64-040 Modul InfB-RS: Rechnerstrukturen

https://tams.informatik.uni-hamburg.de/lectures/2016ws/vorlesung/rs

– Kapitel 9 –

Andreas Mäder

Universität Hamburg
Fakultät für Mathematik, Informatik und Naturwissenschaften
Fachbereich Informatik
Technische Aspekte Multimodaler Systeme

Wintersemester 2016/2017

Kapitel 9

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 derselben Abstraktionsebene

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

Code, Codewörter

- **Codewörter**: die Wörter der Repräsentation B aus einem Zeichenvorrat Z
- **Code**: die Menge aller Codewörter
- **Blockcode**: alle Codewörter haben dieselbe 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
  z.B. 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

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

Codierungen für Dezimalziffern (cont.)

- 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 $c_0 = c_n$

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 $c_n \Rightarrow$ 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

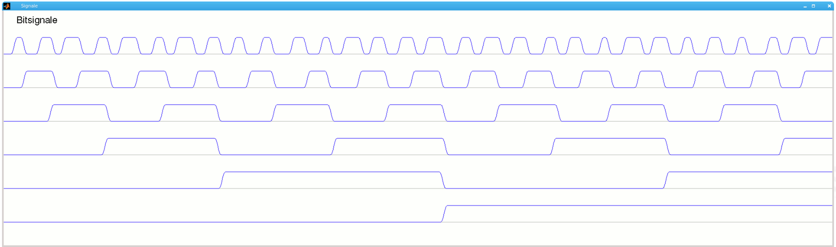
- Einschrittige Codes werden benutzt, wenn ein Ablesen der Bits auch beim Wechsel zwischen zwei Codeworten möglich ist (bzw. nicht verhindert werden kann)

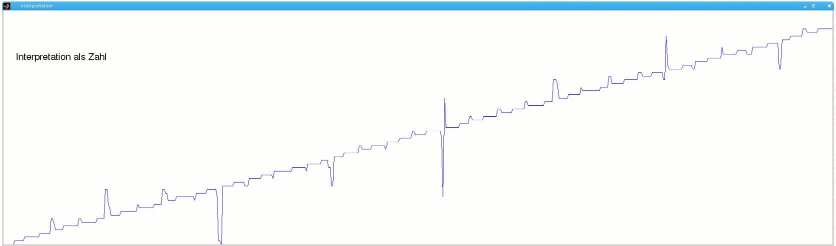
  z.B.: Winkelcodierscheiben oder digitale Schieblehre
- viele interessante Varianten möglich (s. Knuth: *AoCP* [Knu11])

Einschrittige Codes: Matlab-Demo

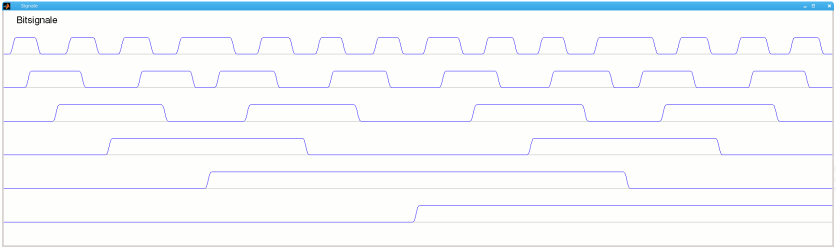
- Ablesen eines Wertes mit leicht gegeneinander verschobenen Übergängen der Bits [Hei05a], Kapitel 1.4
  - `demoeinschritt(0:59)` normaler Dualcode
  - `demoeinschritt(einschritt(60))` einschrittiger Code
- maximaler Ablesefehler
  - 2^{n-1} beim Dualcode
  - 1 beim einschrittigen Code
- Konstruktion eines einschrittigen Codes
  - rekursiv
  - als ununterbrochenen Pfad im KV-Diagramm (s.u.)

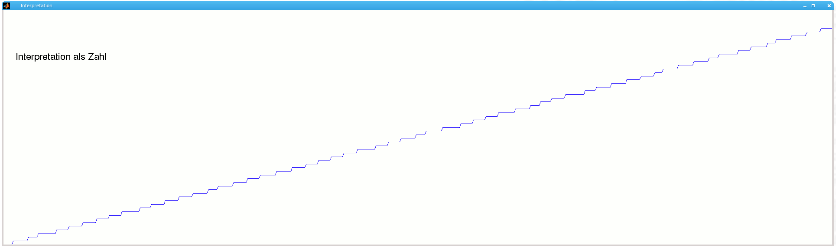
Ablesen des Wertes aus Dualcode



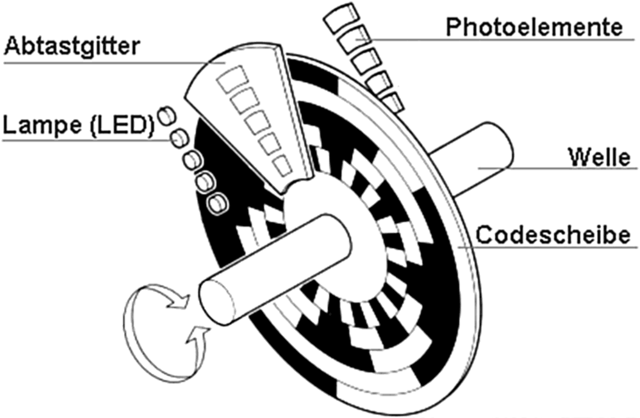


Ablesen des Wertes aus einschrittigem Code

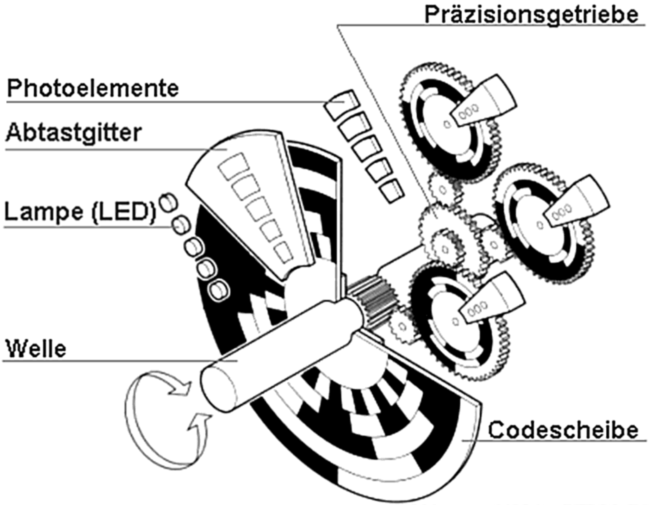




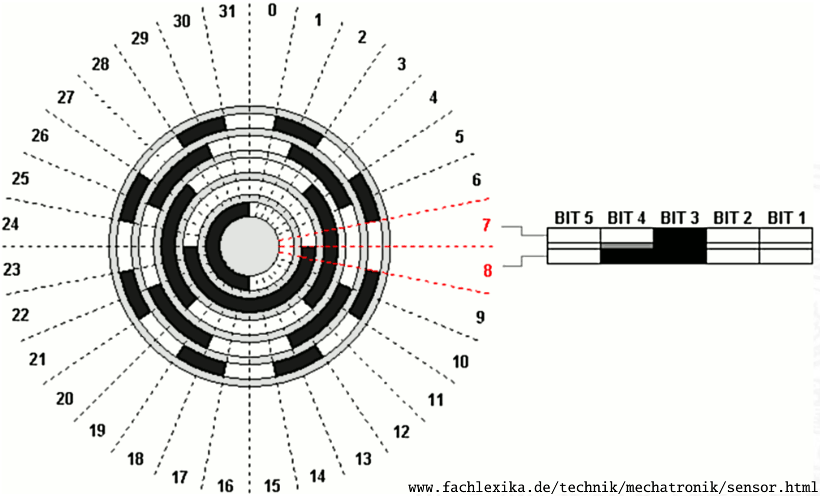
Gray-Code: Prinzip eines Winkeldrehgebers



