Codierung nach Fano: Beispiel

- Urbildmenge: $\{A, B, C, D\}$ und zugehörige Wahrscheinlichkeiten $\{0.45, 0.1, 0.15, 0.3\}$

- Sortierung nach Wahrscheinlichkeiten ergibt $\{A, D, C, B\}$
  Erste Gruppenaufteilung ergibt $\{A\}$ und $\{D, C, B\}$
  Codierung von A mit 0 und den anderen Symbolen als $1*$
  Weitere Teilung ergibt $\{D\}$, und $\{C, B\}$
  Letzte Teilung ergibt $\{C\}$ und $\{B\}$
  Codewörter sind $A = 0$, $D = 10$, $C = 110$ und $B = 111$

- mittlere Codewortlänge:
  $L = 0.45 \cdot 1 + 0.3 \cdot 2 + 0.15 \cdot 3 + 0.1 \cdot 3 = 1.8$
- zum Vergleich: Blockcode mit 2 Bits benötigt $L = 2$

Codierung nach Fano: Deutsche Großbuchstaben

Buchstabe a_i	Wahrscheinlichkeit $p(a_i)$	Code (Fano)	Bits
Leerzeichen	0.15149	000	3
E	0.14700	001	3
N	0.08835	010	3
R	0.06858	0110	4
I	0.06377	0111	4
S	0.05388	1000	4
...	...	...	...
Ö	0.00255	111111110	9
J	0.00165	1111111110	10
Y	0.00017	11111111110	11
Q	0.00015	111111111110	12
X	0.00013	111111111111	12

(Fano-Code der Buchstaben der deutschen Sprache, Ameling 1992a)

Codierung nach Huffman

- Ordnung der Urwörter anhand ihrer Wahrscheinlichkeiten $p(a_1) \geq p(a_2) \geq \cdots \geq p(a_n)$

- in jedem Schritt werden jeweils zwei Urwörter mit der geringsten Wahrscheinlichkeit zusammengefasst und durch ein neues Urwort ersetzt
- das Verfahren wird wiederholt, bis eine Menge mit nur noch zwei Urwörtern resultiert
- rekursive Codierung als Baum (links 0 rechts 1)

- ergibt die kleinstmöglichen mittleren Codewortlängen
- Abweichungen zum Verfahren nach Fano sind aber gering
- vielfältiger Einsatz (u.a. bei JPEG, MPEG, ...)

Codierung nach Huffman: Beispiel

- Urbildmenge: $\{A, B, C, D\}$ und zugehörige Wahrscheinlichkeiten $\{0.45, 0.1, 0.15, 0.3\}$

- Zusammenfassen von B und C als neues Wort E, Wahrscheinlichkeit von E ist dann $p(E) = 0.1 + 01.5 = 0.25$
- Zusammenfassen von D und E als neues Wort F mit $p(F) = 0.55$

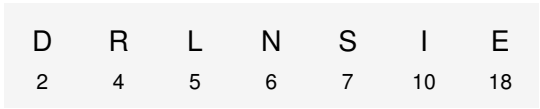
- Zuordnung der Bits entsprechend der Wahrscheinlichkeiten: $F = 0$ und $A = 1$, dann Split von F in $D = 00$ und $E = 01$, schließlich Split von E in $C = 010$ und $B = 011$
- Resultat ist $A = 1$, $D = 00$, $C = 010$ und $B = 011$

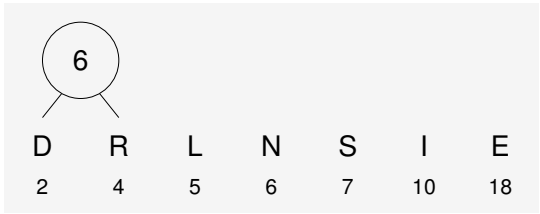
Bildung eines Huffman-Baumes (1)

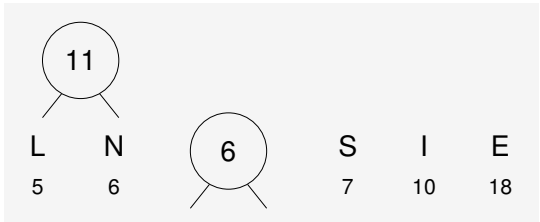
- Alphabet $= \{E, I, N, S, D, L, R\}$
- relative Häufigkeiten
  $E = 18, I = 10, N = 6, S = 7, D = 2, L = 5, R = 4$

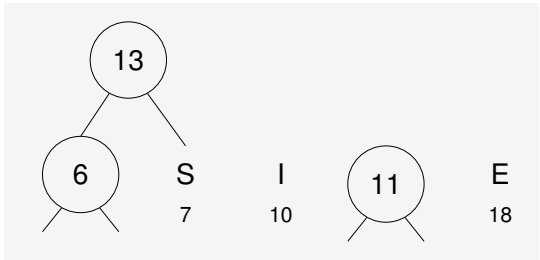
- Sortieren anhand der Häufigkeiten
- Gruppierung (rekursiv)
- Aufbau des Codebaums
- Ablesen der Codebits

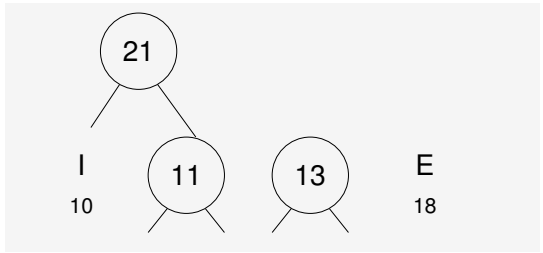
Bildung eines Huffman-Baumes (2)

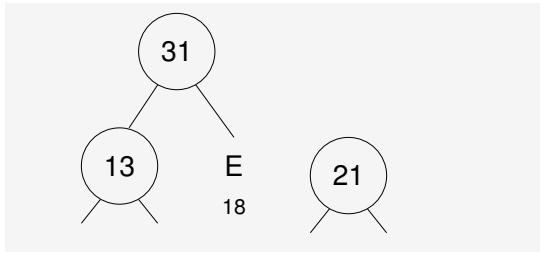


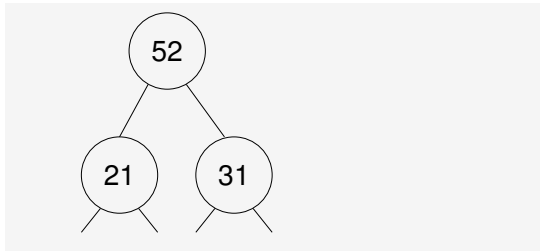




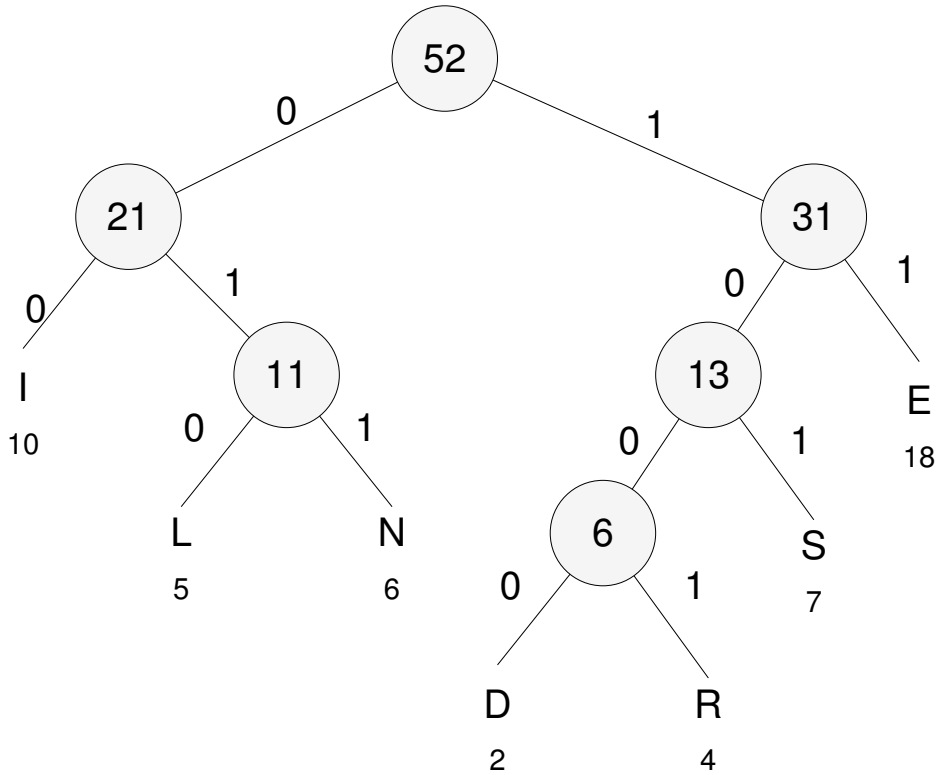








Bildung eines Huffman-Baumes (3)



