Codierungstheorie

Skript zur Vorlesung im WS 2005/06

Prof. Peter Hauck
Arbeitsbereich Diskrete Mathematik
Wilhelm-Schickard-Institut
Universität Tübingen

LaTeX-Fassung von Daniel Raible

Inhaltsverzeichnis

Literatur zur Vorlesung 'Codierungstheorie'

1. E. R. Berlekamp, *Algebraic Coding Theory*. Aegean Park Press, 1984.
2. B. Friedrichs, *Kanalcodierung*. Springer, 1996.
3. D. Jungnickel, *Codierungstheorie*. Spektrum Akademischer Verlag, 1995.
4. J. Justeson and T. Høholdt, *A Course in Error-Correcting Codes*. European Mathematical Society, 2004.
5. S. Lin and D.J. Costello Jr., *Error Control Coding*. Pearson Education International, 2004
6. W. Lütkebohmert, *Codierungstheorie*. Vieweg, 2003.
7. F. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. North Holland, 1977.
8. V. S. Pless and W. C. Huffman (editors), *Handbook of Coding Theory I, II*. Elsevier, 1998.
9. S. Roman, *Coding and Information Theory*. Springer, 1992.
10. R.-H. Schulz, *Codierungstheorie*. Vieweg, 2003.
11. J. van Lint, *Introduction to Coding Theory*. Springer, 1999.
12. S. A. Vanstone and P. C. Oorschot, *An Introduction to Error-Correcting Codes with Applications*. Kluwer, 1989.
13. W. Willems. *Codierungstheorie*. De Gruyter, 1999.

Einführung

Codierung: Sicherung von Daten und Nachrichten gegen zufällige Fehler bei der Übertragung oder Speicherung.

Situation:

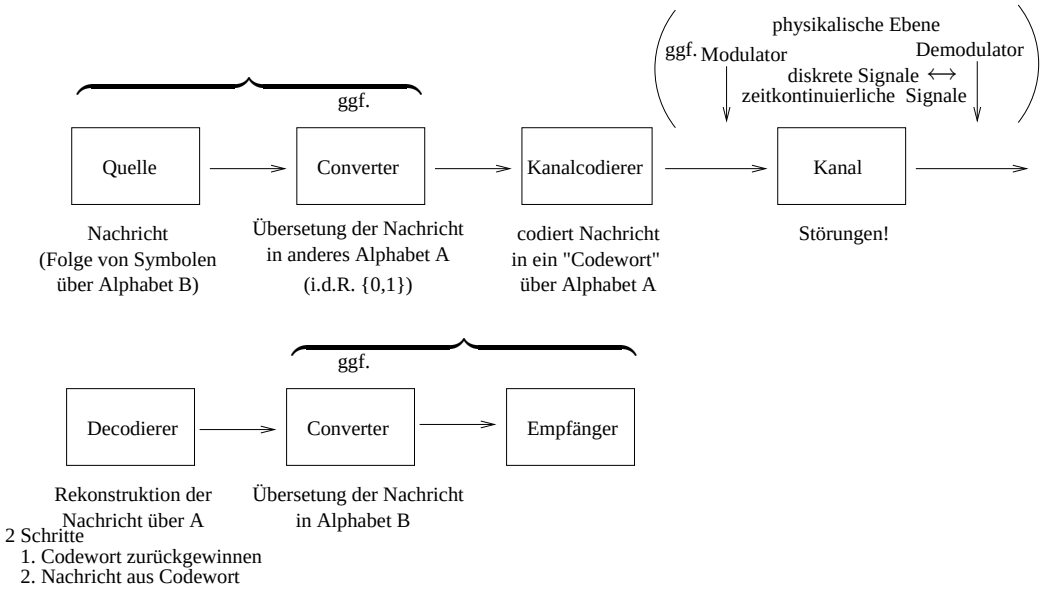


Abbildung 1: Schema der Kanalcodierung

Ziele bei der Codierung:

1. Möglichst viele bei der Übertragung (oder Speicherung) aufgetretene (zufällige) Fehler sollen bei der Decodierung erkannt werden und evtl. sogar korrigiert werden.

2. Der Aufwand zur Codierung und Decodierung soll nicht zu groß sein.

Verwandte Gebiete:

1. **Quellencodierung**: Nachricht wird so codiert, dass häufig auftretende Strings in kurzer Form codiert werden und seltenere in längerer Form (Datenkompression).

2. **Kryptographie**: Sicherung von Nachrichten/Daten gegen Abhören oder Änderungen durch unbefugte Dritte (Datenverschlüsselung).

In der Praxis kann eine Kombination aller drei Verfahren vorkommen: Zunächst wird eine Nachricht komprimiert (z.B. *.zip), dann verschlüsselt und anschließend codiert. In dieser Vorlesung werden Datenkompression und Kryptographie nicht behandelt, sondern nur die Kanalcodierung.

Grundprinzip der Kanalcodierung: Füge der zu übertragenen Nachricht Redundanz zu, so dass auftretende Fehler (häufig) erkannt werden können.

Zwei Hauptaufgaben:

1. Konstruktion geeigneter Codes zur Fehlererkennung und ggf. -korrektur

   Anforderung: Gute Fehlererkennungs, -korrektureigenschaften mit möglichst geringer Redundanz

2. Konstruktion von Codierern, Decodierern

   Anforderung: Effizienz

In der Praxis werden im wesentlichen zwei Typen von Codes unterschieden: Blockcodes und Faltungscodes (convolutional codes).
Wir werden beide Typen in der Vorlesung behandeln.

Ferner gibt es zwei unterschiedliche Prinzipien bei der Kanalcodierung:

FEC-Verfahren (Forward Error Correction)
Aufgetretene Fehler werden (aufgrund der zugefügten Redundanz) erkannt und korrigiert. (Vorteil: keine Verzögerung bei Übertragung; aber gegebenenfalls große Redundanz notwendig!)

ARQ-Verfahren (Automatic Repeat Request)
Aufgetretene Fehler sollen erkannt werden, werden aber nicht korrigiert. Stattdessen wird eine Wiederholung der Übertragung beim Sender angefordert. (Vorteil: geringe Redundanz; aber ggf. erhebliche Verzögerung bei wiederholter Übertragung.)

1 Codes - einige einfache Beispiele

Redundanz als Möglichkeit der Fehlererkennung und -korrektur kennzeichnet schon natürliche Sprachen. Wir werden jetzt einige einfache Beispiele beschreiben, an denen schon eine Reihe wichtiger Prinzipien der Kanalcodierung sichtbar werden.

1.1 Parity-Check Code

Beispiel: Die Nachrichten sind 00, 01, 10, 11 über Alphabet $A = \{0, 1\}$.
Sie werden über demselben Alphabet auf folgende Weise codiert:

$$\begin{aligned} 00 &\to 000 \\ 01 &\to 011 \\ 10 &\to 101 \\ 11 &\to 110 \end{aligned}$$

Es gibt acht 0-1-Wörter der Länge 3, von denen nur die mit einer geraden Anzahl von Einsen tatsächlich Codewörter sind. Entsteht bei der Übertragung eines Codewortes genau ein Fehler, so ist die Anzahl der Einsen ungerade und der Fehler wird entdeckt. Man hat jedoch keinen Anhaltspunkt, in welcher Koordinate der Fehler entstanden ist. Damit ist eine Decodierung nicht eindeutig möglich. Zwei Fehler in einem übertragenen Codewort werden nicht entdeckt.
Der ASCII-Code ist von diesem Typ: 128 Zeichen werden mit 8 Bits codiert, wobei das achte Bit ein Parity-Check-Bit ist.

1.2 Wiederholungscode

Nachrichten und Alphabet wie in 1.1

$$\begin{aligned} 00 &\to 000000 \\ 01 &\to 010101 \\ 10 &\to 101010 \\ 11 &\to 111111 \end{aligned}$$

3-facher Wiederholungscode (große Redundanz)
Zwei verschiedene Codewörter unterscheiden sich an mindestens 3 verschiedenen Stellen. Tritt bei der Übertragung eines Codewortes nur ein Fehler auf, so kann er korrigiert werden. Wähle Codewort, dass sich an möglichst wenigen Stellen vom empfangenen Wort unterscheidet. Bei Auftreten von zwei Fehlern wird erkannt, dass Fehler aufgetreten sind. Mit obiger Methode wird aber nicht korrekt decodiert.

1.3

Nachrichten und Alphabet wie in 1.1/1.2
Codierung:

$$00 \to 00000$$
$$01 \to 01101$$
$$10 \to 10110$$
$$11 \to 11011$$

Geringere Redundanz als in 1.2. Aber weiterhein unterscheiden sich zwei verschiedene Codewörter an mindestens drei Stellen.
ÜA: Ist ein Codierung mit den Eigenschaften aus 1.2,1.3 auch mit Codewörtern der Länge 4 möglich?

1.4 Prüfziffern-Code : ISBN

Die *International Standard Book Number* ist ein 10-stelliger Code, der jedes Buch international erkennbar macht. Die ersten 9 Ziffern kennzeichnen Erscheinungsland, Verlag und Buch; an diese wird eine zehnte Prüfziffer angehängt (Redundanz).

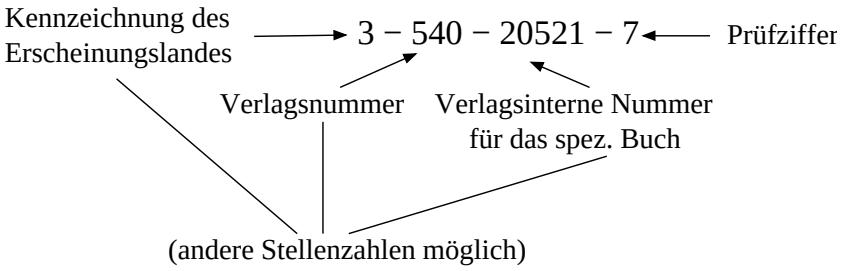


Abbildung 2: Beispiel einer ISBN-Nummer

Uncodierte Wörter sind aus dem Alphabet B = {0, . . . , 9} und Codewörter sind aus dem Alphabet A = {0, 1, . . . , 9, X}. Anhängen der Prüfziffer so, dass für eine Prüfziffer $z_{10}z_9 \dots z_2z_1$ gilt:

$$\sum_{k=1}^{10} kz_k \equiv 0 \mod 11$$

Ist dabei $z_1 = 10$, so wird $z_1 = \mathrm{X}$ gesetzt.
Im obigen Beispiel: $10\cdot3+9\cdot5+8\cdot4+7\cdot0+6\cdot2+5\cdot0+4\cdot5+3\cdot2+2\cdot1+1\cdot7 = 30+45+32+12+20+6+2+7 = 154 \equiv 0 \mod 11$.

Die Redundanz ist bei dieser Codierung sehr gering und dazu werden die zwei häufigsten Fehlertypen bei Lesen oder Abtippen erkannt:

(a) Ändern einer Ziffer (80% aller Fehler)
Wird z_i durch x_i ersetzt, dann:

$$\sum_{\substack{k=1\\k\neq i}}^{10} kz_k + ix_i = \sum_{k=1}^{10} kz_k - i\underbrace{(z_i - x_i)}_{\not\equiv\ 0\mod 11} \not\equiv\ 0 \mod 11$$

(b) Vertauschen zweier Ziffern: z_i und z_j werden vertauscht

$$\sum_{\substack{k=1\\k\neq i,j}} kz_k + iz_j + jz_i = \sum_{k=1}^{10} kz_k + \underbrace{(j-i)(z_i - z_j)}_{\not\equiv\ 0 \mod 11} \not\equiv\ 0 \mod 11$$

Denn für $z_i \neq z_j$ gilt: $1 \leq |z_i - z_j| \leq 10$ und $1 \leq |i-j| \leq 10$ (wesentlich: 11 Primzahl).