Gray-Code: mehrstufiger Drehgeber

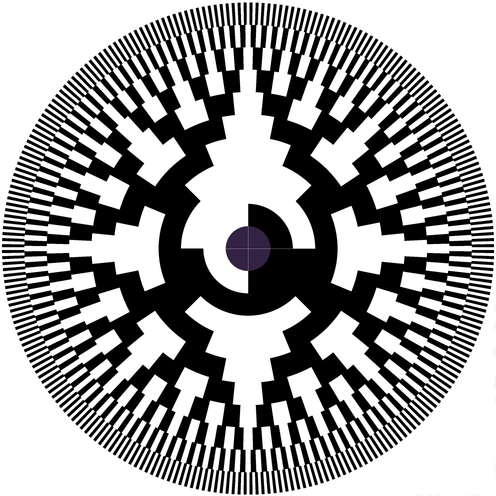


Gray-Code: 5-bit Codierscheibe



www.fachlexika.de/technik/mechatronik/sensor.html

Gray-Code: 10-bit Codierscheibe



Einschrittiger Code: rekursive Konstruktion

- Starte mit zwei Codewörtern: 0 und 1

- Gegeben: Einschrittiger Code C mit n Codewörtern
- Rekursion: Erzeuge Code C_2 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, 11, 10 }

{ 000, 001, 011, 010, 110, 111, 101, 100 }

...

⇒ Gray-Code

Karnaugh-Veitch Diagramm

$x_3 x_2$ \ $x_1 x_0$	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

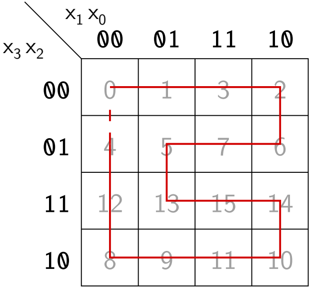
$x_3 x_2$ \ $x_1 x_0$	00	01	11	10
00	0000	0001	0011	0010
01	0100	0101	0111	0110
11	1100	1101	1111	1110
10	1000	1001	1011	1010

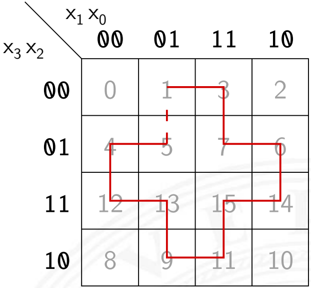
- 2D-Diagramm mit $2^n = 2^{n_y} \times 2^{n_x}$ Feldern
- gängige Größen sind: 2×2, 2×4, 4×4

  darüber hinaus: mehrere Diagramme der Größe 4×4
- Anordnung der Indizes ist im einschrittigen-Code / Gray-Code

⇒ benachbarte Felder unterscheiden sich gerade um 1 Bit

Einschrittiger Code: KV-Diagramm

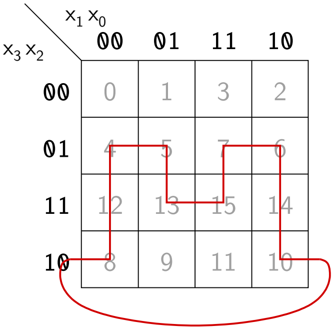


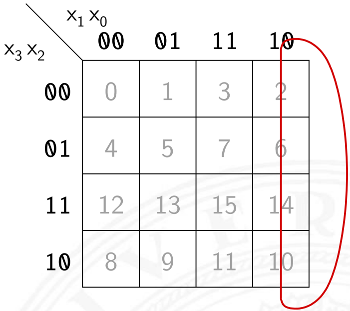


- Pfade

  0,1,3,2,6,7,5,13,15,14,10,11,9,8,12,4 1,3,7,6,14,15,11,9,13,12,4,5
- jeder Pfad entspricht einem einschrittigen Code
- geschlossener Pfad: zyklisch einschrittiger Code

Einschrittiger Code: KV-Diagramm (cont.)





- Pfade
  4,5,13,15,7,6,14,10,8,12   2,6,14,10

- linke und rechte Spalte unterscheiden sich um 1 Bit
  obere und untere Zeile unterscheiden sich um 1 Bit

⇒ KV-Diagramm als „außen zusammengeklebt“ denken

⇒ Pfade können auch „außen herum“ geführt werden

Gray-Code: Umwandlung in/von Dualcode

Umwandlung: Dual- in Graywort

1. MSB des Dualworts wird MSB des Grayworts
2. von links nach rechts: bei jedem Koeffizientenwechsel im Dualwort wird das entsprechende Bit im Graywort 1, sonst 0

- Beispiele 0011 $\rightarrow$ 0010, 1110 $\rightarrow$ 1001, 0110 $\rightarrow$ 0101 usw.

- `gray(x) = x ^ (x >>> 1)`

- in Hardware einfach durch paarweise XOR-Operationen

  [HenHA] Hades Demo: `10-gates/15-graycode/dual2gray`

Gray-Code: Umwandlung in/von Dualcode (cont.)

Umwandlung: Gray- in Dualwort

1. MSB wird übernommen
2. 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

- Beispiele 0010 $\rightarrow$ 0011, 1001 $\rightarrow$ 1110, 0101 $\rightarrow$ 0110 usw.

- in Hardware einfach durch Kette von XOR-Operationen

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

$\Rightarrow$ 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 Codeworts

- 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 2502	: Ortsnetz (keine führende Null)
040 42883 2502	: nationales Netz
0049 40 42883 2502	: internationale Rufnummer

- Morse-Code: Fano-Bedingung verletzt

Morse-Code

Codetabelle

● kurzer Ton —langer Ton

A	●—	S	●●●	.	●—●—●—	S-Start	—●—●—
B	—●●●	T	—	,	——●●——	Verst.	●●●—●
C	—●—●	U	●●—	?	●●——●●	S-Ende	●—●—●
D	—●●	V	●●●—	´	●————●	V-Ende	●●●—●—
E	●	W	●——	!	—●—●——	Error	●●●●●●●●
F	●●—●	X	—●●—	/	—●●—●		
G	——●	Y	—●——	(	—●——●	Ä	●—●—
H	●●●●	Z	——●●	)	—●——●—	À	●——●—
I	●●	0	—————	&	●—●●●	É	●●—●●
J	●———	1	●————	:	———●●●	È	●—●●—
K	—●—	2	●●———	;	—●—●—●	Ö	———●
L	●—●●	3	●●●——	=	—●●●—	Ü	●●——
M	——	4	●●●●—	+	●—●—●	ß	●●●——●●
N	—●	5	●●●●●	-	—●●●●—	CH	————
O	———	6	—●●●●	_	●●——●—	Ñ	——●——
P	●——●	7	——●●●	"	●—●●—●	...	
Q	——●—	8	———●●	$	●●●—●●—		
R	●—●	9	————●	@	●——●—●	SOS	●●● ——— ●●●

Morse-Code (cont.)