I	00
L	010
N	011
D	1000
R	1001
S	101
E	11

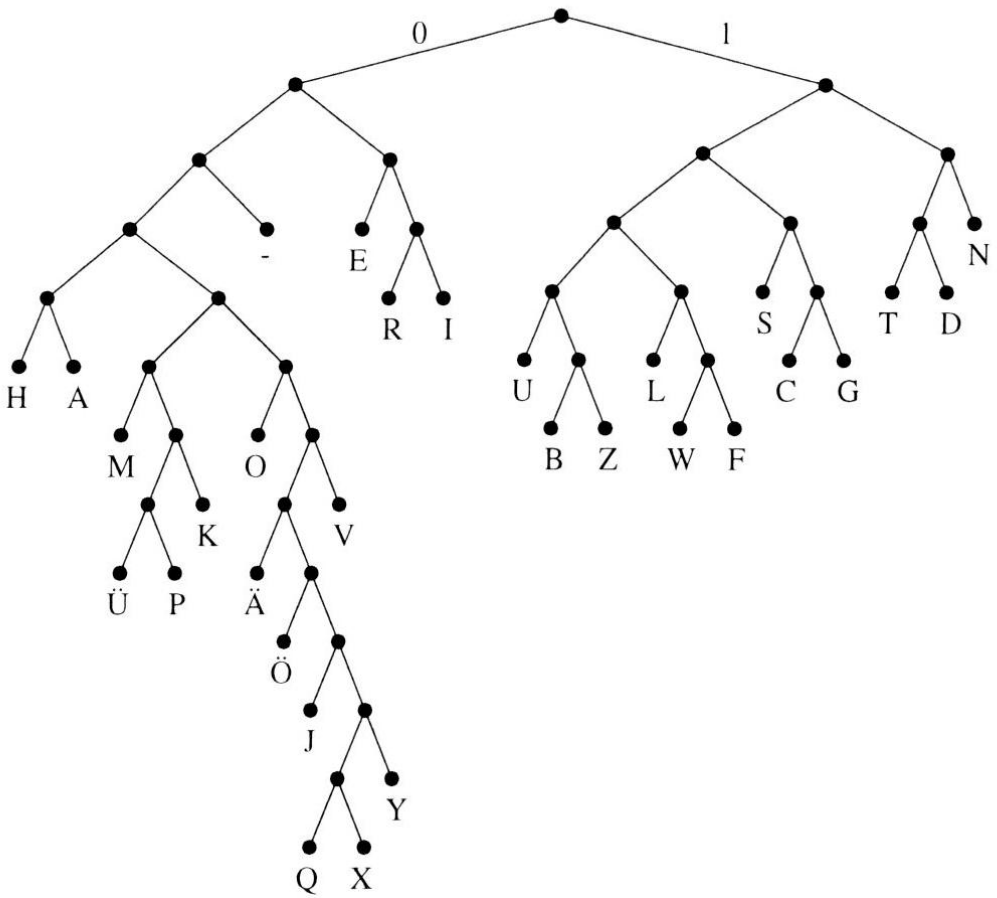
1001 00 11 101 11

R I E S E

Codierung nach Huffman: Deutsche Großbuchstaben

Zeichen	Code	Zeichen	Code
–	001	O	000110
E	010	B	100010
N	111	Z	100011
R	0110	W	100110
I	0111	F	100111
S	1010	K	0001011
T	1100	V	0001111
D	1101	Ü	00010100
H	00000	P	00010101
A	00001	Ä	00011100
U	10000	Ö	000111010
L	10010	J	0001110110
C	10110	Y	00011101111
G	10111	Q	000111011100
M	000100	X	000111011101

Codierung nach Huffman: Codebaum



(ca. 4.5 Bits/Zeichen, 1.7-Mal besser als ASCII)

Codierung nach Huffman: Minimale Codelänge

- Sei C ein Huffman-Code mit durchschnittlicher Codelänge L.
- Sei D ein weiterer Präfix-Code mit durchschnittlicher Codelänge M, mit $M < L$ und M minimal

- Berechne die C und D zugeordneten Decodierbäume A und B
- Betrachte die beiden Endknoten für Symbole kleinster Wahrscheinlichkeit: Weise dem Vorgängerknoten das Gewicht $p_{s-1} + p_s$ zu, streiche die Endknoten. Codelänge reduziert sich um $p_{s-1} + p_s$.

Fortsetzung führt dazu, dass Baum C sich auf Baum mit durchschnittlicher Länge 1 reduziert, und D auf Länge < 1. Dies ist aber nicht möglich.

Codierung nach Huffman: Symbole mit $p \geq 0.5$

- Was passiert, wenn ein Symbol eine Häufigkeit $p_0 \geq 0.5$ aufweist?
- die Huffman-Codierung müsste weniger als ein Bit zuordnen, dies ist jedoch nicht möglich.

- Huffman- (und Fano-) Codierung ist in diesem Fall ineffizient

- Beispiel: Codierung eines Bildes mit einheitlicher Hintergrundfarbe
- andere Ideen notwendig: Lauflängenkodierung (Fax, GIF, PNG), Cosinustransformation (JPEG), usw.

Dynamic Huffman Coding

was tun, wenn

- die Symbolhäufigkeiten nicht vorab bekannt sind?
- die Symbolhäufigkeiten sich ändern können?

Dynamic Huffman Coding (Knuth 1985)

- Encoder protokolliert die (bisherigen) Symbolhäufigkeiten
- Codebaum wird dynamisch aufgebaut und ggf. umgebaut

- Decoder arbeitet entsprechend: Codebaum wird mit jedem dekodierten Zeichen angepasst
- Symbolhäufigkeiten werden nicht explizit übertragen

(D.E. Knuth, Dynamic Huffman Coding, J.of Algorithms 6, 163–180, 1985)

Kraft-Ungleichung (1949)

- Eine notwendige und hinreichende Bedingung für die Existenz eines eindeutig decodierbaren s-elementigen Codes C mit Codelängen $l_1 \leq l_2 \leq l_3 \leq \cdots \leq l_s$ über einem q-nären Zeichenvorrat F ist:

$$\sum_{i=1}^{s} \frac{1}{q^{l_i}} \leq 1$$

Beispiel: $\{1, 00, 01, 11\}$ ist nicht eindeutig decodierbar, denn $\frac{1}{2} + 3 \cdot \frac{1}{4} = 1.25 > 1$

Kraft-Ungleichung: Beispiel

Sei $F = \{0, 1, 2\}$ (ternäres Alphabet)

Seien die geforderten Längen der Codewörter 1,2,2,2,2,2,3,3,3

Einsetzen in die Ungleichung: $\frac{1}{3} + 5 \cdot \frac{1}{3^2} + 3 \cdot \frac{1}{3^3} = 1$

Also existiert ein passender Präfixcode. Konstruktion entsprechend des Beweises:

0 10 11 12 20 21 220 221 222

Kraft-Ungleichung: Beweis

Sei $l_s = m$ und seien u_i die Zahl der Codewörter der Länge i

- Wir schreiben

$$\sum_{i=1}^{s} \frac{1}{q^{l_i}} = \sum_{j=1}^{m} \frac{u_j}{q^j} = \frac{1}{q^m} \sum_{j=1}^{m} u_j \cdot q^{m-j} \leq 1$$

$$u_m + \sum_{j=1}^{m} u_j \cdot q^{m_j} \leq q^m \quad (*)$$

- Jedes Codewort der Länge i „verbraucht“ q^{m-i} Wörter aus F^m
- Summe auf der linken Seite von (*) ist die Zahl der durch den Code C benutzten Wörter von F^m
- Wenn C die Präfix-Bedingung erfüllt, gilt (*)

Informationstheorie

- Informationsbegriff
- Maß für die Information?
- Entropie
- Kanalkapazität

Informationsbegriff

- n mögliche sich gegenseitig ausschliessende Ereignisse A_i
- die zufällig nacheinander mit Wahrscheinlichkeiten p_i eintreten
- stochastisches Modell $W\{A_i\} = p_i$

- angewendet auf Informationsübertragung: das Symbol a_i wird mit Wahrscheinlichkeit p_i empfangen

- Beispiel: $p_i = 1$ und $p_j = 0 \quad \forall j \neq i$
- dann wird mit Sicherheit das Symbol A_i empfangen
- der Empfang bringt keinen Informationsgewinn

- Informationsgewinn („Überraschung“) größer, je kleiner p_j

Geeignetes Maß für die Information?

- Wir erhalten die Nachricht A mit Wahrscheinlichkeit p_A und anschließend die unabhängige Nachricht B mit Wahrscheinlichkeit p_B

- Wegen der Unabhängigkeit ist die Wahrscheinlichkeit beider Ereignisse gegeben durch das Produkt $p_A \cdot p_B$.

- Informationsgewinn („Überraschung“) größer, je kleiner p_j
- Wahl von $1/p$ als Maß für den Informationsgewinn?
- möglich, aber dann wäre der Gesamtinformationsgehalt von zwei (mehreren) Ereignissen gleich dem Produkt der einzelnen Informationsgehalte
- additive Größe wäre besser: Logarithmus von $1/p$ bilden

Erinnerung: Logarithmus

- Umkehrfunktion zur Exponentialfunktion
- formal: Logarithmus ist Lösung der Gleichung $a = b^x$ für gegebenes a und b
- falls die Lösung existiert: $x = \log_b(a)$

Beispiel: $3 = \log_2(8), \quad$ denn $2^3 = 8$

$log(x \cdot y) = log(x) + log(y)$

$b^{\log_b(x)} = x \quad$ und $\quad \log_b(b^x) = x$

$\log_b(x) = \frac{\log_a(x)}{log_a(b)}$

$\log_2(x) = \log(x)/\log(2) = \log(x)/0.693141718$

Exkurs: Binärer Logarithmus

- $\log_2(x) = 0.b_1 b_2 b_3 \cdots = \sum_{k>0} b_k 2^{-k}$ mit $b_k \in \{0, 1\}$
- $\log_2(x^2) = b_1.b_2 b_3 \ldots$ wegen $\log(x^2) = 2\log(x)$

```
INPUT:  1 < x < 2  (ggf. vorher skalieren)
OUTPUT: Nachkommastellen bi der Binärdarstellung von log2(x)
i = 0
LOOP
   i = i + 1
   x = x*x
   IF (x >= 2) THEN
      x = x / 2
      bi = 1
   ELSE
      bi = 0
   END IF
END LOOP
```

Definition: Informationsgehalt

Informationsgehalt eines Ereignisses A_i mit Wahrscheinlichkeit p_i?