1.5 Prüfziffern-Code: EAN-13

Die *Europäische Artikelnummer* ist ein 13-stelliger Code, der sich auf vielen Produktverpackungen befindet.
Nachrichten und Codewörter sind aus dem Alphabet A = B = $\{0, \ldots, 9\}$.
Die gültigen Codewörter sind aus der folgenden Menge:

$$\mathcal{C} = \{c_1 \ldots c_{13}\ ;\ c_i \in A,\ c_1 + 3c_2 + c_3 + 3c_4 + \ldots + 3c_{12} + c_{13} \equiv\ 0 \mod\ 10\}$$

Die eigentliche Information steckt in den zwölf Ziffern $c_1, \ldots, c_{12}$, von denen die ersten beiden das Herstellungsland angeben (40-43 Deutschland). Die Ziffern c_3 bis c_{12} sind für die Herstellungsfirma reserviert (in der Regel geben c_3 bis c_7 den Hersteller und c_8 bis c_{12} das Produkt an). Zur Codierung wird eine 13te Ziffer angehängt. Diese wird eindeutig so gewählt, dass sich ein gültiges Codewort ergibt.

Beispiel: $\underbrace{40}_{\text{Land}}\ \underbrace{12345}_{\text{Firma}}\ \underbrace{10010}_{\text{Produkt}}$

$4 + 1 + 3 \cdot 2 + 4 + 3 \cdot 4 + 5 + 3\dot{1} + 1 = 35 \Rightarrow c_{13} = 5$
Gültige EAN-13-Nummer: 4012345100105

EAN-13 erkennt, wenn eine Ziffer verändert wurde:
$x \to 3x \mod\ 10$ ist eine Permutation auf A.

EAN-13 erkennt Vertauschungen von c_i und c_j $(i \neq j)$, außer wenn der Abstand der Indizes i und j gerade ist oder i gerade, j ungerade (oder umgekehrt) und $|c_i - c_j| = 5$.
Erkennung von Zahlendrehern ist nur gesichert, falls je zwei benachbarte Ziffern Abstand $\neq 5$ haben.

EAN-13-Code ist kompatibel mit dem amerikanischem UPC-A Code (12-stellig): Stelle UPC-A-Nr. eine 0 voran (daher: Ländernummern mit führender Null bezeichnen USA, Kanada)
Der EAN-13 Code wird in einen Barcode/Strichcode übersetzt, der optisch-elektronisch mit einem Scanner gelesen werden kann.
Umsetzung EAN-13-Code in Strichcode:
$1 \mathrel{\hat{=}}$ schwarzer Balken, $0 \mathrel{\hat{=}}$ weißer Balken
Ziffern $c_2, \ldots, c_7$ (linke Hälfte) werden nach Code A oder Code B binär codiert.
Welcher der Codes verwendet wird, wird durch Ziffer c_1 bestimmt; sie bestimmt einen Code D, der angibt, mit welchem der Codes A und B jeweils die Ziffern $c_2, \ldots, c_7$ zu codieren sind.
$c_8, \ldots, c_{13}$ (reche Hälfte) werden mit Code C codiert.

Näheres: http:// www.barcodeisland.com/ean13.phtml
Mehr über Prüfziffersysteme: Schulz, Codierungstheorie, Kapitel 8

	Ziffern $c_2 - c_7$		Ziffern $c_8 - c_{13}$	bestimmt durch c_1
Zeichen	Code A	Code B	Code C	Code D
0	0001101	0100111	1110010	AAAAAA
1	0011001	0110011	1100110	AABABB
2	0010011	0011011	1101100	AABBAB
3	0111101	0100001	1000010	AABBBA
4	0100011	0011101	1011100	ABAABB
5	0110001	0111001	1001110	ABBAAB
6	0101111	0000101	1010000	ABBBAA
7	0111011	0010001	1000100	ABABAB
8	0110111	0001001	1001000	ABABBA
9	0001011	0010111	1110100	ABBABA

Tabelle 1: Die vier Codes des EAN-13

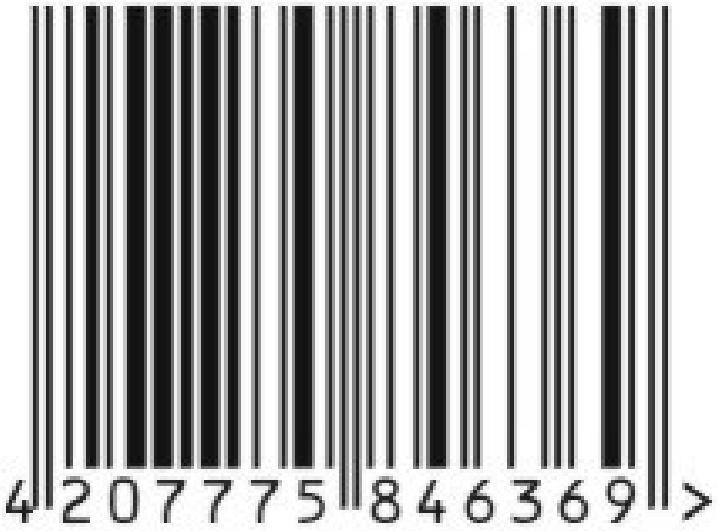


Abbildung 3: Beispiel eines EAN-13 Codewortes

2 Blockcodes - grundlegende Definitionen

Wesentlich an den Beispielen aus Kapitel 1:
Zur Codierung wurden Wörter über einem Alphabet A von fester Länge n gebildet, wobei nur gewisse aller möglichen Wörter der Länge n über A tatsächlich Codewörter sind. Diese Tatsache lässt sich, wie gesehen, zum Entdecken und ggf. Korrigieren von Fehlern ausnutzen.
Wir geben eine formale Definition:

2.1 Definition.
Es sei A eine endliche Menge (Alphabet) und $n \in \mathbb{N}$. Ein *Blockcode* $\mathcal{C}$ der *(Block-)Länge* n über A ist eine Teilmenge von $A^n = \underbrace{A \times \ldots \times A}_{\leftarrow n \rightarrow}$.

Die Elemente von $\mathcal{C}$ heißen *Codewörter*.
Ist $|A| = 2$ (i.Allg. $A = \{0, 1\}$), so heißt $\mathcal{C}$ *binärer (Block-)Code*.

(Solange wir uns mit Blockcodes beschäftigen, sagen wir auch kurz 'Code' statt 'Blockcode')

Sei $\mathcal{C}$ ein Blockcode mit m Codewörtern, d.h. $|\mathcal{C}| = m$ (klar $m \leq |A|^n$).
Dann lassen sich 'Alphabete' B mit m vielen 'Buchstaben' codieren. (Diese Alphabete können selbst aus Strings über einem anderen Alphabet bestehen, z.B. 00,01,10,11; vgl. Beispiele 1.1-1.3). Folgen von Buchstaben aus B (d.h. Wörter über B) werden dann in Folgen von Codewörtern codiert.
Die Zuordnung: 'Element von B $\rightarrow$ Codewort' ist die Codierungsfunktion.

2.2 Definition.
Sei A ein endliches Alphabet, $n \in \mathbb{N}$.
Für $a = (a_1, \ldots, a_n),\ b = (b_1, \ldots, b_n) \in A^n$ sei

$$d(a, b) = \#\{i :\ 1 \leq i \leq n,\ a_i \neq b_i\}$$

d heißt *Hamming-Abstand* von a und b.

(Richard W. Hamming, 1915-1998; Bell Labs, dann Professor für Computer Science an der Naval Postgraduate School Monterey, USA; Mitbegründer der Codierungstheorie mit einer Arbeit aus dem Jahre 1950.)

2.3 Satz.
Für den Hamming-Abstand gilt:

(1) $0 \leq d(a, b) \leq n$ *für alle* $a, b \in A^n$

(2) $d(a,b) = 0 \Leftrightarrow a = b$ *für alle* $a, b \in A^n$

(3) $d(a,b) = d(b,a)$ *für alle* $a, b \in A^n$

(4) $d(a,b) \leq d(a,c) + d(c,b)$ *für alle* $a, b, c \in A^n$ *(Dreiecks-Ungleichung)*

Damit ist d eine Metrik.
Ist A eine kommutative Gruppe (bezgl. +), so gilt ferner:

(5) $d(a+c, b+c) = d(a,b)$ *für alle* $a, b, c \in A^n$ *(Translations-Invarianz)*

Beweis. Nur (4): Ist $a_i \neq b_i$, so ist $a_i \neq c_i$ oder $b_i \neq c_i$. Daraus folgt die Behauptung.

□

Beachte:
Wird ein Wort $x \in \mathcal{C}$, $\mathcal{C}$ ein Code, gesendet und als Wort $y \in A^n$ empfangen mit $d(x,y) = k$, so sind k Fehler während der Übertragung aufgetreten.

2.4 Definition.

(a) *Hamming-Decodierung* (Maximum-Likelihood-Decodierung) für einen Blockcode $\mathcal{C}$ in A^n.
Wird ein Wort $y \in A^n$ empfangen, so wird y als ein x' decodiert mit

$$d(x', y) = \min_{x \in \mathcal{C}} d(x, y).$$

(x' ist im Allgemeinen nicht eindeutig bestimmt!)
Diese Decodierung ist sinnvoll, wenn jedes Symbol in einem Codewort mit gleicher Wahrscheinlichkeit gestört werden kann.

(b) Sei $\mathcal{C}$ ein Blockcode in A^n.
Der *Minimalabstand* von $\mathcal{C}$ ist:

$$d(\mathcal{C}) = \min_{\substack{x, x' \in \mathcal{C} \\ x \neq x'}} d(x, x')$$

(c) Ein Blockcode $\mathcal{C}$ heißt *t-Fehler-korrigierend*, falls

$$d(\mathcal{C}) \geq 2t + 1.$$

Ein Blockcode $\mathcal{C}$ heißt *t-Fehler-erkennend*, falls

$$d(\mathcal{C}) \geq t + 1.$$

Zur Bezeichnung in 2.4(c): Ist $d(\mathcal{C}) \geq 2t+1$ und ist $x \in \mathcal{C}$, so liegt in der 'Kugel'

$$K_t(x) = \{y \in A^n : \ d(x,y) \leq t\}$$

genau ein Codewort - nämlich x.
(Sind $y, y' \in K_t(x)$, so $d(y,y) \leq d(y,x) + d(x,y') \leq 2t$).
Treten also bei der Übermittlung eines Codewortes höchstens t Fehler auf, so wird mit der Hamming-Decodierung korrekt decodiert.

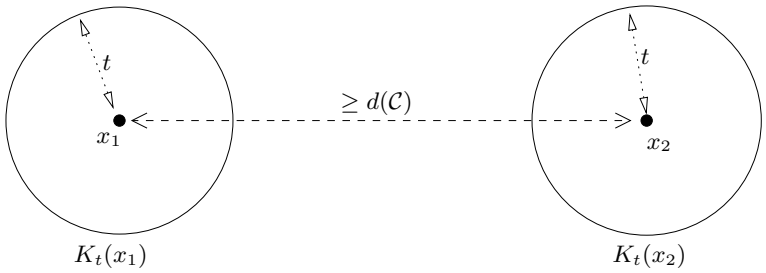


Abbildung 4: Kugeln von Radius t um Codewörter sind disjunkt, falls $d(\mathcal{C}) \geq 2t+1$.

Ist $d(\mathcal{C}) \geq t+1$ und treten bei der Übermittlung eines Codewortes x maximal t Fehler auf, empfangen wird y, so $d(x,y) \leq t$. y ist also kein Codewort, und es wird folglich erkannt, dass Fehler aufgetreten sind.
Hamming-Decodierung kann aber inkorrekt sein, wenn mehr als $\lfloor \frac{t}{2} \rfloor$ Fehler aufgetreten sind. (vgl. Beispiele aus Kapitel 1).

2.5 Beispiele.

(a) m-facher Wiederholungscode: Block der Länge k wird m-fach gesendet. ($n = k \cdot m$)
$d(\mathcal{C}) = m$. $\mathcal{C}$ ist $\lfloor \frac{m-1}{2} \rfloor$-Fehler-korrigierend.
(vgl. Beispiel 1.2; dort $m = 3$).

(b) Code aus Beispiel 1.3: $d(\mathcal{C}) = 3$, $\mathcal{C}$ ist 1-Fehler-korrigierend, 2-Fehler-erkennend.

Ziel der Codierungstheorie:
Gewünscht: Codes $\mathcal{C}$ mit großem Minimalabstand. Andererseits sollte er zur Codierung eines Bits (oder eines Blocks der Länge k) nicht zu viele Bits verwenden (Effizienz).