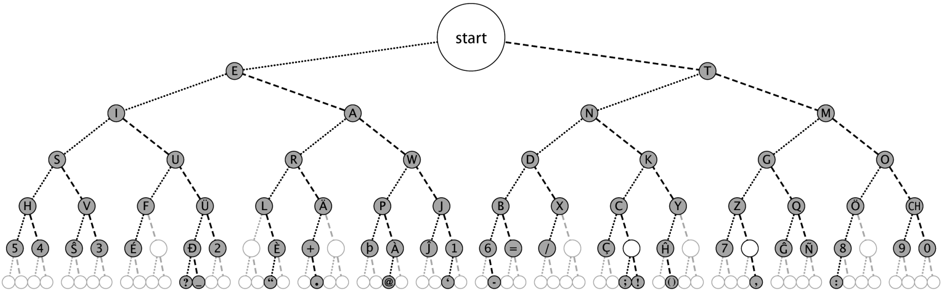
- Eindeutigkeit

Codewort:	• • • • • — •
E	•
I	• •
N	— •
R	• — •
S	• • •

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

- Codierung
  - Häufigkeit der Buchstaben = 1 / Länge des Codewortes
  - Effizienz: kürzere Codeworte
  - Darstellung als Codebaum

Morse-Code: Codebaum (Ausschnitt)



- Symbole als Knoten oder Blätter
- Knoten: Fano-Bedingung verletzt
- Codewort am Pfad von Wurzel zum Knoten/Blatt ablesen

Morse-Code: Umschlüsselung

Umschlüsselung des 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 als Umschlüsselung des langen Tons –
  01 als Umschlüsselung des kurzen Tons •
  00 als Trennzeichen zwischen Morse-Codewörtern

Codierung nach Fano (Shannon-Fano Codierung)

Gegeben: die zu codierenden Urwörter a_i
und die zugehörigen Wahrscheinlichkeiten $p(a_i)$

- Ordnung der Urwörter anhand ihrer Wahrscheinlichkeiten $p(a_1) \geq p(a_2) \geq \cdots \geq p(a_n)$
- Einteilung der geordneten Urwörter in zwei Gruppen mit möglichst gleicher Gesamtwahrscheinlichkeit. Eine Gruppe bekommt als erste Codewortstelle eine 0, die andere eine 1
- Diese Teilgruppen werden wiederum entsprechend geteilt, und den Hälften wieder eine 0, bzw. eine 1, als nächste Codewortstelle zugeordnet
- Das Verfahren 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 $\{0.45, 0.1, 0.15, 0.3\}$

0. Sortierung nach Wahrscheinlichkeiten ergibt $\{A, D, C, B\}$
1. Gruppenaufteilung ergibt $\{A\}$ und $\{D, C, B\}$
   Codierung von A mit 0 und den anderen Symbolen als $1*$
2. weitere Teilung ergibt $\{D\}$, und $\{C, B\}$
3. letzte Teilung ergibt $\{C\}$ und $\{B\}$

$\Rightarrow$ Codewörter sind $A = 0$, $D = 10$, $C = 110$ und $B = 111$

mittlere Codewortlänge L

- $L = 0.45 \cdot 1 + 0.3 \cdot 2 + 0.15 \cdot 3 + 0.1 \cdot 3 = 1.8$
- 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

Ameling: *Fano-Code der Buchstaben der deutschen Sprache*, 1992

Codierung nach Huffman

Gegeben: die zu codierenden Urwörter a_i
und die zugehörigen Wahrscheinlichkeiten $p(a_i)$

- Ordnung der Urwörter anhand ihrer Wahrscheinlichkeiten $p(a_1) \leq p(a_2) \leq \cdots \leq p(a_n)$
- in jedem Schritt werden die zwei Wörter mit der geringsten Wahrscheinlichkeit zusammengefasst und durch ein neues ersetzt
- das Verfahren wird wiederholt, bis eine Menge mit nur noch zwei Wörtern resultiert
- rekursive Codierung als Baum (z.B.: 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 $\{0.45, 0.1, 0.15, 0.3\}$

0. Sortierung nach Wahrscheinlichkeiten ergibt $\{B, C, D, A\}$
1. Zusammenfassen von B und C als neues Wort E, Wahrscheinlichkeit von E ist dann $p(E) = 0.1 + 0.15 = 0.25$
2. Zusammenfassen von E und D als neues Wort F mit $p(F) = 0.55$
3. Zuordnung der Bits entsprechend der Wahrscheinlichkeiten
   - $F = 0$ und $A = 1$
   - Split von F in $D = 00$ und $E = 01$
   - Split von E in $C = 010$ und $B = 011$

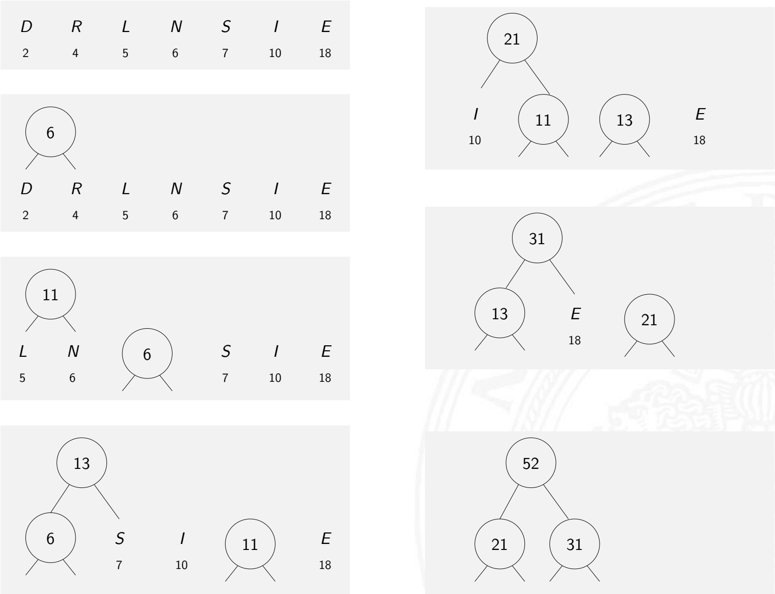
$\Rightarrow$ Codewörter sind $A = 1$, $D = 00$, $C = 010$ und $B = 011$

Bildung eines Huffman-Baums

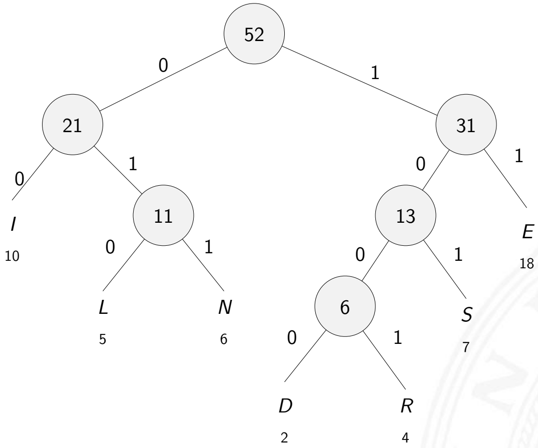
- Alphabet $= \{E, I, N, S, D, L, R\}$
- relative Häufigkeiten
  $E = 18, I = 10, N = 6, S = 7, D = 2, L = 5, R = 4$

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

Bildung eines Huffman-Baums (cont.)