- als meßbare und daher additive Größe
- durch Logarithmierung (Basis 2) der Wahrscheinlichkeit:

$$I(A_i) = \log_2(\frac{1}{p_i}) = -\log_2(p_i)$$

- **Informationsgehalt** I (oder Information) von A_i
- auch **Entscheidungsgehalt** genannt

- im Beispiel mit zwei Nachrichten A und B
- $I(A) + I(B) = \log_2(\frac{1}{p_A \cdot p_B}) = \log_2 \frac{1}{p_A} + \log_2 \frac{1}{p_B}$

Informationsgehalt: Einheit Bit

$$I(A_i) = \log_2(\tfrac{1}{p_i}) = -\log_2(p_i)$$

- Wert von I ist eine reelle Größe
- gemessen in der Einheit **1 Bit**

- Beispiel: nur zwei mögliche Symbole 0 und 1 mit gleichen Wahrscheinlichkeiten $p_0 = p_1 = \frac{1}{2}$.

  Der Informationsgehalt des Empfangs einer 0 oder 1 ist dann

  $I(0) = I(1) = \log_2(1/\frac{1}{2}) = 1\,\text{Bit}$

- Achtung: die Einheit Bit nicht verwechseln mit Binärstellen oder den Symbolen 0 und 1

Ungewissheit, Überraschung, Information

Vor dem Empfang einer Nachricht gibt es *Ungewissheit* über das Kommende. Beim Empfang gibt es die *Überraschung*. Und danach hat man den Gewinn an *Information*.

- Alle drei Begriffe in der oben definierten Einheit *Bit* messen.
- Diese Quantifizierung der *Information* ist zugeschnitten auf die Nachrichtentechnik.

- umfasst nur einen Aspekt des umgangssprachlichen Begriffs *Information*

Informationsgehalt: Beispiel Meteorit

- die Wahrscheinlichkeit, an einem Tag von einem Meteor getroffen zu werden, sei $p_M = 10^{-16}$
- Kein Grund zur Sorge, weil die Ungewissheit von $I = \log_2(1/(1 - p_M)) \approx 3.2 \cdot 10^{-16}$ sehr klein ist.

  Ebenso klein ist die Überraschung, wenn das Unglück nicht passiert. Informationsgehalt der Nachricht „Ich wurde nicht vom Meteor erschlagen“ ist sehr klein
- Umgekehrt wäre die Überraschung groß: $\log_2(1/p_M) = 53.15$

Beispiel: Würfeln

- bei vielen Spielen hat die 6 eine besondere Bedeutung
- hier betrachten wir aber zunächt nur die Wahrscheinlichkeit von Ereignissen, nicht deren Semantik

- die Wahrscheinlichkeit, eine 6 zu würfeln, ist $1/6$
- $I(6) = -\log_2(\frac{1}{6}) = 2.585$

Beispiel: Information eines Buches

- Gegeben seien zwei Bücher mit je 200 Seiten zu je 40 Zeilen von je 72 Zeichen:
  a) deutscher Text
  b) mit Zufallsgenerator mit Gleichverteilung aus Alphabet mit 80-Zeichen erzeugt
- Informationsgehalt in beiden Fällen?

- Im deutschen Text abhängig vom Kontext. Beispiel: Empfangen wir als deutschen Text `"Der Begrif"` so ist `f` als nächstes Symbol sehr wahrscheinlich.
- beim Zufallstext liefert jedes neue Symbol die zusätzliche Information $I = \log_2(1/(1/80))$
- der Zufallstext enthält die größtmögliche Information

Beispiel: Einzelner Buchstabe

- die Wahrscheinlichkeit, in einem Text an einer gegebenen Stelle das Zeichen `A` anzutreffen sei $W\{A\} = p = 0.01$.

- Informationsgehalt $I(A) = \log_2(1/0.01) = 6.6439$
- wenn der Text in ISO-8859-1 codiert vorliegt, werden 8 Binärstellen zur Repräsentation des `A` benutzt
- der Informationsgehalt ist jedoch geringer

Entropie

obige Definition der Information lässt sich nur jeweils auf den Empfang eines speziellen Zeichens anwenden

- **durchschnittliche Information** bei Empfang eines Symbols?
- diesen Erwartungswert bezeichnet man als **Entropie** des Systems
- Wahrscheinlichkeiten aller möglichen Ereignisse A_i seien $W\{A_i\} = p_i$
- da jeweils eines der möglichen Symbole eintrifft, gilt $\sum_i p_i = 1$
- Dann berechnet sich die Entropie als Erwartungswert

$$H = E\{I(A_i)\} = \sum_i p_i \cdot I(A_i) = \sum_i p_i \log_2(\frac{1}{p_i}) = -\sum_i p_i \log_2(p_i)$$

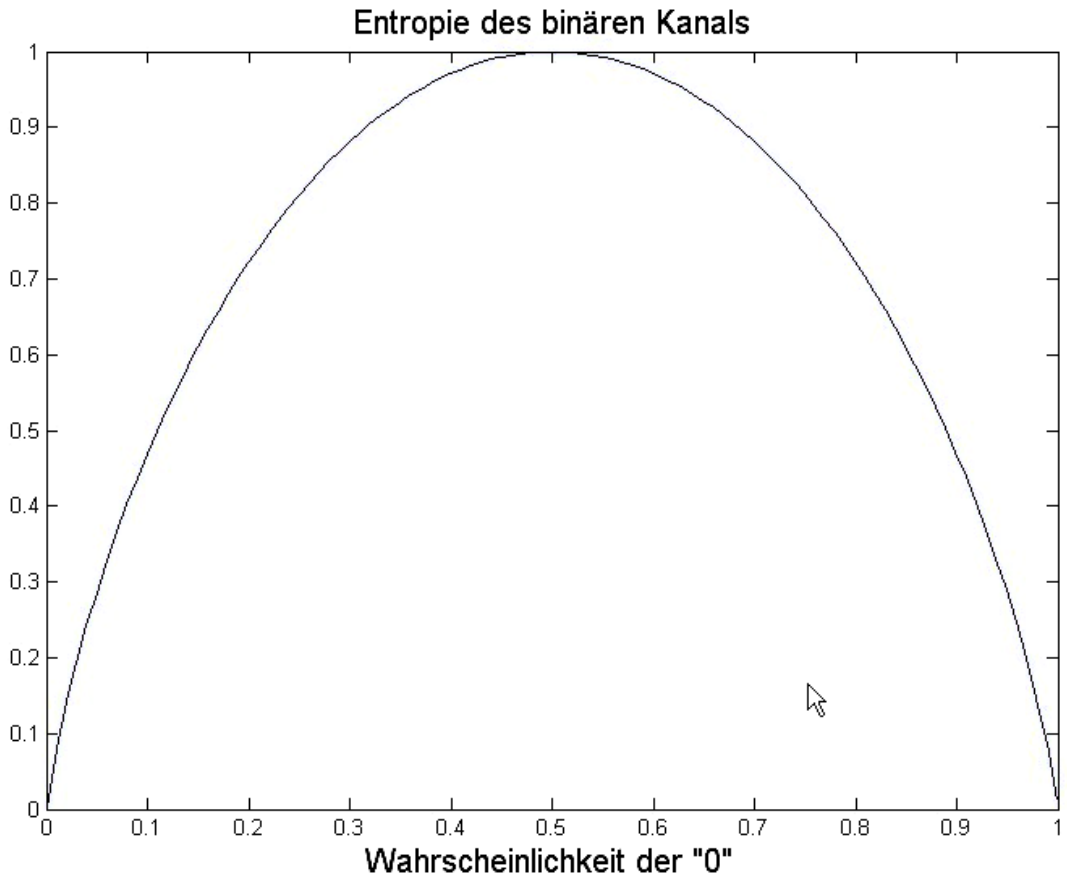
- als Funktion der Symbol-Wahrscheinlichkeiten, also nur abhängig vom stochastischen Modell

Entropie: Beispiele

- drei mögliche Ereignisse mit Wahrscheinlichkeiten $\{\frac{1}{2}, \frac{1}{3}, \frac{1}{6}\}$
- dann berechnet sich die Entropie zu
  $H = -(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{3}\log_2\frac{1}{3} + \frac{1}{6}\log_2\frac{1}{6}) = 1.4591$

- Empfang eines Binärstelle mit den Wahrscheinlichkeiten
  $p_0 = q$ und $p_1 = (1 - q)$.
- für $q = \frac{1}{2}$ erhält man
  $H = -(\frac{1}{2}\log_2\frac{1}{2} + (1 - \frac{1}{2})\log_2(1 - \frac{1}{2}) = 1.0$
- mittlerer Informationsgehalt beim Empfang einer Binärstelle mit gleicher Wahrscheinlichkeit für beide Symbole ist genau 1 Bit

Entropie: Diagramm



Entropie bei Empfang eines Binärstelle mit den Wahrscheinlichkeiten $p_0 = q$ und $p_1 = (1 - q)$.