2.6 Definition.
Sei $\mathcal{C}$ ein Blockcode in A^n, $|A| = q$, d.h. $\mathcal{C}$ ist Blockcode der Länge n über A.

$$R = \frac{1}{n}\ log_q|\mathcal{C}|$$

heißt *Informationsrate* (kurz: *Rate*) von $\mathcal{C}$.
($1 - R$ wird gelegentlich *Redundanz* von $\mathcal{C}$ genannt.)

Beachte: $\log_q|A^n| = n$. Anschaulich gibt die Informationsrate an, welcher Anteil eines Codeworts reine Information ist.

2.7 Beispiele.

(a) m-facher Wiederholungscode. (Blöcke der Länge k m-fach senden) über Alphabet A mit $|A| = q$.
$n = k \cdot m$, $|\mathcal{C}| = q^k$
$\Rightarrow R = \frac{k}{k \cdot m} = \frac{1}{m}$ ($q = 2$: Ein Infobit auf m Bits)

(b) Code aus Beispiel 1.3
4 Codewörter über $\{0, 1\}$ der Länge 5.
$R = \frac{2}{5}$

(c) ASCII-Code: $R = \frac{7}{8}$.

Ziel: Finde Codes mit hoher Rate und kleiner Decodierfehlerwahrscheinlichkeit (bzgl. Hamming-Decodierung).

Das geht 'im Prinzip', wie ein wichtiger Satz von Shannon besagt, den wir im nächsten Kapitel kurz skizzieren.

3 Grundbegriffe der Informationstheorie und der Kanalcodierungssatz von Shannon

Wir gehen (in leichter Änderung der Bezeichnung aus der Einführung) von folgender Situation aus:

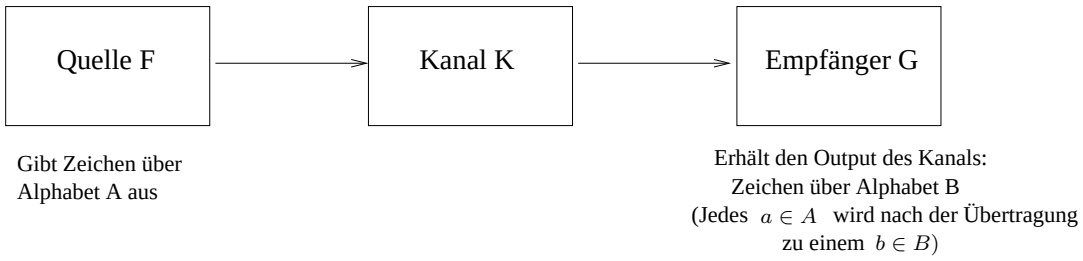


Abbildung 5: Allgemeine Situation.

(Oft A=B, aber auch A $\neq$ B kann sinnvoll sein; z.B. Auslöschung interpretierbar als neues Zeichen '?')

3.1. Voraussetzung

Die Quelle F gebe die Zeichen aus A mit Wahrscheinlichkeit $p_F(a)$, $a \in A$, *aus.*

$$\textit{Also: } 0 \leq p_F(a) \leq 1, \quad \sum_{a \in A} p_F(a) = 1$$

3.2 Definition.

Der *Informationsgehalt* eines von F ausgegebenen Zeichens $a \in A$ ist

$$I_F(a) := \log\left(\frac{1}{p_F(a)}\right) = -\log\left(p_F(a)\right) \in \mathbb{R}_{\geq 0} \cup \{\infty\} \quad (\log = \log_2)$$

Interpretation:

Je größer $p_F(a)$, desto geringer ist die Überraschung, wenn a auftritt. Informationsgehalt=Maß für die Überraschung (Beispiel: Erdbeben, kein Erdbeben)

Dass man als Maß gerade $\log\left(\frac{1}{p_F(a)}\right)$ wählt, hat im Wesentlichen damit zu tun, dass sich für zwei unabhängig voneinander ausgegebene Zeichen der Informationsgehalt addieren soll, dass die Funktion stetig sein soll und man geeignet normiert. Das führt auf $\log\left(\frac{1}{p_F(a)}\right)$.

3.3 Definition.
Die *Entropie* einer Quelle F ist definiert durch

$$H(F) := \sum_{a \in A} p_F(a) I_F(a) = -\sum_{a \in A} p_F(a) \log(p_F(a)) \in \mathbb{R}_{\geq 0}$$

(Ist $p_F(a) = 0$, so definiert man $p_F(a) I_F(a) = 0$; entspricht
$\lim_{x \to 0} x \cdot \log\left(\frac{1}{x}\right) = 0$)

Interpretation:
Die Entropie ist der mittlere Informationsgehalt pro Zeichen einer Quelle F. Die Entropie wird in bit/Symbol gemessen. Sie lässt sich auch so interpretieren: Wieviel Bits werden zur eindeutigen Codierung des Alphabets A mindestens benötigt (wenn man aus einem Bitstring die Elemente aus A eindeutig zurückgewinnen will). Vergleiche Beispiel 3.4c).

3.4 Beispiele. $A = \{a_1, \ldots, a_n\}$

(a) $p_F(a_1) = 1$, $p_F(a_i) = 0$, $i \geq 2$. Dann $H(F) = 0$.

(b) $p_F(a_i) = \frac{1}{n}$, $i = 1, \ldots, n$. Dann $I_F(a_i) = \log(n)$, $i = 1, \ldots, n$,
$H(F) = \log(n)$.

(c) Sei $|A| = 8$
Ist $p_F(a) = \dfrac{1}{8}$ für alle $a \in A$, so $H(F) = 3$ nach b). Kompakteste Codierung von A benötigt 3 Bits pro Symbol.
Sei jetzt $p_F(a_1) = p_F(a_2) = \frac{1}{4}, p_F(a_3) = p_F(a_4) = \frac{1}{8}, p_F(a_5) = \ldots = p_F(a_8) = \frac{1}{16}$
$H(F) = 2 \cdot \frac{1}{4} \cdot 2 + 2 \cdot \frac{1}{8} \cdot 3 + 4 \cdot \frac{1}{16} \cdot 4 = 2,75$

Codierung (z.B. Huffman)

$a_1 \rightarrow 11$
$a_2 \rightarrow 10$
$a_3 \rightarrow 011$
$a_4 \rightarrow 010$
$a_5 \rightarrow 0011$
$a_6 \rightarrow 0010$
$a_7 \rightarrow 0001$
$a_8 \rightarrow 0000$

3.5 Bemerkung.
Für alle Wahrscheinlichkeitsverteilungen p_F gilt: $0 \leq H(F) \leq \log(n)$. Maximum wird bei Gleichverteilung angenommen. (Beweis: z.B. Roman, Theorem 1.2.3).

3.6. Voraussetzung
Der Kanal sein ein sogenannter diskreter gedächnisloser Kanal.
Das bedeutet: Es gibt feste Wahrscheinlichkeiten $p(b|a)$*, wobei* $p(b|a)$ *=Wahrscheinlichkeit, dass* $b \in B$ *empfangen wird, falls* $a \in A$ *gesendet wird.* $p(b|a)$ *ist* <u>*nicht*</u> *abhängig von zuvor gesendeten Symbolen. Man nennt die* $p(b|a)$ *die* Übergangswahrscheinlichkeiten *des Kanals (Bedingte Wahrscheinlichkeiten).*

Klar: $\sum_{b\in B} p(b|a) = 1$.

3.7 Definition.
Die Wahrscheinlichkeitsverteilung p_G beim Empfänger G (*Outputverteilung*) ist gegeben durch

$$p_G(b) = \sum_{a\in A} p(b|a)p_F(a), b \in B.$$

($p_G(b)$ ist die Wahrscheinlichkeit, dass b empfangen wird, wenn ein Symbol aus A von F generiert wurde und durch den Kanal geschickt worden ist.)
Wir haben also auch eine *Outputentropie*:

$$H(G) = -\sum_{b\in B} p_G(b)\log(p_G(b)).$$

3.8 Beispiele.

(a) $A = B$, $p(b|a) = \begin{cases} 0\ , b \neq a \\ 1\ , b = a \end{cases}$
Ungestörter Kanal. Dann ist $H(G) = H(F)$, denn $p_G(a) = p_F(a)$.

(b) $p(b|a) = p_G(b)$ für alle $a \in A$, $b \in B$. Wahrscheinlichkeit, dass b empfangen wird, hängt nicht von dem gesendeten $a \in A$ ab. *Total gestörter Kanal.*
z.B. $A = B = \{0, 1\}$, $p(b|a) = \frac{1}{2}$ für alle $a \in A$, $b \in B$

(c) $A = B = \{0, 1\}$
$p(0|1) = p(1|0) = p$, $p(0|0) = p(1|1) = 1 - p$.

Binärer symmetrischer Kanal. Dies wird für unsere Anwendungen der wichtigste Fall sein.

<u>Wichtige Frage:</u>
Wieviel Information transportiert ein Kanal, der von einer Quelle F gespeist wird, im Mittel pro eingegebenem Zeichen?
Beachte: Außer der Eingangsquelle trägt auch der Kanal (Rauschquelle) zur Entropie des Ausgangs bei. Wir fragen also nach dem Anteil von $H(G)$, der nicht von Kanalstörungen geliefert wird.

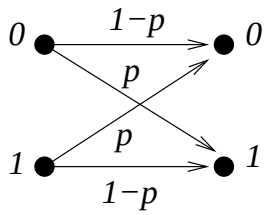


Abbildung 6: Binärer symmetrischer Kanal

3.9 Definition.
Seien $a \in A$, $b \in B$. Der *bedingte Informationsgehalt* von b, falls a gesendet wurde, ist

$$I(b|a) := \log\left(\frac{1}{p(b|a)}\right) = -\log(p(b|a)).$$

Interpretation:
Wenn a eingegeben wurde, erwarten wir mit Wahrscheinlichkeit $p(b|a)$ die Ausgabe b. $I(b|a)$ ist also ausschließlich durch den Kanal verursacht worden.

3.10 Definition.
Der *mittlere bedingte Informationsgehalt* bei Eingabe a (*bedingte Entropie* für a) ist

$$H(G|a) = \sum_{b\in B} p(b|a)I(b|a) = \sum_{b\in B} p(b|a)\log\left(\frac{1}{p(b|a)}\right).$$

Die *Streuentropie* oder *Irrelevanz* des Kanals mit Quelle F ist

$$H(G|F) = \sum_{a\in A} p_F(a)H(G|a) = \sum_{a\in A}\sum_{b\in B} p_F(a)p(b|a)\log\left(\frac{1}{p(b|a)}\right).$$

Interpretation:
$H(G|F)$ ist der Anteil der Entropie von G, die im Mittel (pro Zeiteinheit) vom Kanal (also der Rauschquelle) beigesteuert wird.
Leicht zu sehen: $H(G|F) \leq H(G)$.

3.11 Beispiele.

(a) Ungestörter Kanal: $H(G|F) = \sum_{a\in A} p_F(a)\log(1) = 0$.

(b) Total gestörter Kanal: $H(G|a) = H(G)$ für alle $a \in A$, also auch $H(G|F) = H(G)$.

(c) Binärer symmetrischer Kanal: $H(G|a) = -(1-p)\log(1-p) - p\log p = H(G|F)$.

3.12 Definition.
Die *mittlere Transinformation* eines Kanals mit Ausgang G und Quelle F ist definiert durch

$$I(F,G) := H(G) - H(G|F)$$

Bemerkung.
Man kann leicht zeigen, dass auch gilt: $I(F,G) = H(F) - H(F|G) \leq H(F)$

Interpretation:
$I(F|G)$ gibt an, wieviel nutzbare Information pro Zeiteinheit im Mittel über den Kanal transportiert werden kann (für gegebene Quelle F).

Wir kommen jetzt zur wichtigsten Definition:

3.13 Definition.
(Quellenalphabet A, Ausgangsalphabet B)
Die *Kanalkapazität* C eines Kanals K ist definiert durch

$$C := \max_{\substack{\text{alle W-Vert.} \\ \text{einer Quelle F}}} I(F,G)$$

Bemerkung.
Es ist stets $0 \leq C \overset{\text{Bem. nach 3.12}}{\leq} H(F) \overset{3.5}{\leq} \log(n)$, falls $|A| = n$.