Bildung eines Huffman-Baums (cont.)



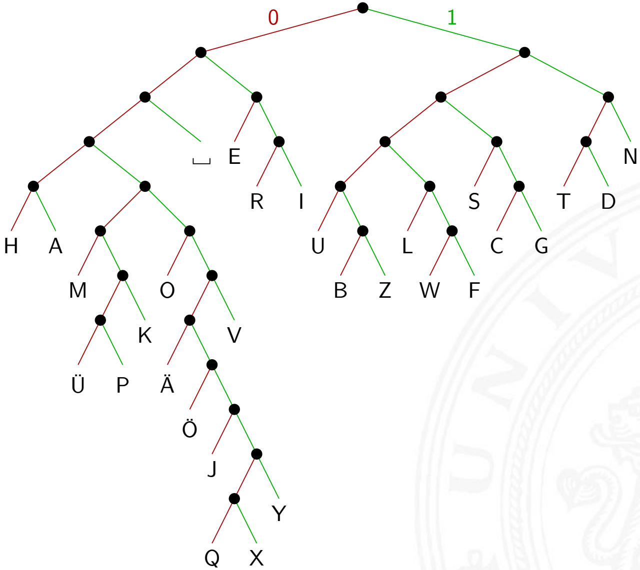
I	00
L	010
N	011
D	1000
R	1001
S	101
E	11

1001	00	11	101	11
R	*I*	*E*	*S*	*E*

Codierung nach Huffman: Deutsche Großbuchstaben

Zeichen	Code	Zeichen	Code
Leerzeichen	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
  - mittlere 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

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

- Beispiel: Bild mit einheitlicher Hintergrundfarbe codieren
- andere Ideen notwendig
  - Lauflängencodierung (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 decodierten Zeichen angepasst
- Symbolhäufigkeiten werden nicht explizit übertragen

D. E. Knuth: *Dynamic Huffman Coding*, 1985 [Knu85]

Kraft-Ungleichung

- Leon G. Kraft, 1949
  https://de.wikipedia.org/wiki/Kraft-Ungleichung

- 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: $\frac{1}{3} + 5 \cdot \frac{1}{3^2} + 3 \cdot \frac{1}{3^3} = 1$

$\Rightarrow$ 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

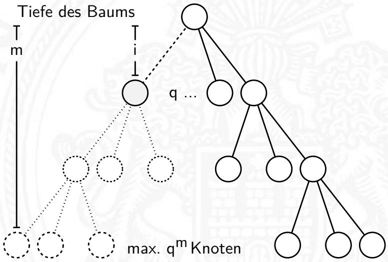
- Wir schreiben

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

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

- 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

$\Rightarrow$ erfüllt C die Präfix-Bedingung, dann 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

$\Rightarrow$ Informationsgewinn („Überraschung“) wird größer, je kleiner p_i

Geeignetes Maß für die Information?

- Wir erhalten die Nachricht A mit der Wahrscheinlichkeit p_A und anschließend die unabhängige Nachricht B mit der 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_i
- Wahl von $1/p$ als Maß für den Informationsgewinn?
- möglich, aber der Gesamtinformationsgehalt zweier (mehrerer) Ereignisse wäre das Produkt der einzelnen Informationsgehalte
- additive Größe wäre besser $\Rightarrow$ Logarithmus von $1/p$ bilden

Erinnerung: Logarithmus

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

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

- Rechenregeln
  - $log(x \cdot y) = log(x) + log(y)$ (Addition statt Multiplikation)
  - $b^{\log_b(x)} = x$ und $\log_b(b^x) = x$
  - $\log_b(x) = \frac{\log_a(x)}{log_a(b)}$
  - $\log_2(x) = \ln(x)/\ln(2) = \ln(x)/0,693141718$

Erinnerung: Binärer Logarithmus

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

  $\log_2(x^2) = \; b_1.b_2 b_3 \ldots$ wegen $\log(x^2) = 2\log(x)$

- Berechnung

  Input: $1 < x < 2$ (ggf. vorher skalieren)