Entropie: Gleichverteilte Symbole

- mittlerer Informationsgehalt einer Binärstelle nur dann 1 Bit, wenn beide möglichen Symbole gleich wahrscheinlich

- entsprechendes gilt auch für größere Symbolmengen
- Beispiel: 256 Symbole (8-bit Bytes), gleich wahrscheinlich:
  $H = \sum_i p_i \log_2(1/p_i) = 256 \cdot (1/256) \cdot \log_2(1(1/256)) = 8\,\text{Bit}$

- **Redundanz**: die Differenz zwischen dem aufgrund der Symbole (z.B. Wortlängen) möglichen und dem tatsächlich genutzten Informationsinhalt

Entropie: Einige Eigenschaften

E1 $H(p_1, p_2, \ldots, p_n)$ maximal, falls $p_i = 1/n \quad (1 \leq i \leq n)$

E2 H ist symmetrisch: für jede Permutation π von $1, 2, \ldots, n$ gilt:
$H(p_1, p_2, \ldots, p_n) = H(p_{\pi(1)}, p_{\pi(2)}, \ldots, p_{\pi(n)})$

E3 $H(p_1, p_2, \ldots, p_n) \geq 0 \quad$ mit $H(0, \ldots, 0, 1, \ldots, 0) = 0$

E4 $H(p_1, p_2, \ldots, p_n, 0) = H(p_1, p_2, \ldots, p_n)$

E5 $H(1/n, 1/n, \ldots, 1/n) \leq H(1/(n+1), 1/(n+1), \ldots, 1/(n+1))$

E6 H ist stetig in seinen Argumenten

E7 Additivität: seien $n, m \in N^+$:
$H(\frac{1}{n \cdot m}, \frac{1}{n \cdot m}, \ldots, \frac{1}{n \cdot m}) = H(\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n}) + H(\frac{1}{m}, \frac{1}{m}, \ldots, \frac{1}{m})$

Redundanz

- **Redundanz** (engl. *code redundancy*): die Differenz zwischen dem aufgrund der Symbole (z.B. Wortlängen) möglichen und dem tatsächlich genutzten Informationsinhalt, $R = H_0 - H$
- Relative Redundanz: $r = \frac{H_0 - H}{H_0}$

- für binäre Blockcodes mit Wortlänge N bits: $H_0 = N$
- gegebener Code mit N Wörtern a_i und $p(a_i)$:

$$R = H_0 - H = H_0 - \left(- \sum_{i=1}^{N} p(a_i) \log_2 p(a_i) \right)$$

$$= N + \sum_{i=1}^{N} p(a_i) \log_2 p(a_i)$$

Kanalkapazität

Informationstheorie ursprünglich entwickelt zur:

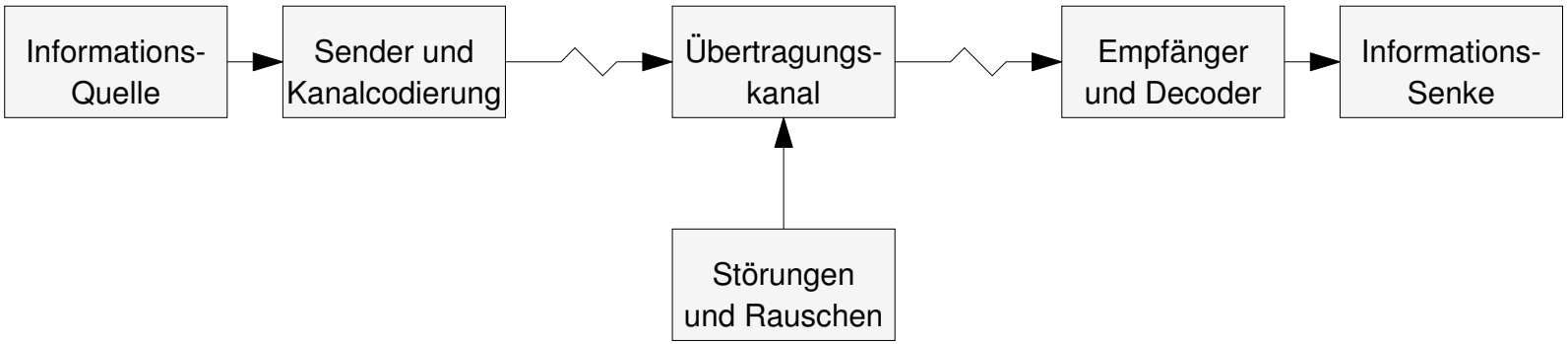
- formalen Behandlung der Übertragung von Information
- über reale nicht fehlerfreie Kanäle
- deren Verhalten als stochastisches Modell formuliert werden kann

- zentrales Resultat ist die **Kanalkapazität** C des **binären symmetrischen Kanals**
- d.h. der maximal pro Binärstelle übertragbare Informationsgehalt

$$C = 1 - H(F)$$

mit $H(F)$ der Entropie des Fehlerverhaltens

Erinnerung: Modell der Informationsübertragung

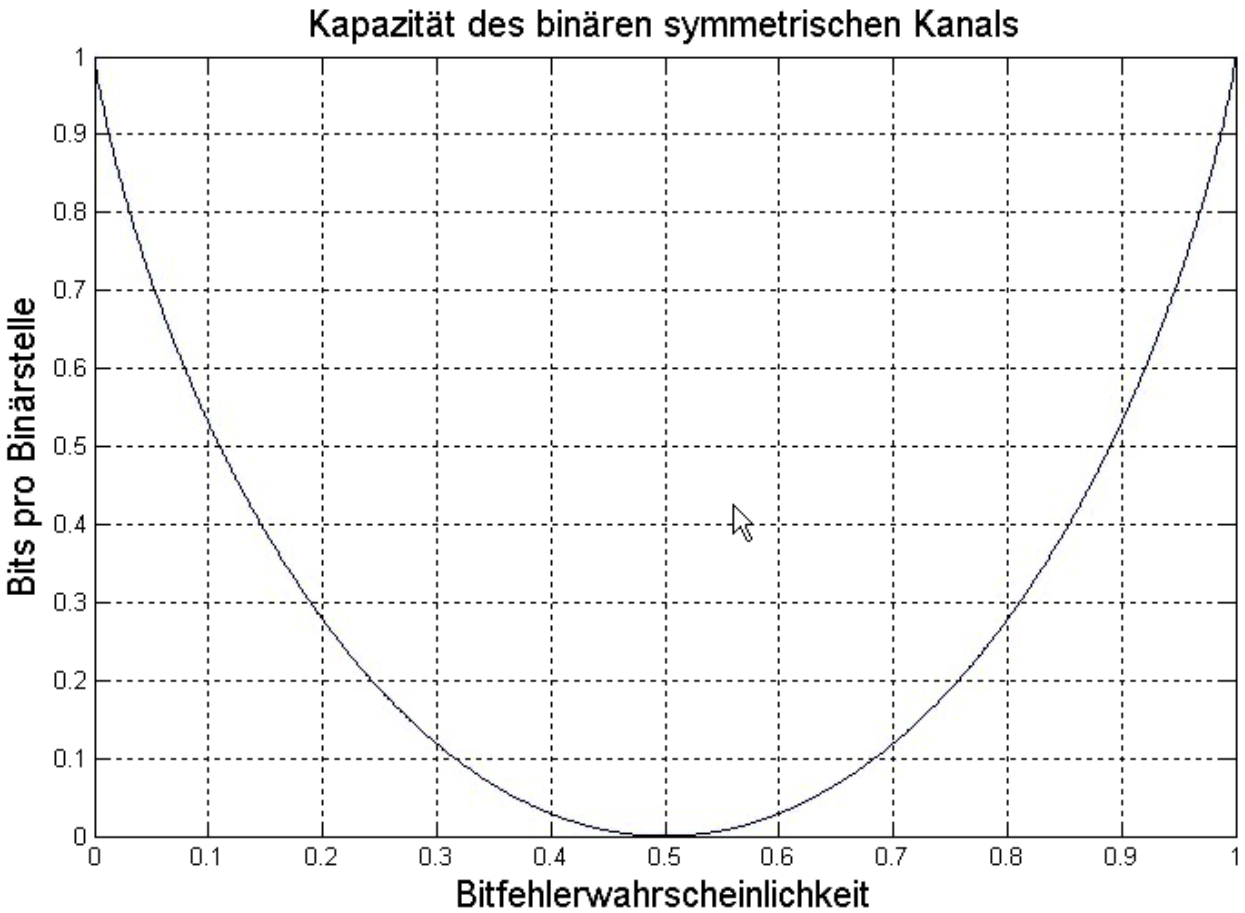


- Informationsquelle
- Sender mit möglichst effizienter Kanalcodierung
- gestörter und verrauschter Übertragungskanal
- Empfänger mit Decodierer und Fehlererkennung/-korrektur
- Informationssenke und -verarbeitung

Binärer symmetrischer Kanal

- Wahrscheinlichkeit der beiden Symbole 0 und 1 ist gleich
- Wahrscheinlichkeit P, dass bei Übertragungsfehlern aus einer 0 eine 1 wird, gleich der Wahrscheinlichkeit, dass aus einer 1 eine 0 wird
- Wahrscheinlichkeit eines Fehlers an Binärstelle i ist unabhängig vom Auftreten eines Fehlers an anderen Stellen
- Dann ist $H(F) = P \cdot \log_2(1/P) + (1 - P) \cdot \log_2(1/(1 - P))$
- Kanalkapazität ist $C = 1 - H(F)$

Kanalkapazität



Kanalkapazität

- bei $P = 0.5$ ist die Kanalkapazität $C = 0$
- der Empfänger kann die empfangenen Daten nicht von einer zufälligen Sequenz unterscheiden

- bei $P > 0.5$ steigt die Kapazität wieder an

  (rein akademischer Fall: Invertieren aller Bits)

- die Kanalkapazität ist eine obere Schranke
- wird in der Praxis nicht erreicht
- Theorie liefert keine Hinweise, wie die fehlerfreie Übertragung praktisch durchgeführt werden kann

Shannon-Theorem (1949)

- Gegeben: binärer symmetrischer Kanal mit Störwahrscheinlichkeit P und Kapazität $C(P)$
- **Theorem**: falls die Übertragungsrate R kleiner als $C(P)$ ist, findet man zu jedem $\epsilon > 0$ einen Code $\mathcal{C}$ mit Übertragungsrate $R(\mathcal{C})$ und $C(P) \geq R(\mathcal{C}) \geq R$ und Fehlerdekodierwahrscheinlichkeit $< \epsilon$

- leider liefert die Theorie keine Ideen zur Realisierung
- die Nachrichten müssen sehr lang sein
- der Code muss im Mittel sehr viele Fehler in jeder Nachricht korrigieren
- mittlerweile sehr nah am Limit (Turbo-Codes, LDPC, usw.)