In der Regel ist C schwer zu bestimmen. Für manche Kanäle ist es einfach.

3.14 Beispiele.

(a) K ungestörter Kanal:
$I(F|G) \overset{3.11a)}{=} H(G) \overset{3.8a)}{=} H(F)$, also $C = \log(n)$ nach 3.5.

(b) K total gestörter Kanal:
$C = 0$ nach 3.11b).

Auch für den von uns wichtigsten Kanaltyp, den binären symmetrischen Kanal, lässt sich die Kapazität explizit angeben:

3.15 Satz.
Die Kapazität eines binären symmetrischen Kanals ist

$$C = 1 + p\log(p) + (1-p)\log(1-p) \leq 1 \quad (= \log(2))$$

$$\left[C = 1 \iff \underbrace{p = 0}_{\substack{\text{ungestörter} \\ \text{Kanal}}} \quad \text{oder} \quad \underbrace{p = 1}_{\substack{\text{Jedes Bit} \\ \text{wird umgedreht}}} \quad ; \underbrace{C = 0 \iff p = \frac{1}{2}}_{\text{total gestörter Kanal}} \right]$$

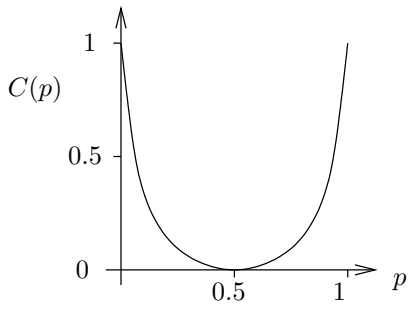


Abbildung 7: Kapazität eines binären symmetrischen Kanals

Beweis:
Da $H(G|F) = -(1-p)\log(1-p) - p\log p$ (3.11c) wird
$I(F,G) := H(G) + (1-p)\log(1-p) + p\log p$ maximal, wenn $H(G)$ maximal wird. Dies ist für $p_G(0) = p_G(1) = \frac{1}{2}$ der Fall.

Aus: $p_G(0) = (1-p)p_F(0) + p \cdot p_F(1)$
$p_G(1) = p \cdot p_F(0) + (1-p)p_F(1)$

folgt dann auch $p_F(0) = p_F(1) = \frac{1}{2}$.
Umgekehrt folgt aus $p_F(0) = p_F(1) = \frac{1}{2}$ auch $p_G(0) = p_G(1) = \frac{1}{2}$.
Dann ist $H(F) = H(G) = \log_2(2) = 1$ und die Behauptung folgt. □

Um den Satz von Shannon formulieren zu können, müssen wir noch ein Wort zu Decodierungsprinzipen sagen:

3.16 Definition (allgemeines Prinzip der Maximum-Likelihood-Decodierung)**.**
(MLD) *Maximum-Likelihood-Decodierung* bei Übertragung von Codeworten über einen beliebigen diskreten gedächnislosen Kanal:
Zu einem empfangenen Wort $y \in B^n$ wird zur Decodierung ein Codewort $\hat{x} \in \mathcal{C} \subseteq A^n$ gewählt, so dass

$$\underbrace{p(y|\hat{x})}_{\prod_{i=1}^n p(y_i|\hat{x}_i)} \geq \underbrace{p(y|x)}_{\prod_{i=1}^n p(y_i|x_i)} \quad \text{für alle } x \in \mathcal{C}$$

Ob das das bestmögliche Decodierungsprinzip ist, hängt von der Wahrscheinlichkeitsverteilung der Codewörter an der Quelle ab.
(Grenzfall: Es wird immer nur ein und dasselbe Codewort gesendet; dann ist natürlich Decodierung in dieses Codewort, egal was empfangen wurde, die beste (und immer korrekte).)

Man kann aber zeigen:
Sind alle Codewörter gleich wahrscheinlich, so ist Maximum-Likelihood-Decodierung die bestmögliche. Aber auch wenn das nicht der Fall ist, ist Maximum-Likelihood-Decodierung häufig die Methode der Wahl; zumal man häufig nicht genau die Quellenstatistik kennt.

3.17 Bemerkung.
Für einen binären symmetrischen Kanal ist die Hamming-Decodierung gerade die Decodierung entsprechend Definition 3.16.
(Daher auch die Wahl der Bezeichnung in 2.4a). Sind also alle Codewörter gleich wahrscheinlich, so ist bei einem binären symmetrischen Kanal die Hamming-Decodierung die bestmögliche.
Nun zum Satz von Shannon: (Claude Shannon, 1916-2001, u.a. MIT)

3.18 Satz (Kanalcodierungssatz von Shannon)**.**
Sei K ein diskreter gedächnisloser Kanal. Sei $|A| = 2$.
Sei C die Kapazität von K. Wähle $\epsilon, \epsilon' > 0$ beliebig.
Dann existiert ein n und ein Blockcode der Länge n über A, dessen Rate R die Bedingung $C - \epsilon' \leq R < C$ erfüllt und wobei die Decodierfehlerwahrscheinlichkeit bei MLD kleiner als ϵ ist.
Umgekehrt:
Ist die Rate R eines Codes größer als C, so kann die Decodierfehlerwahrscheinlichkeit eine gewisse Grenze nicht unterschreiten.

Beweis:
Roman, Kapitel 3.4
Fall des binären symmetrischen Kanals: Friedrichs, Kapitel 2.7.

Beweisidee für binären symmetrischen Kanal
Ist $x \in \{0,1\}^n$ und wird x über Kanal übertragen, so enthält das empfangene Codewort y im Mittel $n \cdot p$ Fehler. Je größer n wird, umso größer ist die Wahrscheinlichkeit, dass die Anzahl der Fehler nahe bei $n \cdot p$ liegt.
Sei $p < \frac{1}{2}$.
Betrachte Kugel vom Radius $n \cdot p$ um x. Sie enthält

$$\sum_{k=0}^{n \cdot p} \binom{n}{k} \text{ viele Wörter.}$$

Eine einfache Abschätzung zeigt:

$$\sum_{k=0}^{n\cdot p} \binom{n}{k} \approx 2^{n(\overbrace{-p\log p-(1-p)\log(1-p)}^{=:H(p)})} \quad \text{(siehe z.B. Friedrichs, Anhang A.2)}$$

Könnte man $\{0,1\}^n$ in disjunkte Kugeln vom Radius np zerlegen, so erhält man

$$\frac{2^n}{2^{nH(p)}} = 2^{n(1-H(p))} \text{ viele Kugeln.}$$

Wählt man die Mittelpunkte dieser Kugeln als Codewörter, so wird (bei großem n) fast immer korrekt decodiert. Die Rate des Codes ist

$$\frac{\log(2^{n(1-H(p))})}{n} = 1 - H(p) = C.$$

Für den exakten Beweis, vergrößert man den Radius der Kugeln geringfügig in geeigneter von ϵ' und p abhängiger Weise und zeigt, dass per Zufallswahl von $R \cdot n$ $(R < C)$ vielen Wörtern in $\{0,1\}^n$ der Erwartungswert für einen Decodierfehler beliebig klein wird. Daher muss es auch mindestens einen solchen Code geben. □

(Also: So gut wie alle Codes erfüllen den Shannon-Satz!)

Ziel im Folgenden: Möglichst gute Codes explizit konstruieren, die auch schnell zu decodieren sind.
(Dies ist der Ansatz von Hamming).

4 Die Kugelpackungsschranke und perfekte Codes

Blockcode $\mathcal{C}$ der Länge n über Alphabet A mit Minimalabstand d. Dann sind Kugeln vom Radius $\left\lfloor \frac{d-1}{2} \right\rfloor$ um die Codewörter disjunkt. Liegt jedes Element in einer dieser Kugeln, so lässt sich jedes Element von A^n bezüglich der Hamming-Decodierung eindeutig decodieren. Sei im Folgenden

$$K_t(x) = \{y \in A^n : d(x,y) \leq t\}$$

eine Kugel vom Radius t um x bezüglich Hamming-Metrik.

4.1 Definition (perfekter Code)**.**
Sei $\mathcal{C}$ ein Code der Länge n über A. $\mathcal{C}$ heißt *perfekt*, falls ein $e \in \mathbb{N}$ existiert mit:

1. $K_e(x) \cap K_e(x') = \emptyset \quad \forall\ x, x' \in \mathcal{C},\ x \neq x'$
2. $A^n = \bigcup_{x \in \mathcal{C}} K_e(x)$.

4.2 Satz.
Sei $\mathcal{C}$ ein perfekter Code mit $|\mathcal{C}| > 1$ und e wie in 4.1.
Dann gilt

$$d(\mathcal{C}) = 2e + 1.$$

Perfekte Codes haben also ungeraden Minimalabstand.

Beweis. Klar: $d(\mathcal{C}) > e$, da $|\mathcal{C}| > 1$.
1. Seien $x, x' \in \mathcal{C}$ $x \neq x'$ (ex. da $|\mathcal{C}| > 1$). Angenommen $d(x,x') \leq 2e$. Ändere x an e Positionen (beachte $d(\mathcal{C}) > e$), an denen sich x von x' unterscheidet, auf solche Weise, dass die betreffenden Einträge danach mit denen von x' übereinstimmen. Sei $y \in A^n$ das dadurch entstandene Wort. Dann ist $d(x,y) = e$. Da $d(x,x') \leq 2e$, folgt $d(x',y) \leq e$. Damit ist $y \in K_e(x) \cap K_e(x')$. Widerspruch zu Eigenschaft 1 eines perfekten Codes. Daher ist $d(\mathcal{C}) \geq 2e+1$.

2. Wähle ein $x \in \mathcal{C}$ und ändere es an $e+1$ Stellen zu $z \in A^n$ ab. Dann gilt $d(x,z) = e+1$ und $z \notin K_e(x)$. Da der Code perfekt ist, existiert $x' \in \mathcal{C},\ x \neq x'$ mit $d(x',z) \leq e$ ($z \in K_e(x')$). Es gilt dann:
$d(\mathcal{C}) \leq d(x,x') \leq d(x,z) + d(z,x') \leq (e+1) + e = 2e + 1$.
Insgesamt ist somit $d(\mathcal{C}) = 2e + 1$. □

4.3 Lemma.
Sei $|A| = q$, $y \in A^n$ und $e \in \mathbb{N}_0$ ($e \leq n$). Dann ist

$$|K_e(y)| = \sum_{j=0}^{e} \binom{n}{j} (q-1)^j$$

Beweis.
In $K_e(y)$ liegen alle Elemente aus A^n, die Abstand $0, 1, \ldots, e$ von y haben. Sei $0 \leq j \leq e$. Dann gibt es $\binom{n}{j}$ Möglichkeiten j Positionen aus den n Positionen des Wortes y auszuwählen. Für jede Auswahl gibt es $(q-1)^j$ Möglichkeiten, ein Element aus A^n zu bilden, welches sich von y an genau diesen j Positionen unterscheidet. Das heisst:

$$|\{y' \in A^n : \ d(y,y') = j\}| = \underbrace{\binom{n}{j}}_{\substack{\text{Anzahl der} \\ \text{Auswahl von } j \\ \text{Positionen in } y}} \quad \underbrace{(q-1)^j}_{\substack{\text{Anzahl der Werte, die} \\ \text{sich an vorgeg. } j \text{ Stellen} \\ \text{von } y \text{ unterscheiden.}}} .$$

Daraus folgt die Behauptung. □

4.4 Satz.
Sei $\mathcal{C}$ ein Code der Länge n über A, $|A| = q$. Sei $e \in \mathbb{N}_0$ maximal mit $d(\mathcal{C}) \geq 2e+1$.

(a) (Hamming-Schranke, Kugelpackungsschranke) Es gilt:

$$|\mathcal{C}| \leq \frac{q^n}{\sum_{j=0}^{e} \binom{n}{j}(q-1)^j}$$

(b) $\mathcal{C}$ ist genau dann ein perfekter Code, wenn in (a) die Gleichheit gilt:

$$|\mathcal{C}| = \frac{q^n}{\sum_{j=0}^{e} \binom{n}{j}(q-1)^j}$$

(Dann ist $d(\mathcal{C}) = 2e+1$).

Beweis.