  Output: Nachkommastellen b_i der Binärdarstellung von $ld(x)$

```
i = 0
loop
  i = i+1
  x = x*x
  if (x >= 2)     then      x    = x/2
                           b[i] = 1
                  else     b[i] = 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

- Beispiel: 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

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

- 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 „bit“ 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

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 $\Rightarrow$ Informationsgehalt der Nachricht „Ich wurde nicht vom Meteor erschlagen“ ist sehr klein
- Umgekehrt wäre die Überraschung groß: $\log_2(1/p_M) = 53,15$

Informationsgehalt: Beispiele (cont.)

Würfeln

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

- die Wahrscheinlichkeit, eine 6 zu würfeln, ist $\frac{1}{6}$
- $I(6) = \log_2(1/\frac{1}{6}) = 2,585$

Informationsgehalt: Beispiele (cont.)

Information eines Buchs

- Gegeben seien zwei Bücher
  1. deutscher Text
  2. mit Zufallsgenerator mit Gleichverteilung aus Alphabet mit 80-Zeichen erzeugt
- Informationsgehalt in beiden Fällen?
  1. Im deutschen Text abhängig vom Kontext!
     Beispiel: Empfangen wir als deutschen Text *„Der Begrif"*, so ist *„f"* als nächstes Symbol sehr wahrscheinlich
  2. beim Zufallstext liefert jedes neue Symbol die zusätzliche Information $I = \log_2(1/(1/80))$

$\Rightarrow$ der Zufallstext enthält die größtmögliche Information

Informationsgehalt: Beispiele (cont.)

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

Bit : als Maß für den Informationsgehalt
bit : Anzahl der Binärstellen 0 und 1

Entropie

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

- Was ist die **durchschnittliche Information** bei Empfang eines Symbols?
- diesen Erwartungswert bezeichnet man als **Entropie** des Systems (auch **mittlerer Informationsgehalt**)

- Wahrscheinlichkeiten aller möglichen Ereignisse A_i seien $W\{A_i\} = p_i$
- da jeweils eines der möglichen Symbole eintrifft, gilt $\sum_i p_i = 1$

Entropie (cont.)

- dann berechnet sich die Entropie H als Erwartungswert

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

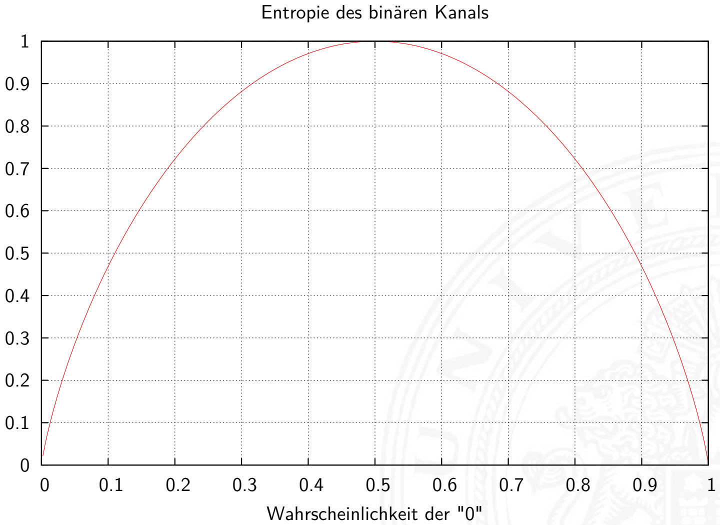
- als Funktion der Symbol-Wahrscheinlichkeiten nur abhängig vom stochastischen Modell

Entropie: Beispiele

1. drei mögliche Ereignisse mit Wahrscheinlichkeiten $\{\frac{1}{2}, \frac{1}{3}, \frac{1}{6}\}$
   - dann berechnet sich die Entropie zu
     $H = -(\frac{1}{2}\log_2(\frac{1}{2}) + \frac{1}{3}\log_2(\frac{1}{3}) + \frac{1}{6}\log_2(\frac{1}{6})) = 1,4591$

2. Empfang einer 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 einer Binärstelle mit den Wahrscheinlichkeiten $p_0 = q$ und $p_1 = (1 - q)$

Entropie: Symbolverteilung

- 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), gleich wahrscheinlich
  $H = \sum_i p_i \log_2(1/p_i) = 256 \cdot (1/256) \cdot \log_2(1/(1/256)) = 8\,\text{Bit}$

Entropie: einige Eigenschaften

1. $H(p_1, p_2, \ldots, p_n)$ ist maximal, falls $p_i = 1/n \quad (1 \leq i \leq n)$
2. H ist symmetrisch, für jede Permutation π von $1, 2, \ldots, n$ gilt:
   $H(p_1, p_2, \ldots, p_n) = H(p_{\pi(1)}, p_{\pi(2)}, \ldots, p_{\pi(n)})$
3. $H(p_1, p_2, \ldots, p_n) \geq 0 \quad$ mit $H(0, 0 \ldots 0, 1, 0 \ldots 0, 0) = 0$
4. $H(p_1, p_2, \ldots, p_n, 0) = H(p_1, p_2, \ldots, p_n)$
5. $H(1/n, 1/n, \ldots, 1/n) \leq H(1/(n+1), 1/(n+1), \ldots, 1/(n+1))$
6. H ist stetig in seinen Argumenten
7. Additivität: seien $n, m \in N^+$
   $H(\frac{1}{n \cdot m}, \frac{1}{n \cdot m}, \ldots, \frac{1}{n \cdot m}) = H(\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n}) + H(\frac{1}{m}, \frac{1}{m}, \ldots, \frac{1}{m})$

Möglicher Informationsgehalt

- **möglicher Informationsgehalt** H_0 ist durch Symbolcodierung festgelegt (entspricht **mittlerer Codewortlänge** $\bar{l}$)

$$H_0 = \sum_i p_i \cdot \log_2(q^{l_i})$$

  - stochastisches Modell $W\{A_i\} = p_i$ (Wahrscheinlichkeiten von Ereignissen A_i)
  - Codierung der Ereignisse (der Symbole) $C(A_i)$ durch Code der Länge l_i über einem q-nären Alphabet

- für Binärcodes gilt $$H_0 = \sum_i p_i \cdot l_i$$

- binäre Blockcodes mit Wortlänge N bits: $H_0 = N$

Redundanz