C. E. Shannon: Communication in the Presence of Noise; Proc. IRE, Vol.37, No.1, 1949

Fehlererkenennde / -korrigierende Codes

Informationstheorie

Kanalkapazität

Shannon-Theorem

- zuverlässige Datenübertragung ist möglich
- aber (bisher) keine Ideen für die Realisierung

- fehlererkennende Codes
- fehlerkorrigierende Codes

Fehlertypen

diverse mögliche Fehler bei der Datenübertragung

- Verwechslung eines Zeichens $a \rightarrow b$
- Vertauschen benachbarter Zeichen $ab \rightarrow ba$
- Vertauschen entfernter Zeichen $abc \rightarrow cba$
- Zwillings-/Bündelfehler $aa \rightarrow bb$
- usw.

- abhängig von der Technologie / Art der Übertragung
  - Bündelfehler durch Kratzer auf einer CD
  - Bündelfehler bei Funk durch längere Störimpulse
  - Buchstabendreher beim „Eintippen“ eines Textes

Begriffe: zur Fehlerbehandlung

- **Block-Code**: k-Informationsbits werden in n-Bits codiert
- **Faltungscodes**: ein Bitstrom wird in einen Codebitstrom höherer Bitrate codiert

- **linearer** (n, k)**-Code**: ein k-dimensionaler Unterraum des $GF(2)^n$
- **modifizierter Code**: eine oder mehrere Stellen eines linearer Codes werden systematisch verändert (d.h. im $GF(2)$ invertiert). Null- und Einsvektor gehören nicht mehr zum Code.
- **nichtlinearer Code**: weder linear noch modifiziert.

Begriffe: $GF(2)$, $GF(2)^n$

Boole'sche Algebra:

- basiert auf UND, ODER, Negation
- UND $\approx$ Multiplikation, ODER $\approx$ Addition
- aber: kein inverses Element für die ODER-Operation

Galois-Field mit zwei Elementen: GF(2)

- Körper, zwei Verknüpfungen: UND und XOR
- XOR als Addition `mod 2`
- UND als Multiplikation
- additives Inverses existiert: $x \oplus x = 0$

(Details: Mathe-Skript, Wikipedia, vdHeide: Technische Informatik 1)

Begriffe: zur Fehlerbehandlung

- **systematischer Code**: wenn die zu codierende Information direkt (als Substring) im Codewort enthalten ist

- **zyklischer Code:** ein Block-Code heisst zyklisch, wenn mit jedem Codewort auch sämtliche zyklischen Verschiebungen (Rotationen, z.b. rotate-left) des Codewortes auch zum Code gehören. Bei serieller Übertragung erlaubt dies die Erkennung/Korrektur von Bündelfehlern.

Begriffe: ARQ und FEC

- **Automatic Repeat Request** (ARQ): der Empfänger erkennt ein fehlerhaftes Symbol und fordert dies vom Sender erneut an
  - bidirektionale Kommunikation erforderlich
  - unpraktisch bei großer Entfernung / Echtzeitanforderungen

- **Vorwärtsfehlerkorrektur** (*Forward Error Correction*, FEC): die übertragene Information wird durch zusätzliche Redundanz (z.B. Prüfziffern) gesichert. Der Empfänger erkennt fehlerhafte Codewörter und kann diese selbständig korrigieren.

- je nach Einsatzzweck beide Verfahren üblich
- auch kombiniert

Hamming-Abstand

- **Hamming-Abstand**: die Anzahl der Stellen, an denen sich zwei Binärcodewörter der Länge w unterscheiden
- **Hamming-Gewicht**: Hamming-Abstand eines Codewortes vom Null-Wort

$a = 0110\ 0011$

$b = 1010\ 0111$

- Hamming-Abstand von a und b ist 3
- Hamming-Gewicht von b ist 5

- Java: `Integer.bitcount( a ^ b )`

Fehlererkennende und -korrigierende Codes

- Zur *Fehlererkennung* und *Fehlerkorrektur* ist eine Codierung mit Redundanz erforderlich
- Repräsentation enthält mehr Bits, als zur reinen Speicherung nötig wären

- Codewörter so wählen, dass sie paarweise mindestens den Hamming-Abstand d haben
- dieser Abstand heißt dann **Minimalabstand** d

- Fehlererkennung bis zu $(d-1)$ fehlerhaften Stellen
- Fehlerkorrektur bis zu $((d-1)/2)$ fehlerhaften Stellen

Prüfinformation

Man fügt den Daten **Prüfinformation** hinzu, oft **Prüfsumme** genannt

- zur Fehlerentdeckung
- zur Fehlerkorrektur
- zur Korrektur einfacher Fehler, Entdeckung schwerer Fehler

- Prüfziffer, Parität
- Summenbildung
- CRC-Verfahren (*cyclic-redundancy check*)
- BCH-Codes (Bose, Ray-Chauduri, Hocquengham)
- RS-Codes (Reed-Solomon)

Paritätscode

- das Anfügen eines **Paritätsbits** an ein Binärcodewort $z = (z_1, \ldots, z_n)$ ist die einfachste Methode zur Erkennung von Einbitfehlern

- die Parität wird berechnet als

$$p = \left(\sum_{i=1}^{n} z_i \right) \mod 2$$

- **gerade Parität** (*even parity*): $y_{\text{even}} = (z_1, \ldots, z_n, p)$
  dann ist $p(y_{\text{even}}) = (\sum_i y_i) \mod 2 = 0$
- **ungerade Parität** (*odd parity*): $y_{\text{odd}} = (z_1, \ldots, z_n, \overline{p})$
  dann ist $p(y_{\text{odd}}) = (\sum_i y_i) \mod 2 = 1$

Paritätscode: Eigenschaften

- in der Praxis meistens Einsatz der ungeraden Parität:

  pro Codewort y_{odd} mindestens je eine Null und Eins

- Hamming-Abstand zweier Codewörter im Paritätscode ist mindestens 2, weil sich bei Ändern eines Nutzbits jeweils auch die Parität ändert: $d = 2$

- Erkennung von Einbitfehlern möglich:

  Berechnung der Parität im Empfänger und Vergleich mit der erwarteten Parität

  außerdem: Erkennung von (ungeraden) Mehrbitfehlern

Zweidimensionale Parität

- Anordnung der Daten (=Informations)-Bits als Matrix
- Berechnung der Parität für alle Zeilen und Spalten
- optional auch für Zeile/Spalte der Paritäten

- entdeckt 1-bit Fehler in allen Zeilen und Spalten
- erlaubt Korrektur von allen 1-bit und vielen n-bit Fehlern

- natürlich auch weitere Dimensionen möglich. Am besten n-dimensionale Anordnung der Daten und Berechnung von n Paritätsbits

Zweidimensionale Parität: Beispiel

```
H   100 1000 | 0          100 1000 | 0
A   100 0001 | 0          100 0101 | 0
M   100 1101 | 0          110 1101 | 0
M   100 1101 | 0          100 1101 | 0
I   100 1001 | 1          000 1001 | 1
N   100 1110 | 0          100 1110 | 0
G   100 0111 | 0          100 0111 | 0
-------------+--          ---------+--
I   100 1001 | 1          100 1000 | 1
```

- 64-Bits pro Symbol, davon 49 für Nutzdaten und 15 für Parität
- links: Beispiel für ein Codewort und Paritätsbits
- rechts: empfangenes Codewort mit vier Einzelfehlern, davon ein Fehler in den Paritätsbits

Zweidimensionale Parität: Einzelfehler

```
H   100 1000 | 0          100 1000 | 0
A   100 0001 | 0          100 0101 | 0
M   100 1101 | 0          100 1101 | 0
M   100 1101 | 0          100 1101 | 0
I   100 1001 | 1          100 1001 | 1
N   100 1110 | 0          100 1110 | 0
G   100 0111 | 0          100 0111 | 0
-------------+--          ---------+--
I   100 1001 | 1          100 1001 | 1
```

- Empfänger berechnet Parität und vergleicht mit gesendeter
- Einzelfehler: Abweichung in je einer Zeile und Spalte
- Fehler kann daher zugeordnet und korrigiert werden
- Mehrfachfehler: nicht alle, aber viele erkennbar (korrigierbar)

Zweidimensionale Parität: Dezimalsystem

Beispiel: Parität als Zeilen/Spaltensumme mod 10 hinzufügen

```
3  7  4
5  4  8
1  3  5
```

Bildung der Zeilen/Spaltensummen, und für gestörte Daten

```
3  7  4 | 4        3  7  4 | 4
5  4  8 | 7        5  4  3 | 2    <-- Fehler
1  3  5 | 9        1  3  5 | 9
---------+--      ---------+--
9  4  7 |          9  4  2 |
                             ^---------- Fehler
```

International Standard Book Number

ISBN-10 (1970), ISBN-13

- Codierung eines Buches als Tupel
  (Sprachraum, Verlag, verlagsinterne Nummer, Prüfziffer)

  Sprachraum als Fano-Code:
  $0-7$, $80-94$, $950-995$, $9960-9989$, $99900-99999$
  Beispiel: Sprachraum $0,1$: Australien, UK, USA, ...
  Beispiel: Sprachraum 3: Österreich, DE, CH

  Verlagsnummer:
  $00-19$ (1 Mio Titel), $20-699$ (100 000 Titel) usw.

Prüfverfahren für ISBN

- gegeben: ISBN-Zahl $(z_9, z_8, \ldots, z_0)$
- Zahl zulässig, genau dann wenn

$$\sum_{i=0}^{n-1} (i \cdot z_i) \mod 11 = 0$$

Berechnung der Prüfziffer als (mod11)-Operationen,

Symbol X steht für Ziffer 10

Beispiel: 3-486-21153-6 ergibt

$3 \cdot 10 + 4 \cdot 9 + 8 \cdot 8 + 6 \cdot 7 + 2 \cdot 6 + 1 \cdot 5 + 1 \cdot 4 + 5 \cdot 3 + 3 \cdot 2 + 6 \cdot 1 = 220$

Prüfung: $220 \mod 11 = 0$

Prüfverfahren für ISBN: Fehlertypen

- Prüfziffer schützt gegen Verfälschung einer Ziffer
- Prüfziffer schützt gegen Vertauschung zweier Ziffern
- Prüfziffer schützt gegen „Falschdopplung“ einer Ziffer

Beispiel: vertausche i-te und j-te Ziffer (mit $i \neq j$)

Prüfsumme (korrekt - falsch):

$= i \cdot z_i + j \cdot z_j - j \cdot z_i - i \cdot z_j = (i - j) \cdot (z_i - z_k)$ mit $z_i \neq z_j$.