(a) Wegen $d(\mathcal{C}) \geq 2e+1$ ist $K_e(x) \cap K_e(x') = \emptyset$ für $x, x' \in \mathcal{C},\ x \neq x'$.
Also: $A^n \supseteq \dot{\bigcup}_{x \in \mathcal{C}} K_e(x)$
Mit 4.3 folgt:

$$q^n = |A^n| \geq |\dot{\bigcup}_{x \in \mathcal{C}} K_e(x)| = \sum_{x \in \mathcal{C}} |K_e(x)| = |\mathcal{C}| \sum_{j=0}^{e} \binom{n}{j}(q-1)^j$$

Dies liefert die Behauptung.

(b) Sei $\mathcal{C}$ perfekt. Nach 4.2: $d(\mathcal{C}) = 2e + 1$.

Dann $A^n = \dot{\bigcup}_{x \in \mathcal{C}} |K_e(x)|$. also $q^n = |\mathcal{C}| \sum_{j=0}^{e} \binom{n}{j} (q-1)^j$.
Umgekehrt existiere $e \in \mathbb{N}_0$ mit

$$|\mathcal{C}| = \frac{q^n}{\sum_{j=0}^{e} \binom{n}{j} (q-1)^j}.$$

Dann überdecken wegen der Disjunkheit die Kugeln vom Radius e um die Codewörter ganz A^n. $\mathcal{C}$ ist perfekt.

□

Bemerkung.
Die folgenden Codes werden triviale perfekte Codes genannt:

- einelementige Codes
- $\mathcal{C} = A^n$ $(e = 0)$
- n-fache Wiederholungscodes $\mathcal{C} = \{\underbrace{(1, \ldots, 1)}_{2e+1}, \underbrace{(0, \ldots, 0)}_{2e+1}\}$, $n = 2e + 1$.

4.5 Beispiel.
Sei $A = \{0, 1\} = \mathbb{Z}_2$ (Körper). Wir geben einen (linearen) Code $\mathcal{C}$ der Länge 7 über $\mathbb{Z}_2$ an (d.h. $\mathcal{C} \subseteq \mathbb{Z}_2^7$), der perfekt mit $d(\mathcal{C}) = 3$ ist und $16 = 2^4$ Codewörter enthält.

$$\begin{aligned} \mathcal{C} = \{(c_1 \ldots c_7) : \ c_i \in \mathbb{Z}_2, \ & c_1 + c_4 + c_6 + c_7 = 0, \\ & c_2 + c_4 + c_5 + c_7 = 0, \\ & c_3 + c_5 + c_6 + c_7 = 0\}. \end{aligned}$$

$\mathcal{C}$ ist ein $\mathbb{Z}_2$-Unterraum der Dimension 4. Werden $c_4, \ldots, c_7$ frei gewählt, so sind c_1, c_2, c_3 eindeutig bestimmt. Anhand der definierenden Gleichungen ergibt sich:

$$\mathcal{C} = < \underbrace{(1101000), (0110100), (1010010), (1110001)}_{\text{Basis von } \mathcal{C}} >$$

$|\mathcal{C}| = 2^4 = 16$
Da $\mathcal{C}$ eine additive Gruppe ist und die Hamming-Distanz d translationsinvariant ist, ist

$$d(\mathcal{C}) = \min_{x \in \mathcal{C}, x \neq 0} d(0, x)$$

und es folgt, dass $d(\mathcal{C})$ die Minimalanzahl von Einsen in einem nichttrivialen Codewort ist.
Jedes Codewort$\neq 0$ enthält jedoch mindestens 3 von Null verschiedene Einträge:
$c_7 = 1 \overset{\text{Add. der drei Gl.}}{\Rightarrow} c_1 + c_2 + c_3 = 1.$
$c_1 = 1, c_2 = c_3 = 0$, so (2.Gl.) $c_4 + c_5 = 1$
$c_2 = 1, c_1 = c_3 = 0$, so (1. Gl) $c_4 + c_6 = 1$
$c_3 = 1, c_1 = c_2 = 0$, so (1. Gl) $c_4 + c_6 = 1$

$c_7 = 0 \overset{\text{Add. der drei Gl.}}{\Rightarrow} c_1 + c_2 + c_3 = 0.$
$c_1 = c_2 = c_3 = 0 \Rightarrow c_4 = c_5 = c_6 = 1$
$c_1 = c_2 = 1$ oder $c_1 = c_3 = 1 \overset{\text{1. Gl.}}{\Rightarrow} c_4 + c_6 = 1$
$c_2 = c_3 = 1 \overset{\text{2. Gl.}}{\Rightarrow} c_4 + c_5 = 1$

Da es Codewörter mit 3 Einsen gibt, ist $d(\mathcal{C}) = 3$.
$\mathcal{C}$ ist 1-Fehler-korrigierend. $|K_1(x)| = 1 + 7 = 8$ (4.3).
$\sum_{x\in\mathcal{C}} |K_1(x)| = 16 \cdot 8 = 128 = 2^7 = |\mathbb{Z}_2^7|$.
Also ist $\mathcal{C}$ perfekt nach 4.4b).
$\mathcal{C}$ heißt binärer Hamming-Code der Länge 7 (Shannon, Golay, Hamming, 1948-1950).

4.6 Bemerkung.
Sei $\mathcal{C}$ der binäre Hamming-Code der Länge 7.
Die Rate von $\mathcal{C}$ ist $\frac{4}{7} \approx 0,57$.
Angenommen wir haben einen binären symmetrischen Kanal mit $p = 0,001 = 10^{-3}$. Wie groß ist die Decodierfehlerwahrscheinlichkeit bei Hamming-Decodierung, wenn wir $\mathcal{C}$ verwenden?
Wenn 2 oder mehr Fehler in einem Codewort auftauchen, wird falsch decodiert. Wahrscheinlichkeit für genau 2 Fehler: $p^2(1-p)^5$.
An wievielen Stellen können diese auftauchen? $\binom{7}{2}$.
Also: Wahrscheinlichkeit für 2 Fehler: $\binom{7}{2}p^2(1-p)^5$.
Entsprechend für $3, \ldots, 7$ Fehler.
Also Decodierfehlerwahrscheinlichkeit: $\sum_{k=2}^{7} \binom{7}{k} p^k (1-p)^{7-k}$.
Mit $p = 0,001$ rechnet man aus:
Decodierfehlerwahrscheinlichkeit $\leq 2,1 \cdot 10^{-5}$.
Beachte: Kapazität des Kanals ist $C = 1 + p\log(p) + (1-p)\log(1-p) \approx 0,9887$.
Aber: R ist deutlich kleiner als C. Bei $p = 10^{-3}$ ist Hamming-Code weit weg vom Shannon-Satz.
Für $p = 0,085$ ist $C \approx 0,58$, also R nahe an C. Dann ist aber Decodierfeh-

lerwahrscheinlichkeit $\approx 0,114$.

Bemerkung.

(a) Der binäre Hamming-Code der Länge 7 ist ein Beispiel aus einer Serie perfekter Codes (Hamming-Codes). Für jede Primzahlpotenz q und jede natürliche Zahl $\ell \geq 2$ existiert ein perfekter linearer Code über $\mathbb{F}_q$ (Körper mit q Elementen)

der Länge $\frac{q^\ell - 1}{q-1}$ und der Dimension $\frac{q^\ell - 1}{q-1} - \ell$ mit $d(\mathcal{C}) = 3$.

(Für $\ell = 2$, $q = 2$: Länge 3, $|\mathcal{C}| = 2$, trivialer Code $\{(1,1,1),(0,0,0)\}$). Das obige Beispiel erhält man mit $\ell = 3, q = 2$. Allgemeine Konstruktion später.

(b) Es gibt zwei weitere Beispiele perfekter linearer Codes:

Der binäre Golay-Code $\mathcal{G}_{23}$ über $\mathbb{Z}_2$ hat die Länge 23 und Dimension 12 mit Minimalabstand $d(\mathcal{C}) = 7$ (3-Fehler-korrigierend).

Der ternäre Golay-Code $\mathcal{G}_{11}$ über $\mathbb{Z}_3$ hat die Länge 11 und Dimension 6 (d.h. $|\mathcal{C}| = 3^6$) mit Minimalabstand $d(\mathcal{C}) = 5$ (2-Fehler-korrigierend).

(Zur Konstruktion: van Lint, §4.2,4.3)
(Der binäre Golay-Code wurde für die Voyager I,II-Missionen (1977-81) verwandt um Farbbilder von Jupiter und Saturn zu codieren.)

(c) Die Hamming- und die beiden Golay-Codes sind die einzigen nichttrivialen perfekten linearen (vgl. Kap. 5) Codes (bis auf Äquivalenz- wird später erklärt).
(van Lint (1971), Tietäväinen (1973).

(d) Es gibt nicht-lineare perfekte Codes $\mathcal{C}$ mit $d(\mathcal{C}) = 3$, die die gleichen Parameter haben wie die Hamming-Codes (Vasilev, 1962)

(e) Die Golay-Codes sind die einzigen perfekten Codes über Alphabeten von Primzahlpotenzordnung mit $d(\mathcal{C}) \geq 5$ (van Lint, Tietäväinen).

(f) Der binäre Golay-Code ist der einzige perfekte Code mit $d(\mathcal{C}) \geq 7$ (Best 1978, Hong 1984).

5 Lineare Codes

5.1 Definition.
Sei K ein endlicher Körper und $n \in \mathbb{N}$. Ein *linearer Code* $\mathcal{C}$ ist ein Unterraum des K-Vektorraums K^n.
Ist $\dim(\mathcal{C}) = k\,(\leq n)$ und $d(\mathcal{C}) = d$, so heißt $\mathcal{C}$ ein $[n,k]$-*Code* oder $[n,k,d]$-*Code über* K. n, k, d, q sind die *Parameter* des Codes. (Für $|K| = q$ gibt es auch die Schreibweise $[n,k,d]_q$-Code).

5.2. Endliche Körper

- $\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$, Addition und Multiplikation mod p, ist endlicher Körper, falls p Primzahl.
- Ist K ein endlicher Körper, so ist $|K| = p^m$ für eine Primzahl p und ein $m \in \mathbb{N}$.
- Zu jeder Primzahlpotenz p^m gibt es einen endlichen Körper K mit $|K| = p^m$.
- Sind K, L endliche Körper mit $|K| = |L|$ so ist $K \cong L$. (siehe z.B. Willems, Abschnitt 2.2)
- Konstruktion endlicher Körper: Ist f ein irreduzibles Polynom vom Grad m in $\mathbb{Z}_p[t]$, so ist die Menge der Polynome vom Grad $\leq m-1$ ein Körper der Ordnung p^m, falls Addition wie in $\mathbb{Z}_p[t]$ und Multiplikation modulo f:

$$g \odot f = g \cdot h \mod f \quad \text{(Rest bei Division durch } f\text{)}$$

Beispiel: Körper der Ordnung 4
$\mathbb{Z}_2 = \{0,1\}$, $f = t^2 + t + 1$ ist irreduzibel
$\mathbb{F}_4 = \{0, 1, t, t+1\}$

$\odot$	0	1	t	$t+1$
0	0	0	0	0
1	0	1	t	$t+1$
t	0	t	$t+1$	1
$t+1$	0	$t+1$	1	t

$t^2 \equiv t+1 \mod f$
$t^2 + t \equiv 1 \mod f$
$(t+1)(t+1) = t^2 + 1 \equiv t \mod f$.

5.3 Definition.
K endlicher Körper.

(a) $x \in K^n$, so heißt $wt(x) = d(x,0) = |\{i \mid x_i \neq 0\}|$ das *Gewicht* von x.

(b) Ist $\{0\} \neq \mathcal{C} \subseteq K^n$, so heißt $wt(\mathcal{C}) = \min_{0 \neq x \in \mathcal{C}} wt(x)$ das *Minimalgewicht* von $\mathcal{C}$.