- **Redundanz** (engl. *code redundancy*):
  die Differenz zwischen dem möglichen und dem tatsächlich genutzten Informationsgehalt $R = H_0 - H$
  - möglicher Informationsgehalt H_0 ist durch Symbolcodierung festgelegt = mittlere Codewortlänge
  - tatsächliche Informationsgehalt ist die Entropie H

- **relative Redundanz**: $r = \frac{H_0 - H}{H_0}$

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

$$
\begin{aligned}
R = H_0 - H &= H_0 - \left(- \sum_{i=1}^{m} p(a_i) \cdot \log_2(p(a_i)) \right) \\
&= N + \sum_{i=1}^{m} p(a_i) \cdot \log_2(p(a_i))
\end{aligned}
$$

Informationstheorie ursprünglich entwickelt zur

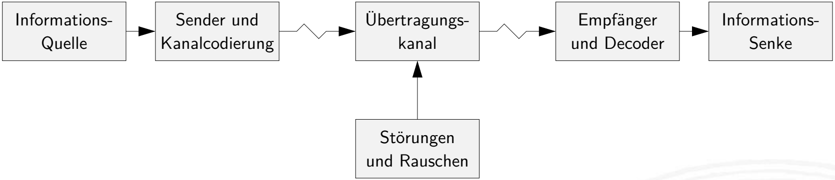
- formalen Behandlung der Übertragung von Information
- über reale, nicht fehlerfreie Kanäle
- deren Verhalten als stochastisches Modell formuliert werden kann

- zentrales Resultat ist die **Kanalkapazität** C des **binären symmetrischen Kanals**
- 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

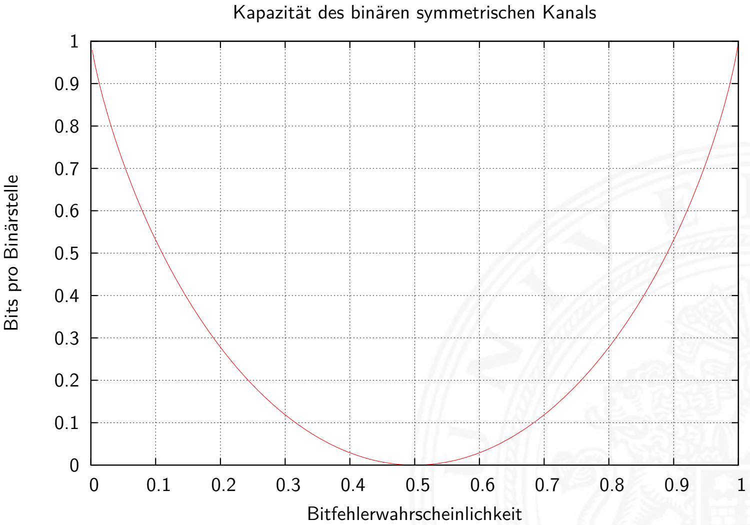
Binärer symmetrischer Kanal

- Wahrscheinlichkeit der beiden Symbole 0 und 1 ist gleich $(\frac{1}{2})$
- Wahrscheinlichkeit P, dass bei Übertragungsfehlern aus einer 0 eine 1 wird = Wahrscheinlichkeit, dass aus einer 1 eine 0 wird
- Wahrscheinlichkeit eines Fehlers an Binärstelle i ist unabhängig vom Auftreten eines Fehlers an anderen Stellen
- Entropie des Fehlerverhaltens

$$H(F) = P \cdot \log_2(1/P) + (1 - P) \cdot \log_2(1/(1 - P))$$

- Kanalkapazität ist $C = 1 - H(F)$

Kanalkapazität: Diagramm



Kanalkapazität: Konsequenzen

- bei $P = 0,5$ ist die Kanalkapazität $C = 0$

$\Rightarrow$ 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 (Fehler)
- Theorie liefert keine Hinweise, wie die fehlerfreie Übertragung praktisch durchgeführt werden kann

Shannon-Theorem

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

Gegeben:
binärer symmetrischer Kanal mit der Störwahrscheinlichkeit P
und der Kapazität $C(P)$

Shannon-Theorem

Falls die Übertragungsrate R kleiner als $C(P)$ ist,
findet man zu jedem $\epsilon > 0$ einen Code $\mathcal{C}$ mit
der Übertragungsrate $R(\mathcal{C})$ und $C(P) \geq R(\mathcal{C}) \geq R$ und
der Fehlerdecodierwahrscheinlichkeit $< \epsilon$

auch: C. E. Shannon: *A Mathematical Theory of Communication*

Shannon-Theorem (cont.)

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

$\Rightarrow$ Wenn die Übertragungsrate kleiner als die Kanalkapazität ist, existieren Codes, die beliebig zuverlässig sind

… und deren Signalübertragungsraten beliebig nahe der Kanalkapazität liegen

- 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 Codes, usw.

Fehlererkennende / -korrigierende Codes

Motivation

- Informationstheorie
- Kanalkapazität
- Shannon-Theorem

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

$\Rightarrow$ fehlererkennende Codes
$\Rightarrow$ 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 / der 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
  - Bitstrom erzeugt Folge von Automatenzuständen
  - Decodierung über bedingte Wahrscheinlichkeiten bei Zustandsübergängen
  - im Prinzip linear, Faltungscodes passen aber nicht in Beschreibung unten

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

Einschub: $GF(2)$, $GF(2)^n$
de.wikipedia.org/wiki/Endlicher_Körper
en.wikipedia.org/wiki/GF(2)

Boole'sche Algebra Details: Mathe-Skript, Wikipedia, v.d. Heide [Hei05a]

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

Galois-Feld mit zwei Elementen: GF(2)

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

Begriffe zur Fehlerbehandlung (cont.)

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

- **zyklischer Code**
  - ein Block-Code (identische Wortlänge aller Codewörter)
  - für jedes Codewort gilt: auch alle zyklischen Verschiebungen (Rotationen, z.B. rotate-left) sind Codeworte

  $\Rightarrow$ bei serieller Übertragung erlaubt dies die Erkennung/Korrektur von Bündelfehlern

Verfahren zur Fehlerbehandlung

- **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 sind 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 Codeworts vom Null-Wort

- Beispiel $\quad$ $\mathrm{a} = 0110\,0011$
  $\mathrm{b} = 1010\,0111$

$\Rightarrow$ 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

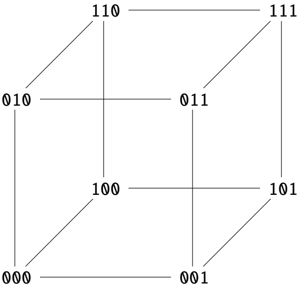