(3,1)-Wiederholungscode / (3,1)-Hamming-Code

- dreifache Wiederholung jedes Datenworts
- Generatormatrix ist

$$G = (111)$$

- Codewörter ergeben sich als Multiplikation von G mit dem Informationsvektor u (jeweils ein Bit)

$u = 0: \quad x = (111) \cdot (0) = (000)$

$u = 1: \quad x = (111) \cdot (1) = (111)$

- Dekodierung durch Mehrheitsentscheid: 1-bit Fehlerkorrektur
- Verallgemeinerung als (n, k)-Wiederholungscode
- systematischer Code mit Minimalabstand $D = n$
- Nachteil: geringe Datenrate

Hamming-Code

- Code mit Hamming-Abstand 3
- korrigiert 1-bit Fehler, erkennt (viele) 2-bit und 3-bit Fehler

(N, n)-Hamming-Code

- Datenwort n-bit $(d_1, d_2, \ldots\ d_n)$

  um k-Prüfbits ergänzen $(p_1, p_2, \ldots\ p_k)$

$\Rightarrow$ Codewort mit $N = n + k$ bit

- Fehlerkorrektur gewährleisten: $2^k \geq N + 1$
  - 2^k Kombinationen mit k-Prüfbits
  - 1 fehlerfreier Fall
  - N zu markierende Bitfehler

Hamming-Code: Konstruktion

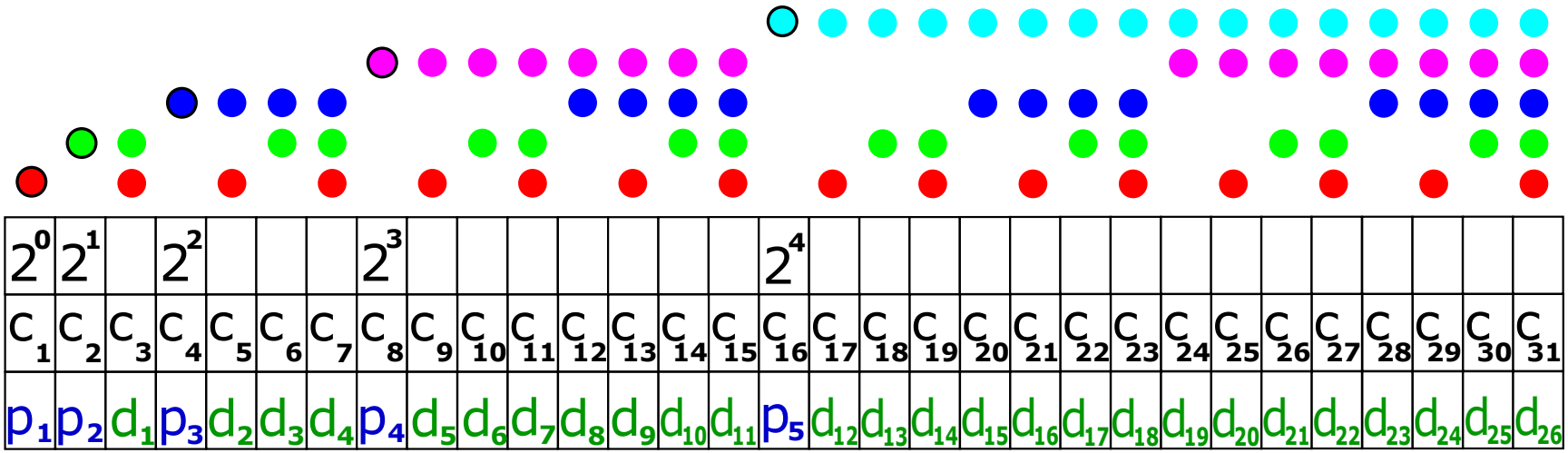
1. bestimme kleinstes k mit $n \leq 2^k - k - 1$
2. Prüfbits an Bitpositionen: $2^0, 2^1, \ldots\ 2^{k-1}$
   Originalbits an den übrigen Positionen

Position	1	2	3	4	5	6	7	8	9	...
Bit	p_1	p_2	d_1	p_3	d_2	d_3	d_4	p_4	d_5	...

3. berechne Prüfbit i als mod 2-Summe der Bits (XOR), deren Positionsnummer ein gesetztes i-bit enthält

$$p_1 = d_1 \oplus d_2 \oplus d_4 \oplus d_5 \oplus \ldots$$
$$p_2 = d_1 \oplus d_3 \oplus d_4 \oplus d_6 \oplus \ldots$$
$$p_3 = d_2 \oplus d_3 \oplus d_4 \oplus d_8 \oplus \ldots$$
$$p_4 = d_5 \oplus d_6 \oplus d_7 \oplus d_8 \oplus \ldots$$
$$\ldots$$

Hamming-Code: Schema für 2...5 Prüfbits



(7,4)-Hamming-Code

- $p_1 = d_1 \oplus d_2 \oplus d_4$
  $p_2 = d_1 \oplus d_3 \oplus d_4$
  $p_3 = d_2 \oplus d_3 \oplus d_4$

(15,11)-Hamming-Code

- $p_1 = d_1 \oplus d_2 \oplus d_4 \oplus d_5 \oplus d_7 \oplus d_9 \oplus d_{11}$
  $p_2 = d_1 \oplus d_3 \oplus d_4 \oplus d_6 \oplus d_7 \oplus d_{10} \oplus d_{11}$
  $p_3 = d_2 \oplus d_3 \oplus d_4 \oplus d_8 \oplus d_9 \oplus d_{10} \oplus d_{11}$
  $p_4 = d_5 \oplus d_6 \oplus d_7 \oplus d_8 \oplus d_9 \oplus d_{10} \oplus d_{11}$

(7,4)-Hamming-Code

- sieben Codebits für je vier Datenbits
- korrigiert 1-bit Fehler, erkennt (viele) 2-bit und 3-bit Fehler
- linearer (7,4)-Block-Code
- Generatormatrix ist

$$G = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Codewort $c = G \cdot d$

(7,4)-Hamming-Code

- Prüfmatrix H orthogonal zu gültigen Codewörtern: $H \cdot c = 0$

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

  für ungültige Codewörter $H \cdot c \neq 0$

$\Rightarrow$ „Fehlersyndrom“ liefert Information über Fehlerposition / -art

Fazit: Hamming-Codes

\+ größere Wortlangen: besseres Verhältnis von Nutz- zu Prüfbits

\+ einfaches Prinzip, einfach decodierbar

– es existieren weit bessere Codes

(7,4)-Hamming-Code: Beispiel

- Codieren von $d = (0, 1, 1, 0)$

$$c = G \cdot d = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

(7,4)-Hamming-Code: Beispiel (cont.)

- Prüfung von Codewort $c = (1, 1, 0, 0, 1, 1, 0)$

$$H \cdot c = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

(7,4)-Hamming-Code: Beispiel (cont.)

- im Fehlerfall $c = (1, 1, 1, 0, 1, 1, 0)$

$$H \cdot c = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$$

$\Rightarrow$ Fehlerstelle:

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Fehlerrate

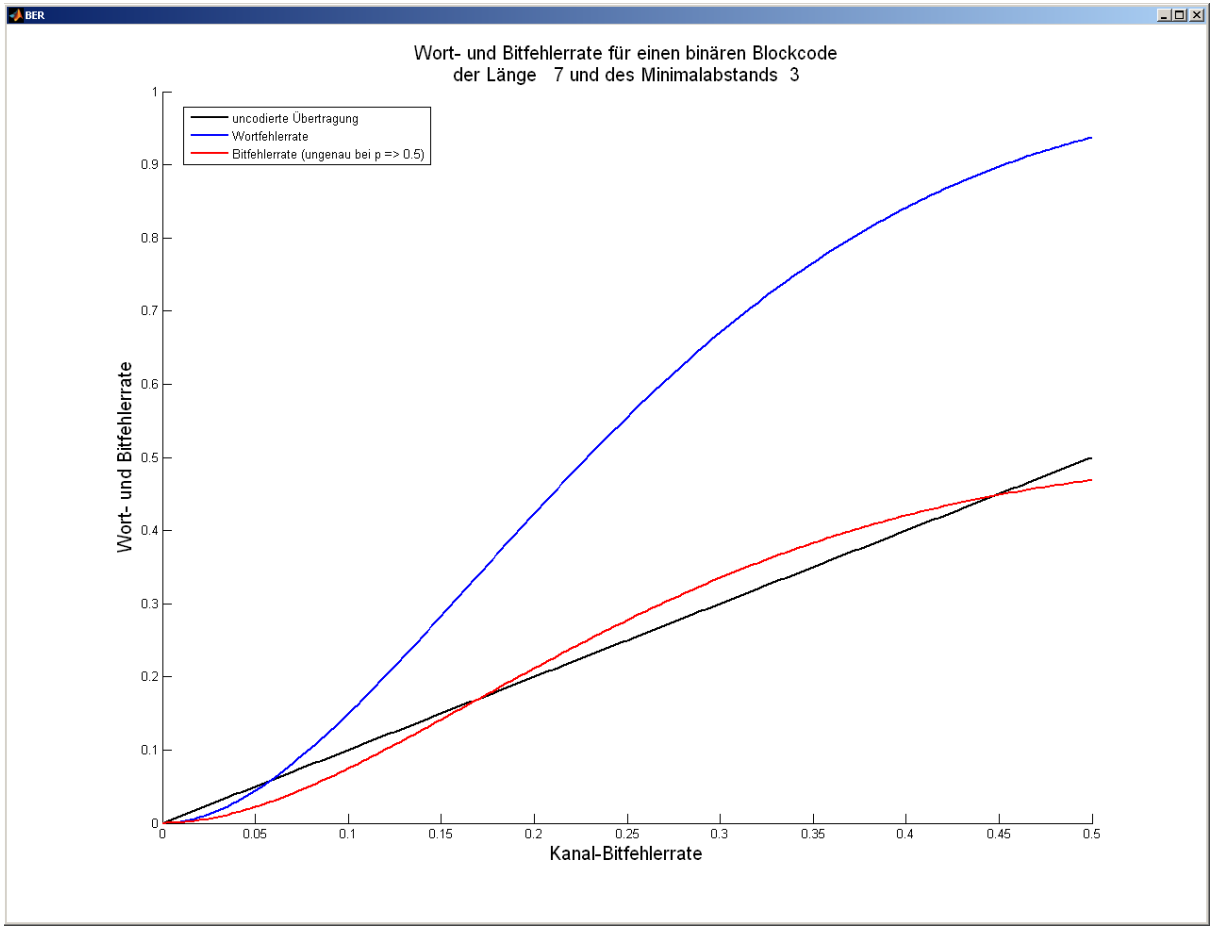
- (n, k)-Code: k-Informationsbits werden in n-Bits codiert
- Minimalabstand d der Codewörter voneinander
- ermöglicht Korrektur von r Bitfehlern $\quad r \leq (d-1)/2$