5.4 Satz.
Sei $\mathcal{C} \neq \{0\}$ ein linearer Code über K. Dann gilt:

$$d(\mathcal{C}) = wt(\mathcal{C})$$

(Für $\mathcal{C} = \{0\} : d(\mathcal{C}) = 0$)

Beweis.
Ist $|\mathcal{C}| = 1$, so $\mathcal{C} = \{0\}$, Aussage richtig.
$|\mathcal{C}| > 1$:
Da $\mathcal{C}$ linear ist $(x, x' \in \mathcal{C} \Rightarrow x - x' \in \mathcal{C})$ und der Hamming-Abstand translationsinvariant ist, gilt:

$$\begin{aligned} d(\mathcal{C}) &= \min\{d(x, x') : \ x, x' \in \mathcal{C}, \ x \neq x'\} \\ &= \min\{d(x - x', 0) : \ x, x' \in \mathcal{C}, \ x \neq x'\} \\ &= \min\{d(y, 0) : \ y \in \mathcal{C}, \ y \neq 0\} \\ &= wt(\mathcal{C}) \end{aligned}$$

□

5.5 Definition.
Sei $\mathcal{C}$ ein $[n, k]$-Code über K und
$g_1 = (g_{11}, \ldots, g_{1n}), \ldots, g_k = (g_{k1}, \ldots, g_{kn})$ eine Basis von $\mathcal{C}$. Dann heißt die $k \times n$-Matrix

$$G = \begin{pmatrix} g_1 \\ \vdots \\ g_k \end{pmatrix} = \begin{pmatrix} g_{11} & \cdots & g_{1n} \\ \vdots & & \vdots \\ g_{k1} & \cdots & g_{kn} \end{pmatrix}$$

eine *Erzeugermatrix* für $\mathcal{C}$.

5.6 Satz.
Genau dann ist eine $(k \times n)$-Matrix G eine Erzeugermatrix eines $[n, k]$-Codes $\mathcal{C}$, wenn $\mathcal{C} = \{uG : \underbrace{u}_{Zeilenvektor} \in K^k\}$

Beweis:
$\Rightarrow$: G Erzeugermatrix von $\mathcal{C}$

$$\begin{aligned} \underbrace{(u_1, \ldots, u_k)}_{1 \times k} \underbrace{\begin{pmatrix} g_1 \\ \vdots \\ g_k \end{pmatrix}}_{k \times n} &= (u_1 g_{11} + \ldots + u_k g_{k1}, \ldots, u_1 g_{1n} + \ldots + u_k g_{kn}) \\ &= u_1 g_1 + \ldots + u_k g_k \in \mathcal{C} \end{aligned}$$

$\Leftarrow$: $(1,0,\ldots,0)\begin{pmatrix} g_1 \\ \vdots \\ g_k \end{pmatrix} = g_1, \ldots, (0,\ldots,0,1)\begin{pmatrix} g_1 \\ \vdots \\ g_k \end{pmatrix} = g_k$

Zeilen von G liegen in $\mathcal{C}$
$\mathcal{C} = \{uG : u \in K^k\} = \{u_1 g_1 + \ldots + u_k g_k : u_i \in K\}$
$\mathcal{C} = \langle g_1, \ldots, g_k \rangle$. $\mathcal{C}$ k-dimensional $\Rightarrow$ $g_1, \ldots, g_k$ ist Basis von $\mathcal{C}$. □

5.7 Bemerkung.

(a) G beschreibt injektive lineare Abbildung $K^k \to K^n$; $\mathcal{C}$ ist das Bild. Mit G wird eine Möglichkeit beschrieben, Klartexte in K^k auf Codewörter abzubilden.

(b) Führt man elementare Zeilenoperationen auf G durch, so erhält man wieder eine Erzeugermatrix für $\mathcal{C}$.

5.8 Beispiel.

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

ist eine Erzeugermatrix für den Hamming-$[7,4]$-Code über $\mathbb{Z}_2$ aus Beispiel 4.5.

Andere Erzeugermatrix

$$G \xrightarrow[\text{1.Zeile + 4.Zeile}]{\text{1.Zeile + 3.Zeile}} \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \xrightarrow[\text{2.Zeile + 3.Zeile}]{\text{2.Zeile + 1.Zeile}} \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

$$\xrightarrow[\text{vertauschen}]{\text{3./4. Zeile}} \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \xrightarrow[\text{3.Zeile + 2.Zeile}]{\text{3.Zeile + 1.Zeile}} \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

$$\xrightarrow[\text{4.Zeile + 3.Zeile}]{\text{4.Zeile + 2.Zeile}} \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} =: \tilde{G} \to$$ Erzeugermatrix des binären Hamming-[7.4]-Code.

Verwendet man diese zur Codierung von Klartext (u_1, u_2, u_3, u_4), so $(u_1, u_2, u_3, u_4)\tilde{G} = (u_1, u_2, u_3, u_4, u_1 + u_3 + u_4, u_2 + u_3 + u_4, u_1 + u_2 + u_3)$. Klartext steht an den ersten 4 Stellen des Codewortes.

Wir definieren daher:

5.9 Definition.
Besitzt ein $[n,k]$-Code eine Erzeugermatrix von der Form

$$G = (I_k \mid B) = \begin{pmatrix} 1 & 0 & \dots & 0 & * & \dots & * \\ 0 & 1 & \dots & 0 & * & \dots & * \\ \vdots & & \ddots & \vdots & * & \dots & * \\ 0 & 0 & \dots & 1 & * & \dots & * \end{pmatrix},$$

so heißt eine solche Erzeugermatrix von *Standardform*.
(Bei Codierung: $u \in K^k \to uG \in \mathcal{C}$ ist dann $uG = (u_1, \dots, u_k, * \dots *)$)

Leider besitzt nicht jeder lineare Code eine Erzeugermatrix in Standardform. Aber (später): es gibt immer einen äquivalenten Code, der eine Erzeugermatrix in Standardform besitzt.

5.10 Satz.
Sei $\mathcal{C}$ ein $[n,k]$-Code in K^n. Dann existiert eine $(n-k) \times n$-Matrix H, so dass gilt:

$$x \in K^n : \ x \in \mathcal{C} \Leftrightarrow Hx^t = 0$$

H heißt Kontrollmatrix *von $\mathcal{C}$. Es ist rg(H)= $n-k$.*
(Dann: $HG^t = 0$, falls G Erzeugermatrix des Codes $\mathcal{C}$).

Beweis.
Sei $g_1, \dots, g_k$ eine Basis von $\mathcal{C}$ und $G = \begin{pmatrix} g_1 \\ \vdots \\ g_k \end{pmatrix}$, $g_i = (g_{i1}, \dots, g_{in})$

Betrachte das lineare Gleichungssystem:

$$\begin{array}{rcl} t_1 g_{11} + \dots + t_n g_{1n} & = & 0 \\ \vdots \qquad\qquad\quad \vdots & \vdots & \vdots \\ t_1 g_{k1} + \dots + t_n g_{kn} & = & 0 \end{array}$$

das heißt $G \cdot \begin{pmatrix} t_1 \\ \vdots \\ t_n \end{pmatrix} = 0$ bzw. $(t_1, \dots, t_n) \cdot G^t = 0$

Da Koeffizientenmatrix G den Rang k hat, ist die Dimension des Lösungsraumes $n-k$. Sei $h_1 = (h_{11}, \dots, h_{1n}), \dots, h_{n-k} = (h_{n-k,1}, \dots, h_{n-k,n})$ eine Basis des Lösungsraumes.

Setze $H = \begin{pmatrix} h_1 \\ \vdots \\ h_{n-k} \end{pmatrix}$. Dann gilt $Hg_i^t = \begin{pmatrix} h_1 \cdot g_i^t \\ \vdots \\ h_{n-k} \cdot g_i^t \end{pmatrix} = 0$ für alle $i = 1, \ldots, k$.
Also: $Hx^t = 0$ für alle $x \in \mathcal{C}$.
Da Rang(H)$= n - k$ ist dim Kern(H) = k, das heißt $\mathcal{C}$ =Kern(H). □

5.11 Bemerkung.
Beweis von 5.10 gibt Verfahren an, aus Erzeugermatrix eine Kontrollmatrix zu bestimmen:
Löse Gleichungssystem $Gx = 0$, $x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$.
Basis des Lösungsraumes $h_1^t, \ldots, h_{n-k}^t$ (h_i Zeilenvektoren),
dann $H = \begin{pmatrix} h_1 \\ \vdots \\ h_{n-k} \end{pmatrix}$ Kontrollmatrix.

5.12 Beispiele.

(a) $[7, 4]$-Hamming-Code war definiert durch

$$\mathcal{C} = \{(c_1 \ldots c_7) : \ c_i \in \mathbb{Z}_2, \ c_1 + c_4 + c_6 + c_7 = 0, \ c_2 + c_4 + c_5 + c_7 = 0, \ c_3 + c_5 + c_6 + c_7 = 0\}.$$

Eine Kontrollmatrix ist 3×7-Matrix, die durch diese Gleichungen bestimmt ist:

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

(b) Sei $\mathcal{C}$ Code mit der Erzeugermatrix $G = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$,
$[4, 2]$-Code über $\mathbb{Z}_2$.

Kontrollmatrix:
$$\begin{aligned} t_1 + t_2 + t_4 &= 0 \\ t_2 + t_4 &= 0 \end{aligned}$$

Werden t_3 und t_4 frei gewählt, so sind t_1 und t_2 festgelegt ($t_2 = t_4$, $t_1 = t_4 + t_2$). Damit bilden (0101) und (0010) eine Basis des Lösungsraumes

und es ergibt sich folgende Kontrollmatrix:

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

.

(c) Die Kontrollmatrix für den ASCII-Code ($[8, 7]$-Code über $\mathbb{Z}_2$) ist (1 1 1 1 1 1 1 1). (Summe der 8 Bits =0)

Anhand der Kontrollmatrix eines linearen Codes kann man seinen Minimalabstand bestimmen:

5.13 Satz.
Sei $\mathcal{C} \neq \{0\}$, K^n ein $[n, k]$-Code über K mit Kontrollmatrix H. Dann gilt:

$$\begin{aligned} d(\mathcal{C}) &= wt(\mathcal{C}) \\ &= \min\{r \in \mathbb{N} : \text{ es gibt } r \text{ linear abhängige Spalten in } H \} \\ &= \max\{r \in \mathbb{N} : \text{ je } r-1 \text{ Spalten in } H \text{ sind linear unabhängig}\} \end{aligned}$$

Beweis.
Seien $s_1, \ldots, s_n$ die Spalten von H.
Da $\mathcal{C} \neq \{0\}$ ist, sind $s_1 \ldots, s_n$ linear abhängig (als Vektoren in K^{n-k}).
Sei $w \in \mathbb{N}$ minimal, so dass es w linear abhängige Spalten gibt, etwa $s_{i_1}, \ldots, s_{i_w}$.
Damit existieren $c_j \in K$ mit $\sum_{j=1}^{n} c_j s_j = 0$ und $c_j \neq 0$
genau für $j \in \{i_1, \ldots, i_w\}$.
Setzt man $c = (c_1, \ldots, c_n)$, so ergibt sich $Hc^t = 0$, also $c \in \mathcal{C}$ mit $wt(c) = w$, das heißt $wt(\mathcal{C}) \leq w$.
Angenommen es gebe $0 \neq \overline{c} \in \mathcal{C}$ mit $\overline{w} = wt(\overline{c}) < w$.
Dann folgt aus $H\overline{c}^t = 0$, dass es $\overline{w}$ linear abhängige Spalten in H gibt. Dies ist ein Widerspruch zur Wahl von w. Also $wt(\mathcal{C}) = w$.
2. Gleichheit klar. □

5.14 Beispiel.
Sei $\mathcal{C}$ ein $[7, 3]$-Code über $\mathbb{Z}_2$ mit Erzeugermatrix

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

Um den Minimalabstand zu erhalten, wird die Kontrollmatrix berechnet:

Löse Gleichungssystem $Gx = 0$, $x = \begin{pmatrix} x_1 \\ \vdots \\ x_7 \end{pmatrix}$.

Zunächst werden die Zeilen in G vertauscht:

$$\begin{matrix} g_2 \\ g_3 \\ g_1 \end{matrix} \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix} \qquad \begin{matrix} x_1 & +x_2 & & +x_4 & +x_5 & & +x_7 = 0 \\ & x_2 & & +x_4 & & & +x_7 = 0 \\ & & x_3 & +x_4 & & +x_6 & +x_7 = 0 \end{matrix}$$

Daran lässt sich die Kontrollmatrix ablesen:

$$H = \left(\begin{array}{ccc|cccc} 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{array}\right)$$

Da die 1. und die 5. Spalte linear abhängig sind und es keine Nullspalte gibt, folgt mit 5.13, dass $d(\mathcal{C}) = 2$ [z.B. $wt(g_2 + g_3) = 2$].