  dieser Abstand heißt dann **Minimalabstand** d

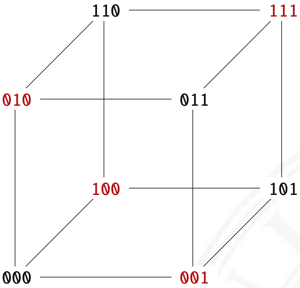
$\Rightarrow$ Fehlererkennung bis zu $(d-1)$ fehlerhaften Stellen

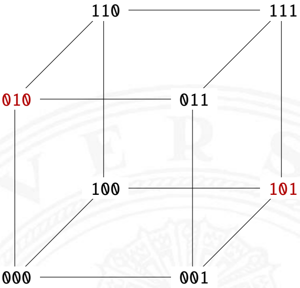
Fehlerkorrektur bis zu $((d-1)/2)$ –"–

Fehlererkennende und -korrigierende Codes (cont.)

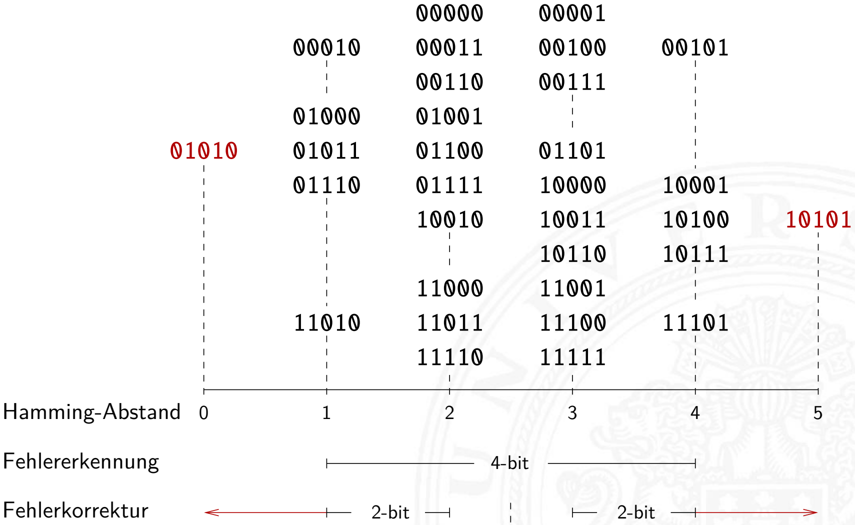
- Hamming-Abstand 2 3







Fehlererkennende und -korrigierende Codes (cont.)



Prüfinformation

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

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

verschiedene Verfahren

- 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*): $y_{even} = (z_1, \ldots, z_n, p)$
  $p(y_{even}) = (\sum_i y_i) \mod 2 = 0$

  ungerade Parität (*odd parity*): $y_{odd} = (z_1, \ldots, z_n, \overline{p})$
  $p(y_{odd}) = (\sum_i y_i) \mod 2 = 1$

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
- 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
  n-dimensionale Anordnung und Berechnung von n Paritätsbits

Zweidimensionale Parität: Beispiel

H	100 1000	0
A	100 0001	0
M	100 1101	0
M	100 1101	0
I	100 1001	1
N	100 1110	0
G	100 0111	0
	100 1001	1

Fehlerfall	100 1000	0
	100 0101	0
	110 1101	0
	100 1101	0
	000 1001	1
	100 1110	0
	100 0111	0
	100 1000	1

- Symbol: 7 ASCII-Zeichen, gerade Parität (*even*)
  64 bits pro Symbol (49 für Nutzdaten und 15 für Parität)
- links: Beispiel für ein Codewort und Paritätsbits
- rechts: empfangenes Codewort mit vier Fehlern,
  davon ein Fehler in den Paritätsbits

Zweidimensionale Parität: Einzelfehler

H	100 1000	0
A	100 0001	0
M	100 1101	0
M	100 1101	0
I	100 1001	1
N	100 1110	0
G	100 0111	0
	100 1001	1

Fehlerfall	100 1000	0	
	100 0101	0	1
	100 1101	0	
	100 1101	0	
	100 1001	1	
	100 1110	0	
	100 0111	0	
	100 1001	1	
	1		

- Empfänger: berechnet Parität und vergleicht mit gesendeter P.
- 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

- Parität als Zeilen/Spaltensumme mod 10 hinzufügen

- Daten

3	7	4
5	4	8
1	3	5

Parität

3	7	4	4
5	4	8	7
1	3	5	9
9	4	7	

Fehlerfall

3	7	4	4
5	4	3	2
1	3	5	9
9	4	2	

International Standard Book Number
ISBN-10 (1970), ISBN-13

- an EAN (*European Article Number*) gekoppelt
- Codierung eines Buches als Tupel

1. Präfix (nur ISBN-13)
2. Gruppennummer für den Sprachraum als Fano-Code:
   0 – 7, 80 – 94, 950 – 995, 9960 – 9989, 99900 – 99999
   - 0, 1: englisch – AUS, UK, USA…
   - 2: französisch – F…
   - 3: deutsch – A, DE, CH
   - …
3. Verlag, Nummer als Fano-Code:
   00 – 19 (1 Mio Titel), 20 – 699 (100 000 Titel) usw.
4. verlagsinterne Nummer
5. Prüfziffer

ISBN-10: Prüfverfahren

- ISBN-10 Zahl: $z_1, z_2, \ldots, z_{10}$
- Prüfsumme berechnen, Symbol X steht für Ziffer 10

$$\sum_{i=1}^{9} (i \cdot z_i) \mod 11 = z_{10}$$

- ISBN-Zahl zulässig, genau dann wenn

$$\sum_{i=1}^{10} (i \cdot z_i) \mod 11 = 0$$

- Beispiel: 0-13-713336-7

$$1 \cdot 0 + 2 \cdot 1 + 3 \cdot 3 + 4 \cdot 7 + 5 \cdot 1 + 6 \cdot 3 + 7 \cdot 3 + 8 \cdot 3 + 9 \cdot 6 = 161$$

$$161 \mod 11 = 7$$

$$161 + 10 \cdot 7 = 231 \qquad 231 \mod 11 = 0$$

ISBN: Fehlertypen

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

- Beispiel: vertausche i-te und j-te Ziffer (mit $i \neq j$)
  Prüfsumme: $\langle korrekt \rangle$ - $\langle falsch \rangle$
  $= i \cdot z_i + j \cdot z_j - j \cdot z_i - i \cdot z_j = (i - j) \cdot (z_i - z_j)$ mit $z_i \neq z_j$.

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

- dreifache Wiederholung jedes Datenworts
- (3,1)-Hamming-Code: Generatormatrix ist $G = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$
- Codewörter ergeben sich als Multiplikation von G mit dem Informationsvektor u (jeweils ein Bit)

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

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

- Verallgemeinerung als n-fach Wiederholungscode
- systematischer Code mit Minimalabstand $D = n$
- Decodierung durch Mehrheitsentscheid: 1-bit Fehlerkorrektur
- – Nachteil: geringe Datenrate

- 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 (cont.)

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	...

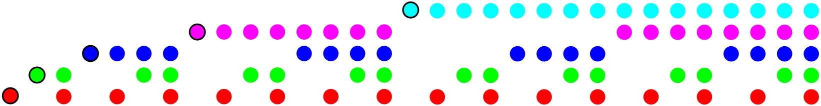
3. berechne Prüfbit i als mod 2-Summe der Bits (XOR), deren Positionsnummer ein gesetztes i-bit enthält