$\Rightarrow$ nicht korrigierbar sind: $r+1, r+2, \ldots \; n$ Bitfehler

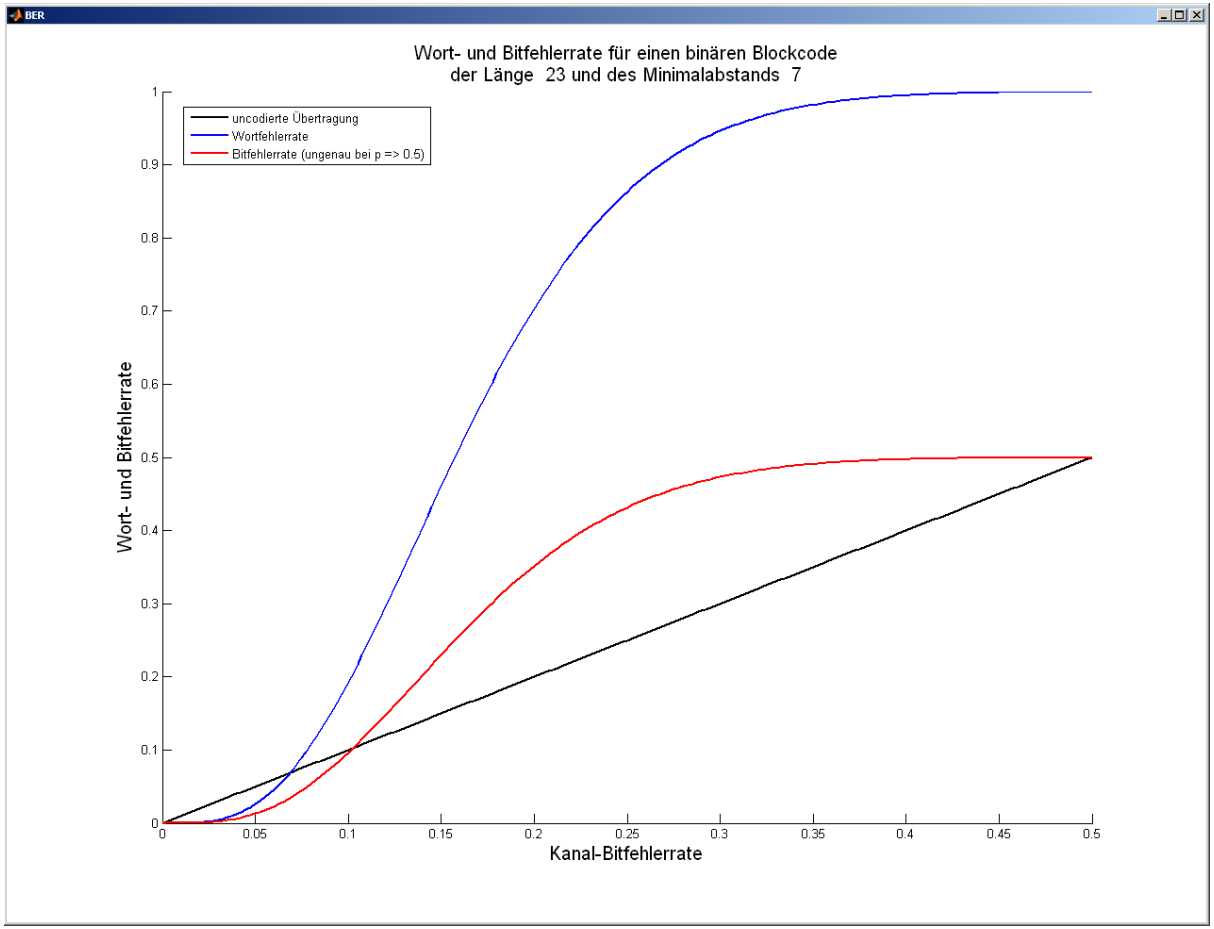
- Übertragungskanal hat Bitfehlerwahrscheinlichkeit

$\Rightarrow$ Wortfehlerwahrscheinlichkeit: Summe der Wahrscheinlichkeiten nicht korrigierbarer Bitfehler

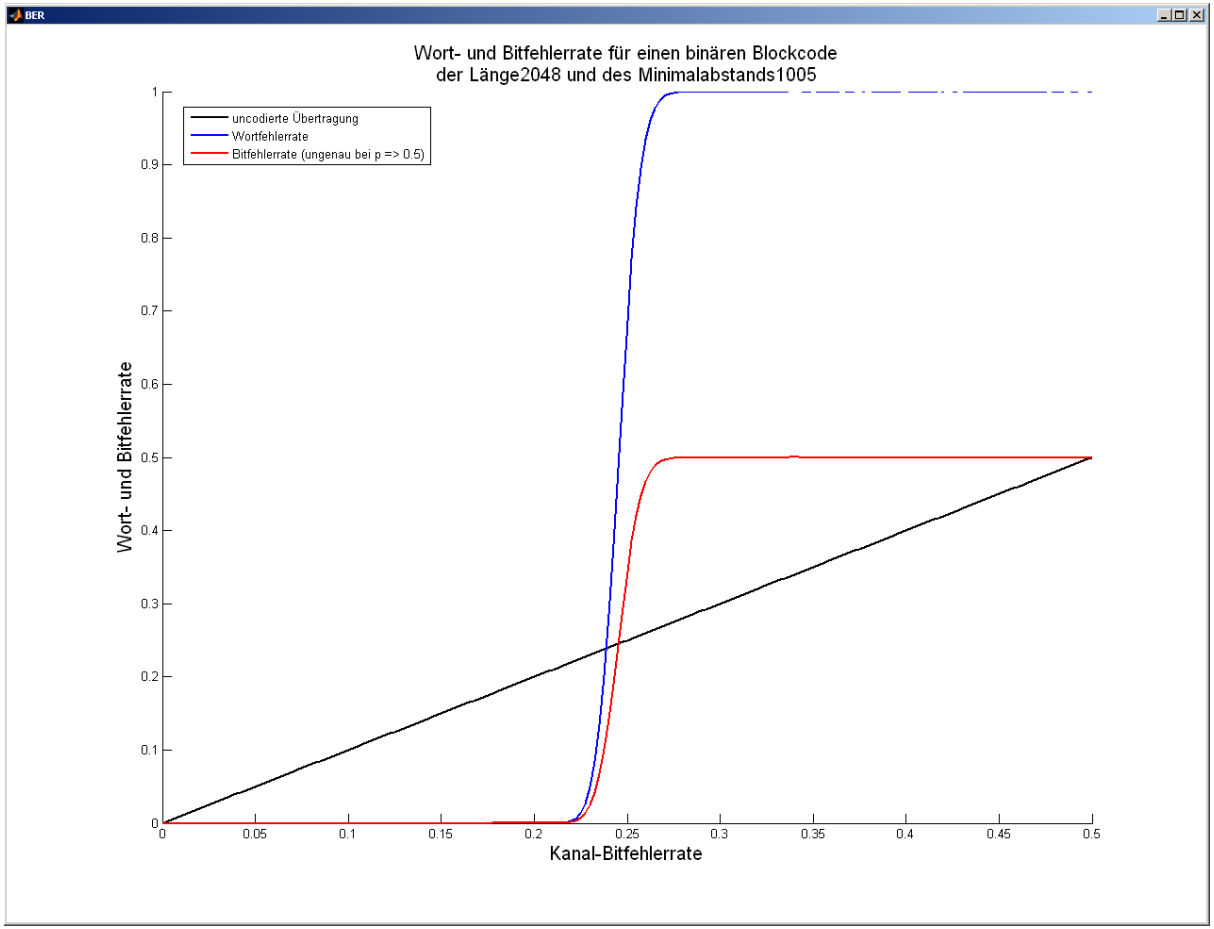
(7,4)-Hamming-Code: Fehlerrate



(23,7)-Golay-Code: Fehlerrate



(2048,1005)-Zufalls-Code: Fehlerrate



Binärpolynome

- jedem n-bit Wort $(d_1, d_2, \ldots, d_n)$ lässt sich ein Polynom über dem Körper $\{0, 1\}$ zuordnen
- Beispiel, mehrere mögliche Zuordnungen

$$\begin{aligned} 100\,1101 &= 1 \cdot x^6 + 0 \cdot x^5 + 0 \cdot x^4 + 1 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0 \\ &= x^6 + x^3 + x^2 + x^0 \\ &= x^0 + x^3 + x^4 + x^6 \\ &= x^0 + x^{-3} + x^{-4} + x^{-6} \\ &\ldots \end{aligned}$$