5.15 Korollar (Singleton-Schranke)**.**
Ist $\mathcal{C}$ ein linearer $[n, k]$-Code über K, $d(\mathcal{C}) = d$, so

$$d \leq n - k + 1$$

.

Beweis:
Nach 2. Gleichheit in 5.13 gilt: $d \leq rg(H) + 1 = n - k + 1$
(H Kontrollmatrix von $\mathcal{C}$). □

5.16 Definition.
Ein linearer $[n, k, d]$-Code über K heißt *MDS-Code*
(maximum distance separable), falls $d = n - k + 1$

5.17 Bemerkung.
Der Begriff MDS-Code lässt sich folgendermaßen erklären:

(a) Ist $\mathcal{C}$ irgendein $[n, k, d]$-Code, so gibt es Indizes $1 \leq j_1, \ldots, j_k \leq n$, so dass gilt

($\star$) Sind $c = (c_1, \ldots, c_n), c' = (c'_1, \ldots, c'_n) \in \mathcal{C}$
mit $c_{j_1} = c'_{j_1}, \ldots, c_{j_k} = c'_{j_k}$, so ist $c = c'$.

Das heißt Codewörter sind durch ihre Koordinaten $j_1, \ldots, j_k$ eindeutig bestimmt, anders ausgedrückt: Zwei verschiedene Codewörter werden durch ihre Koordinaten $j_1, \ldots, j_k$ getrennt.

Beweis:
Erzeugermatrix G von $\mathcal{C}$ enthält wegen Zeilenrang = Spaltenrang k linear unabhängige Spalten. Seien deren Indizes $j_1, \ldots, j_k$. Sei $\tilde{G}$ die

quadratische $k \times k$-Matrix, die aus G durch Auswahl dieser k Spalten entsteht. Da ihre Spalten linear unabhängig sind, sind auch ihre Zeilen linear unabhängig (Zeilenrang=Spaltenrang). Sind also c und c' zwei Codewörter mit $c_{j_1} = c'_{j_1}, \ldots, c_{j_k} = c'_{j_k}$ und sind $\tilde{c}, \tilde{c}' \in K^k$ diejenigen Wörter, die man aus c und c' erhält, wenn man nur die Koordinaten $j_1, \ldots, j_k$ betrachtet, so ist $\tilde{c} - \tilde{c}' = 0$. Ist $c - c' \neq 0$, so existieren $a_1, \ldots, a_k \in K$ (nicht alle $=0$) mit $c - c' = \sum_{i=1}^{k} a_i g_i$, g_i Zeilen von G. Dann $0 = \tilde{c} - \tilde{c}' = \sum_{i=1}^{k} a_i \tilde{g}_i$, $\tilde{g}_i$ Zeilen von $\tilde{G}$; Widerspruch zur linearen Unabhängigkeit von $\tilde{g}_1, \ldots, \tilde{g}_k$. □

(b) $\mathcal{C}$ ist MDS-Code $\iff$ ($\star$) gilt für alle k-Tupel $(j_1, \ldots, j_k)$ paarweise verschiedener Indizes (d.h. beliebige k Koordinaten trennen zwei Codewörter).

Beweis:
$\Rightarrow$
Nach 5.13 sind wegen $d = n-k+1$ je $n-k$ Spalten einer Kontrollmatrix H von $\mathcal{C}$ linear unabhängig. Seien c, c' wie in ($\star$), $c_{j_i} = c'_{j_i}$, $i = 1, \ldots, k$. Dann $\tilde{c} = c - c' \in \mathcal{C}$, und $\tilde{c}_{j_i} = 0$, $i = 1, \ldots, k$. Seien $s_1, \ldots, s_n$ die Spalten von H.

$$\text{Dann } 0 = H\tilde{c}^t = \sum_{j=1}^{n} \tilde{c}_j s_j = \sum_{\substack{j=1 \\ j \neq j_1, \ldots, j_k}}^{n} \tilde{c}_j s_j$$

Da $\{s_j \mid j \neq j_1, \ldots, j_k\}$ linear unabhängig, folgt $\tilde{c}_j = 0$, $j \neq j_1, \ldots, j_k$. Also $\tilde{c}_j = 0$, $j = 1, \ldots, n$, $c = c'$.

$\Leftarrow$
Ist $0 \neq c = (c_1, \ldots, c_n) \in \mathcal{C}$, so enthält c höchstens $k-1$ viele Nullen (denn sonst unterscheidet es sich vom Nullvektor nicht an k vielen Stellen).
Also: $wt(c) \geq n - (k-1)$ und damit $d \geq n - k + 1$.
Nach 5.13 $d \leq n - k + 1$, also Gleichheit. □

(c) MDS-Codes sind im binären Fall selten; Beispiele über anderen Körpern werden wir später noch kennenlernen (Reed-Solomon-Codes).
Es gilt: (Roman, Kapitel 5.3)
Ist $\mathcal{C}$ binärer $[n, k, d]$-MDS-Code, so ist entweder

$\mathcal{C} = \{(0, \ldots, 0), (1, \ldots, 1)\}$	$(n = d, k = 1)$ oder
$\mathcal{C} = \mathbb{Z}_2^n$	$(n = k, d = 1)$
$\mathcal{C} = \{c = (c_1, \ldots, c_n) \in \mathbb{Z}_2^n : wt(c) \text{ gerade}\}$	$(k = n-1, d = 2)$

Das heißt für $K = \mathbb{Z}_2$ gibt es überhaupt keine interessanten MDS-Codes.

Mit Hilfe von Kontrollmatrizen werden wir jetzt die Hamming-Codes konstruieren, eine unendliche Serie perfekter Codes mit Minimalabstand 3.

5.18 Hamming-Codes.
Sei q eine Primzahlpotenz, K Körper mit $|K| = q$.
Sei $l \in \mathbb{N},\ n = \frac{q^l-1}{q-1}$. Dann existiert ein perfekter $[n, n-l]$-Code $\mathcal{C}$ über K mit $d(\mathcal{C}) = 3$, der *Hamming-Code*.

Konstruktion.
K^l enthält $q^l - 1$ von 0 verschiedene Vektoren. Je $q-1$ von diesen erzeugen den gleichen 1-dimensionalen Unterraum. Also hat K^l genau
$n = \frac{q^l-1}{q-1}$ viele 1-dimensionale Unterräume. Wähle aus jedem einen Vektor $\neq 0$ aus und bilde aus diesen n Spaltenvektoren die $(l \times n)$-Matrix H. Sei $\mathcal{C}$ der Code mit der Kontrollmatrix H:

$$\mathcal{C} = \{x \in K^n;\ Hx^t = 0\} \quad (Hamming - Code)$$

Klar Rang $H = l$, denn H enthält l linear unabhängige Spalten (Basis von K^l), und größer kann der Rang nicht werden. Also $\dim(\mathcal{C}) = n - l$, d.h. $|\mathcal{C}| = q^{n-l}$.
Je zwei Spalten in H sind linear unabhängig, aber es gibt drei linear abhängige Spalten.
5.13: $d(\mathcal{C}) = wt(\mathcal{C}) = 3$.
$\mathcal{C}$ ist perfekt:
$d(\mathcal{C}) = 3 = 2 \cdot 1 + 1$, also $e = 1$ in 4.4.

$$\sum_{j=0}^{1} \binom{n}{j} (q-1)^j = 1 + n(q-1) = 1 + \frac{q^l - 1}{q-1}(q-1) = q^l$$

Daraus folgt:

$$\frac{q^n}{q^l} = q^{n-l} = |\mathcal{C}|$$

und $\mathcal{C}$ ist nach 4.4b) perfekt.
Es gibt jeweils mehrere Möglichkeiten für einen $[n, k]$-Hamming-Code, da die Spaltenvektoren von H nicht eindeutig gewählt werden.

5.19 Beispiele.

(a) $q = 2$ und $l = 3$. Dann ist $n = \frac{2^3-1}{2-1} = 7$.

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

$[7, 4]$-Hamming-Code über $\mathbb{Z}_2$ (vgl. 5.12a)).

(b) $q = 3$ und $l = 3$. Dann ist $n = \frac{3^3-1}{3-1} = 13$.

$$H = \begin{pmatrix} 1 & 0 & 0 & 1 & 2 & 0 & 0 & 1 & 2 & 1 & 2 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 2 & 0 & 0 & 1 & 1 & 2 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 2 \end{pmatrix}$$

$\dim \mathcal{C} = 10$. $[13, 10]$-Hamming-Code über $\mathbb{Z}_3$. $|\mathcal{C}| = 3^{10} = 59.049$.

(c) $q = 2$ und $l = 4$. Dann ist $n = \frac{2^4-1}{2-1} = 15$.

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

$\dim \mathcal{C} = 11$. $[15, 11]$-Hamming-Code über $\mathbb{Z}_2$. $|\mathcal{C}| = 2^{11}$.

Ist $y = (110101110000110) \in \mathcal{C}$?

$$Hy^t = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \neq \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \text{ also } y \notin \mathcal{C}.$$

$\mathcal{C}$ perfekter 1-Fehler-korrigierender Code $\Rightarrow$ Es gibt genau ein Codewort x mit $d(x, y) = 1$.
Wie findet man x? (Decodierverfahren!)
Alle Elemente aus $\mathcal{C}$ zu testen ist langwierig. Es geht schneller:
x unterscheidet sich von y an genau einer Stelle i, statt y_i steht dort y_i+1

$$\text{Also } Hy^t = H(x^t + (0, \ldots, 0, \underbrace{1}_{i}, 0, \ldots, 0)^t) = \begin{pmatrix} h_{1i} \\ h_{2i} \\ h_{3i} \\ h_{41} \end{pmatrix}$$

Also: Suche in H Spalte $\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$ und ändere beim Index dieser Spalte den

Eintrag von y. Dies ist hier Spalte 2.
Decodiere y zu $x = (1\underline{0}0101110000110)$

Dies gilt allgemein:

5.20. Decodierung binärer Hamming-Codes
Empfängt man y und ist $Hy^t = 0$, so lasse y unverändert. Ist $Hy^t \neq 0$, so suche in H diejenige Spalte, die mit Hy^t übereinstimmt. Ist dies die i-te Spalte von H, so ändere Komponente y_i von y in $y_i + 1$. Dies liefert $x \in \mathcal{C}$, zu dem y decodiert wird.
Aufgabe: Wie sieht das Decodierverfahren bei beliebigen Hamming-Codes aus?

Wie decodiert man beliebige lineare $[n,k]_q$-Codes? Wurde y empfangen, so wird dies zu x decodiert mit $d(x,y)$ minimal. Man kann alle q^k Codewörter auf den Abstand zu y testen und von den Codewörtern mit minimalem Abstand eines zur Decodierung auswählen. Dies ist im Allgemeinen nicht eindeutig. Bei großen Codes ($2k > n$) gibt es ein effizienteres Verfahren, die Syndrom-Decodierung.

Exkurs (Nebenklassen von Unterräumen in Vektorräumen).
Sei $\mathcal{C}$ ein Unterraum eines Vektorraums V. Dann heißt für jedes $v \in V$

$$v + \mathcal{C} := \{v + x : \ x \in \mathcal{C}\}$$

die *Nebenklasse* von $\mathcal{C}$ zu v.

(a) Für $v_1, v_2 \in V$ gilt entweder $v_1 + \mathcal{C} = v_2 + \mathcal{C}$ oder $(v_1 + \mathcal{C}) \cap (v_2 + \mathcal{C}) = \emptyset$.
Beweis: Sei $(v_1 + \mathcal{C}) \cap (v_2 + \mathcal{C}) \neq \emptyset$. Dann gibt es ein $y \in (v_1 + \mathcal{C}) \cap (v_2 + \mathcal{C})$ mit $y = v_1 + x_1 = v_2 + x_2$ für $x_1, x_2 \in \mathcal{C}$. Daraus folgt: $v_1 = v_2 + (x_2 - x_1) \in v_2 + \mathcal{C}$. Für beliebiges $x \in \mathcal{C}$ ist auch $v_1 + x = v_2 + (x_2 - x_1) + x \in v_2 + \mathcal{C}$. Also ist $v_1 + \mathcal{C} \subseteq v_2 + \mathcal{C}$. Dasselbe Argument, aber mit 1 und 2 getauscht, ergibt $v_2 + \mathcal{C} \subseteq v_1 + \mathcal{C}$. Insgesamt somit $v_1 + \mathcal{C} = v_2 + \mathcal{C}$.