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

Prüfbits werden dabei auch berücksichtigt

Hamming-Code (cont.)

Schema: 2…5 Prüfbits



2^0	2^1		2^2				2^3								2^4															
C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{16}	C_{17}	C_{18}	C_{19}	C_{20}	C_{21}	C_{22}	C_{23}	C_{24}	C_{25}	C_{26}	C_{27}	C_{28}	C_{29}	C_{30}	C_{31}
p_1	p_2	d_1	p_3	d_2	d_3	d_4	p_4	d_5	d_6	d_7	d_8	d_9	d_{10}	d_{11}	p_5	d_{12}	d_{13}	d_{14}	d_{15}	d_{16}	d_{17}	d_{18}	d_{19}	d_{20}	d_{21}	d_{22}	d_{23}	d_{24}	d_{25}	d_{26}

(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
- 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 (cont.)

- 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

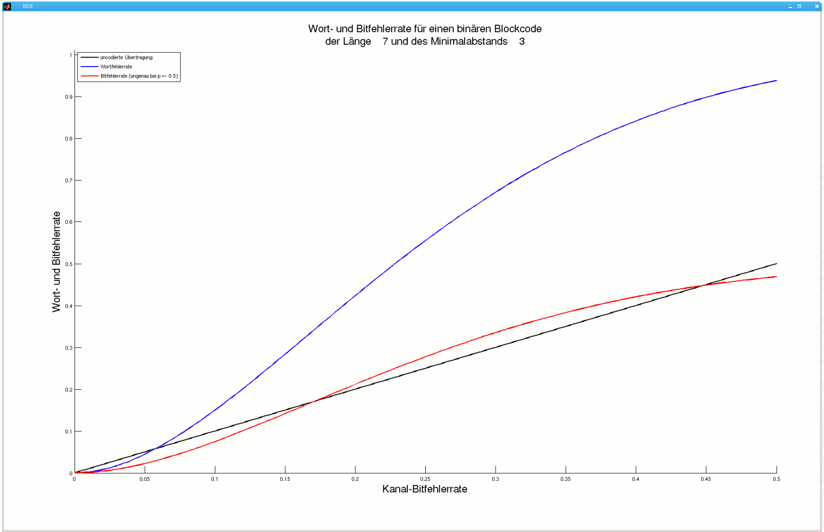
- (n, k)-Code: k-Informationsbits werden in n-Bits codiert
- Minimalabstand d der Codewörter voneinander
- ermöglicht Korrektur von r Bitfehlern $\quad r \leq (d-1)/2$

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

- Übertragungskanal hat Bitfehlerwahrscheinlichkeit

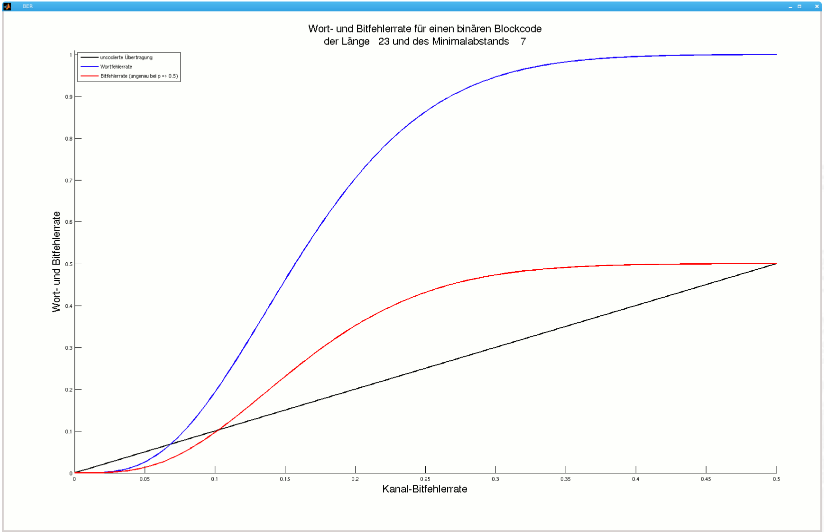
$\Rightarrow$ Wortfehlerwahrscheinlichkeit: Summe der Wahrscheinlichkeiten nicht korrigierbarer Bitfehler

Fehlerrate: (7,4)-Hamming-Code



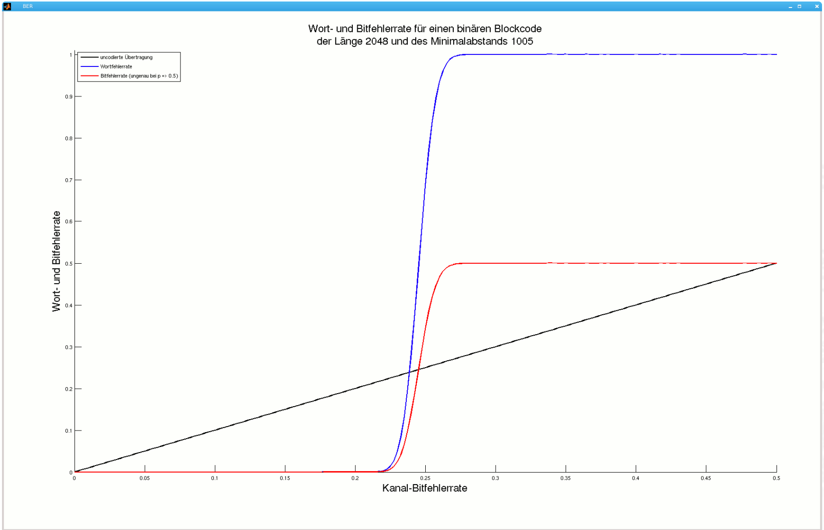
[Hei05a]

Fehlerrate: (23,12)-Golay-Code



[Hei05a]

Fehlerrate: (2048,8)-Randomcode



[Hei05a]

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)

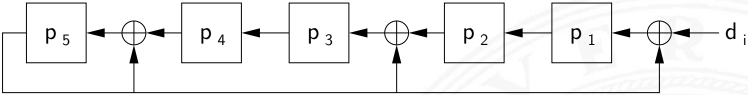
CRC (*Cyclic Redundancy Check*)

- 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 ...

- 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

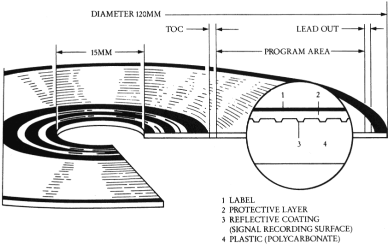
de.wikipedia.org/wiki/European_Article_Number

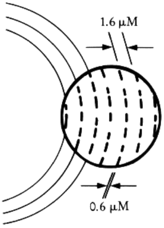
Kombination diverser Codierungen:

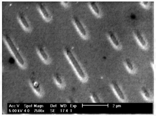
- Land, Unternehmen, Artikelnummer, Prüfsumme
- 95-stelliges Bitmuster
  - schwarz $\widehat{=}$ 1, weiss $\widehat{=}$ 0
  - max. vier aufeinanderfolgende weisse/schwarze Bereiche
  - Randzeichen: 101
    Trennzeichen in der Mitte: 01010
- 13 Ziffern: 7 links, 6 rechts
  - jede Ziffer mit 7 bit codiert, je zwei Linien und Freiräume
  - 3 Varianten pro Ziffer: links ungerade/gerade, rechts
  - 12 Ziffern Code
  - 13. Ziffer als Prüfsumme über Abfolge von u/g Varianten

Compact Disc
Audio-CD und CD-ROM

- Polycarbonatscheibe, spiralförmige geprägte Datenspur







  - spiralförmige Spur, ca. 16000 Windungen, Start innen
  - geprägte Vertiefungen *pits*, dazwischen *lands*
  - Wechsel pit/land oder land/pit codiert 1, dazwischen 0
- Auslesen durch Intensität von reflektiertem Laserstrahl
- 650 MiB Kapazität, Datenrate ≈ 150 KiB/sec (1x speed)

Compact Disc (cont.)

Audio-CD und CD-ROM

- 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: Mehrstufige Codierung

- Daten in *Frames* à 24 Bytes aufteilen
- 75 *Sektoren* mit je 98 Frames pro Sekunde
- Sektor enthält 2 352 Bytes Nutzdaten (und 98 Bytes *Subcode*)

- pro Sektor 784 Byte Fehlerkorrektur hinzufügen
- Interleaving gegen Burst-Fehler (z.B. Kratzer)
- Code kann bis 7 000 fehlende Bits korrigieren

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

- Daten-CD zusätzlich mit äußerem 2D *Reed-Solomon Code*
- pro Sektor 2 048 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

[Ham87] R.W. Hamming: *Information und Codierung*. VCH, 1987. ISBN 3–527–26611–9

[Hei05a] K. von der Heide: *Vorlesung: Technische Informatik 1 — interaktives Skript*. Universität Hamburg, FB Informatik, 2005. tams.informatik.uni-hamburg.de/lectures/2004ws/vorlesung/t1

[Hei05b] K. von der Heide: *Vorlesung: Digitale Datenübertragung*. Universität Hamburg, FB Informatik, 2005, Vorlesungsskript. tams.informatik.uni-hamburg.de/lectures/2005ss/vorlesung/Digit

[HenHA] N. Hendrich: *HADES — HAmburg DEsign System*. Universität Hamburg, FB Informatik, Lehrmaterial. tams.informatik.uni-hamburg.de/applets/hades

Literatur (cont.)

[RL09] W.E. Ryan, S. Lin: *Channel codes: classical and modern*. Cambridge University Press, 2009. ISBN 0–521–84868–7

[Knu85] D.E. Knuth: *Dynamic Huffman Coding*. in: *J. of Algorithms* 6 (1985), Nr. 2, S. 163–180

[Knu11] D.E. Knuth: *The Art of Computer Programming, Volume 4A, Combinatorial Algorithms, Part 1*. Addison-Wesley Professional, 2011. ISBN 978–0–201–03804–0