- mit diesen Polynomen kann „gerechnet“ werden: Addition, Subtraktion, Multiplikation, Division
- Theorie: Galois-Felder

Zyklische Codes (CRC)

CRC (*Cyclic Redundancy Check*)

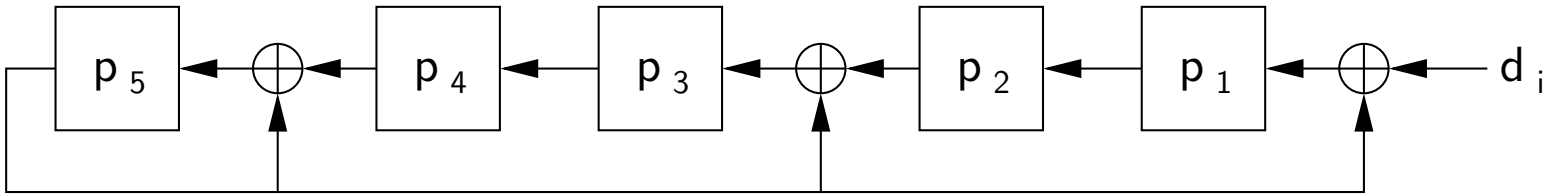
- Polynomdivision als Basis für CRC-Codes erzeugt Prüfbits
- zyklisch: Codewörter werden durch Schieben und Modifikation (mod 2 Summe) ineinander überführt

- Familie von Codes zur Fehlererkennung
  insbesondere auch zur Erkennung von Bündelfehlern

- in sehr vielen Codes benutzt
  - Polynom 0x04C11DB7 (CRC-32) in Ethernet, ZIP, PNG ...
  - weitere CRC-Codes in USB, ISDN, GSM, openPGP ...

Zyklische Codes (CRC) (cont.)

- Sehr effiziente Software- oder Hardwarerealisierung
  - rückgekoppelte Schieberegister und XOR
    LFSR (*Linear Feedback Shift Register*)
  - Beispiel $x^5 + x^4 + x^2 + 1$



- Codewort erstellen
  - Datenwort d_i um k 0-bits verlängern, Grad des Polynoms: k
  - bitweise in CRC-Check schieben
  - Divisionsrest bildet Registerinhalt p_i
  - Prüfbits p_i an ursprüngliches Datenwort anhängen

Zyklische Codes (CRC) (cont.)

- Test bei Empfänger
  - übertragenes Wort bitweise in CRC-Check schieben
    gleiches Polynom / Hardware wie bei Codierung
  - fehlerfrei, wenn Divisionsrest/Registerinhalt $= 0$

- je nach Polynom (# Prüfbits) unterschiedliche Güte
- Galois-Felder als mathematische Grundlage

en.wikipedia.org/wiki/Cyclic_redundancy_check
en.wikipedia.org/wiki/Computation_of_CRC
de.wikipedia.org/wiki/Zyklische_Redundanzprüfung
de.wikipedia.org/wiki/LFSR

Praxisbeispiel: EAN-13 Produktcode

Kombination diverser Codierungen:

- Land, Unternehmen, Artikelnummer, Prüfsumme
- 95-stelliges Bitmuster (schwarz: 1, weiss: 0)
- max. vier aufeinanderfolgende weisse/schwarze Bereiche
- Randzeichen 101 und Trennzeichen 01010 in der Mitte
- jede Ziffer mit 7 bit codiert, je zwei Linien und Freiräume
- insgesamt 12 Ziffern codiert, 3 Varianten (GGU) pro Ziffer
- 13 Ziffer als Prüfsumme über Abfolge von G/U Varianten

- http://de.wikipedia.org/wiki/European_Article_Number
- siehe Übungsaufgabe

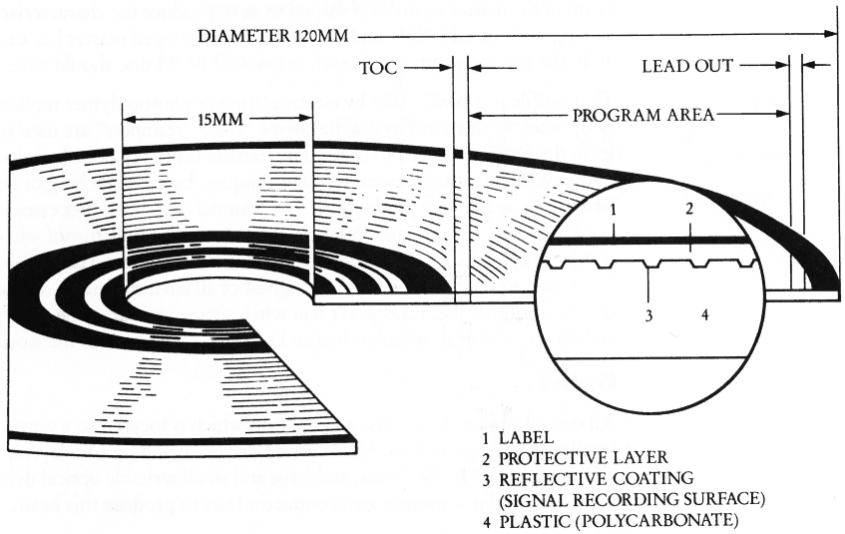
Praxisbeispiel: Compact Disc (Audio-CD und CD-ROM)

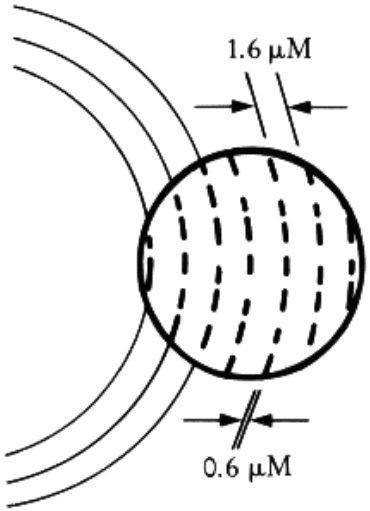
- Polycarbonatscheibe, spiralförmige geprägte Datenspur
- kleine Vertiefungen „*Pits*“ innerhalb der „*Lands*“
- Auslesen durch Intensität von reflektiertem Laserstrahl
- 650 MB Kapazität, Datenrate $\approx$ 150 KB/sec (1x speed)

- von Anfang an auf billigste Fertigung ausgelegt
- mehrstufige Fehlerkorrekturcodierung fest vorgesehen
- Kompensation von Fertigungsmängeln und -toleranzen
- Korrektur von Staub und Kratzern, etc.

- Audio-CD: Interpolation nicht korrigierbarer Fehler
- Daten-CD: geschachtelte weitere Codierung
- Bitfehlerrate $\leq 10^{11}$

Compact Disc





- spiralförmige Spur, ca. 16000 Windungen, Start innen
- geprägte Vertiefungen *pits*, dazwischen *lands*
- Wechsel pit/land oder land/pit codiert 1, dazwischen 0

Compact Disc: Mehrstufige Codierung

- Daten in *Frames* à 24 Bytes aufteilen
- 75 *Sektoren* mit je 98 Frames pro Sekunde
- Sektor enthält 2352 Bytes Nutzdaten (und 98 Bytes *Subcode*)

- pro Sektor 784 Byte Fehlerkorrektur hinzufügen
- Interleaving gegen Burst-Fehler (z.B. Kratzer)
- Code kann bis 7000 fehlende Bits korrigieren

- *eight-to-fourteen* Modulation: 8-Datenbits in 14 Codebits
- 2..10 Nullen zwischen zwei Einsen (pits/land Übergang)

- Daten-CD zusätzlich mit äußerem 2D Reed-Solomon Code
- pro Sektor 2048 Bytes Nutzdaten, 276 Bytes RS-Fehlerschutz

Farbbilder: JPEG

Joint Picture Experts Group Bildformat (1992):

- für die Speicherung von Fotos / Bildern
- verlustbehaftet

mehrere Codierungsschritte:

1	Farbraumkonvertierung: RGB nach YUV	verlustbehaftet
2	Aufteilung in Blöcke zu je 8x8 Pixeln	verlustfrei
3	DCT (*discrete cosinus transformation*)	verlustfrei
4	Quantisierung (einstellbar)	verlustbehaftet
5	Huffman-Codierung	verlustfrei

Video: MPEG

- *Motion Picture Experts Group*: Sammelname der Organisation und diverser aufeinander aufbauender Standards

Codierungsschritte für Video:

1. Einzelbilder wie JPEG (YUV, DCT, Huffman)
2. Differenzbildung mehrerer Bilder (Bewegungskompensation)
3. *Group of Pictures* (*I*-Frames, *P*-Frames, *B*-Frames)
4. Zusammenfassung von Audio, Video, Metadaten im sogenannten PES (*Packetized Elementary Stream*)
5. *Transport-Stream* Format für robuste Datenübertragung

Digitales Fernsehen: DVB

- *Digital Video Broadcast*: Sammelname für die europäischen Standards für digitales Fernsehen

Codierungsschritte:

1. Videocodierung nach MPEG-2 (geplant: MPEG-4)
2. Multiplexing mehrerer Programme nach MPEG-TS
3. optional: Metadaten (Electronic Program Guide)
4. vier Varianten für die eigentliche Kanalcodierung:
   DVB-S: Satellite, DVB-C: Cable, DVB-T: Terrestrial
   DVB-H: Handheld/Mobile

Literatur: Vertiefung

- Richard W. Hamming, *Information und Codierung*, VCH, 1987
- Klaus von der Heide, *Vorlesung Technische Informatik T1*, Universität Hamburg, FB Informatik, 2004
- Klaus von der Heide, *Vorlesung Digitale Datenübertragung*, Universität Hamburg, FB Informatik, 2005
- W.E. Ryan & S. Lin, *Channel Codes*, 2009