(b) $v_1 + \mathcal{C} = v_2 + \mathcal{C} \Leftrightarrow v_1 - v_2 \in \mathcal{C}$
Beweis: 1. $v_1 + \mathcal{C} = v_2 + \mathcal{C} \Rightarrow v_1 = v_1 + 0 \in v_1 + \mathcal{C} = v_2 + \mathcal{C} \Rightarrow v_1 = v_2 + x$ für ein $x \in \mathcal{C} \Rightarrow v_1 - v_2 = x \in \mathcal{C}$. 2. Sei $v_1 - v_2 = x \in \mathcal{C} \Rightarrow v_1 = v_2 + x \in (v_1 + \mathcal{C}) \cap (v_2 + \mathcal{C}) \overset{(a)}{\Rightarrow} v_1 + \mathcal{C} = v_2 + \mathcal{C}$.

(c) Für alle $v \in v_1 + \mathcal{C}$ gilt $v + \mathcal{C} = v_1 + \mathcal{C}$.

(d) Wähle aus jeder Nebenklasse von $\mathcal{C}$ einen Vertreter v_i:

$$V = \dot{\bigcup}_i v_i + \mathcal{C} \quad \text{(disjunkte Vereinigung)}$$

(e) Sei V ein Vektorraum über einem endlichen Körper $\mathbb{F}_q$. Dann gilt $|\mathcal{C}| = |v + \mathcal{C}|$ für alle $v \in V$ (da $x \mapsto v + x$ bijektiv ist).

(f) Aus (d) und (e) folgt: Ist $\mathcal{C}$ ein $[n, k]$-Code über $\mathbb{F}_q$, so hat $\mathcal{C}$ genau q^{n-k} viele verschiedene Nebenklassen.

5.21. Syndrom-Decodierung linearer Codes
$\mathcal{C}$ linearer $[n, k]$-Code über K, $|K| = q$, Kontrollmatrix H $((n-k)\times n)$-Matrix.

Ist $y \in K^n$, so heißt $Hy^t \in K^{n-k}$ das *Syndrom* von y.

(a) $x \in \mathcal{C} \Leftrightarrow Hx^t = 0 \Leftrightarrow x$ hat Syndrom 0 (kein "Krankheitsbild").

(b) y_1 und y_2 liegen in der gleichen Nebenklasse bezüglich $\mathcal{C}$ (d.h. $y_1 + \mathcal{C} = y_2 + \mathcal{C}$) $\Leftrightarrow Hy_1^t = Hy_2^t$;
d.h. $y_1, y_2 \in K^n$ liegen genau dann in der gleichen Nebenklasse bzgl. $\mathcal{C}$, wenn sie das gleiche Syndrom haben.

($y_1 + \mathcal{C} = y_2 + \mathcal{C} \Leftrightarrow y_1 - y_2 \in \mathcal{C} \overset{(a)}{\Leftrightarrow} 0 = H(y_1 - y_2)^t = Hy_1^t - Hy_2^t \Leftrightarrow Hy_1^t = Hy_2^t$)

(c) Jedes $z \in K^{n-k}$ tritt als Syndrom auf (denn H bewirkt lineare Abbildung vom Rang $n - k$).

Daher: Wird $x \in \mathcal{C}$ gesendet und wird $y = x+f$ empfangen (f=Fehlervektor), so haben y und f das gleiche Syndrom.
Hamming-Decodierung:
Bestimme in der Nebenklasse von y einen Vektor e von minimalem Gewicht: *Nebenklassenführer* (im Allgemeinen nicht eindeutig).
Decodierung: $y \to y - e$.
Dazu: Man stellt eine Liste der Nebenklassenführer (q^{n-k} viele) auf mit zugehörigem Syndrom. Bei Decodierung sind maximal q^{n-k} viele Vergleiche durchzuführen. Ist $2k > n$, so ist $q^{n-k} < q^k$, das heißt es sind weniger Vergleiche durchzuführen als bei Test aller Codeworte.

Wenn man Nebenklassenführer in lexikographischer Ordnung ihrer Syndrome speichert (dazu braucht man eine Ordnung auf K), so benötigt man die Syndrome nicht mehr. Besonders effizient im binären Fall: Syndrom gibt Stelle $(0, \ldots, 2^{n-k} - 1)$ des gesuchten Nebenklassenführers an.

Das kann immer noch sehr aufwändig sein:
Z.B. $[70, 50]$-Code über $\mathbb{Z}_2$.
$\mathcal{C}$ hat 2^{20} viele Nebenklassen, jeder Nebenklassenführer besteht aus 70 Bits.
Speicherplatzbedarf bei Syndrom-Decodierung (mit lexikographischer Ordnung der Syndrome): $70 \cdot 2^{20}$ Bit $= 8,75$ Megabyte.
Speichert man alle 2^{50} Codewörter, so Speicherplatzbedarf von $70 \cdot 2^{50}$ Bit $\approx$ 9.000 Terabyte.

5.22 Beispiel. (für Syndrom-Decodierung)
$q = 2$, $n = 5$, $k = 3$
Kontrollmatrix $H = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix}$

$$(x_1, x_2, x_3, x_4, x_5) \in \mathcal{C} \iff \begin{array}{l} x_2 + x_3 + x_5 = 0 \\ x_1 + x_3 + x_4 = 0 \end{array}$$

Erzeugermatrix G für $\mathcal{C}$: $\begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix}$

Nebenklassen von $\mathcal{C}$:

1. $(00000) + \mathcal{C} = \mathcal{C} = \{(\underline{00000}), (11100), (10010), (01001),$
   $(01110), (10101), (11011), (00111)\}$
   $d(\mathcal{C}) = 2$ (war schon an H ablesbar)
2. $(10000) + \mathcal{C} = \{(\underline{10000}), (01100), (\underline{00010}), (11001),$
   $(11110), (00101), (01011), (10111)\}$
   Mögliche Nebenklassenführer: $(10000), (00010)$
3. $(01000) + \mathcal{C} = \{(\underline{01000}), (10100), (11010), (\underline{00001}),$
   $(00110), (11101), (10011), (01111)\}$
   Mögliche Nebenklassenführer: $(01000), (00001)$
4. $(11000) + \mathcal{C} = \{(11000), (\underline{00100}), (01010), (10001),$
   $(10110), (01101), (00011), (11111)\}$
   Nebenklassenführer: (00100)

Angenommen:

$$\begin{array}{llll} f_0 = (00000) & \text{Syndrom} & Hf_0^t = \binom{0}{0} \\ f_1 = (10000) & \text{Syndrom} & Hf_1^t = \binom{0}{1} \\ f_2 = (01000) & \text{Syndrom} & Hf_2^t = \binom{1}{0} \\ f_3 = (00100) & \text{Syndrom} & Hf_1^t = \binom{1}{1} \end{array}$$

Empfangen $y = (10011)$ $\quad Hy^t = \binom{1}{0}$.
Decodierung: $y \to y + f_2 = (11011)$.
Wir haben gesehen, dass auch bei Syndrom-Decodierung der Speicherplatz groß sein kann: Man muss alle q^{n-k} Nebenklassenführer (sämtlich Vektoren in K^n, $|K| = q$) speichern. Dazu benötigt man $q^{n-k} \cdot n \cdot \log(q)$ viele Bits.

Ein anderes Verfahren, dass wir jetzt noch besprechen, benötigt nur die Syndrome und die Gewichte der Nebenklassenführer.
Die Speicherung der Syndrome erfordert $q^{n-k} \cdot (n-k) \cdot \log(q)$ viele Bits. Wenn die Gewichte der Nebenklassenführer nicht zu groß sind, ist dieses Verfahren hinsichtlich Speicherbedarf der Syndrom-Decodierung überlegen.

5.23. Schrittweise Decodierung (Step-by-Step Decodierung)
Zu speichern: Syndrome und Gewichte der Nebenklassenführer.
(inklusive Syndrom 0 und Gewicht des zugehörigen Nebenklassenführers $(0, \ldots, 0)$ (für $\mathcal{C}$), das heißt Gewicht 0.)
Wir beschreiben das Verfahren nur für binäre Codes. Sei $y \in \mathbb{Z}_2^n$ das empfangene Wort. Seien $e_1, \ldots, e_n$ die kanonischen Basisvektoren in $\mathbb{Z}_2^n$. Zur Decodierung wenden wir folgenden Algorithmus an:

(1) $i := 1$

(2) Berechne Hy^t und stelle das Gewicht w des Nebenklassenführers mit gleichem Syndrom fest.

(3) Ist $w = 0$, so gebe y als Decodierung aus. stop.

(4) Hat $H(y + e_i)^t$ kleineres zugeordnetes Gewicht des Nebenklassenführers als Hy^t, so ersetze y durch $y + e_i$.

(5) Setze $i := i + 1$. goto (2).

Wir zeigen, dass der Algorithmus terminiert und eine Hamming-Decodierung liefert.
Dazu: Definiere *lexikograpische Ordnung* auf $\mathbb{Z}_2^n$ durch

$$(a_1, \ldots, a_n) < (b_1, \ldots, b_n),$$

falls $a_i = b_i$, $i = 1, \ldots, l$, $a_{l+1} = 1$, $b_{l+1} = 0$ für ein $0 \leq l < n$.

5.24 Satz.
Schrittweise Decodierung (binärer Codes) entspricht der Syndromdecodierung, wenn aus jeder Nebenklasse unter den Vektoren mit minimalem Gewicht der lexikograpisch erste als Nebenklassenführer ausgewählt wird.

Beweis:
Für $z \in \mathbb{Z}_2^n$ sei $d(z, \mathcal{C}) = \min_{x \in \mathcal{C}} d(z, x)$.
Sei $y \in \mathbb{Z}_2^n$ das empfangene Wort, $d(y, \mathcal{C}) = m$.
Sei f der lexikograpisch erste Nebenklassenführer von $y + \mathcal{C}$. Also $wt(f) = m$.
Sei $f = (0, \ldots, 0, \underbrace{1}_{i}, *, \ldots, *)$. Angenommen $d(y + e_j, \mathcal{C}) < m$ für ein $j < i$.
Dann $d(y + e_j, \mathcal{C}) = m - 1$. Daher existiert $\tilde{f}$, $wt(\tilde{f}) = m$,

$\tilde{f} = (*, \ldots, \underbrace{1}_{j}, *, \ldots, *)$ und $y + \tilde{f} \in \mathcal{C}$. Dann ist $\tilde{f}$ ein Nebenklassenführer von $y + \mathcal{C}$ und $\tilde{f}$ kommt lexikograpisch vor f, Widerspruch.
Andererseits $d(y+e_i, \mathcal{C}) = m-1$, denn $(y+e_i)+(f-e_i) \in \mathcal{C}$. In dieser Weise fortfahrend folgt die Behauptung.

□

5.25 Beispiel.
Wir verwenden Beispiel 5.22

Syndrom	$\binom{0}{0}$	$\binom{0}{1}$	$\binom{1}{0}$	$\binom{1}{1}$
Gewicht Nebenklassenführer	0	1	1	2

$$\begin{aligned} y = (10011): &\quad Hy^t = \tbinom{1}{0} \quad wt = 1 \\ y + e_1 = (00011): &\quad H(y+e_1)^t = \tbinom{1}{1} \quad wt = 2 \ (y \text{ nicht verändern}) \\ y + e_2 = (11011): &\quad H(y+e_2)^t = \tbinom{0}{0} \quad wt = 0 \ (y \text{ so decodieren}) \\ & y \to (11011) y + e_2 \in \mathcal{C} \end{aligned}$$

(e_2 ist unter den möglichen Nebenklassenführern $e_2 = (0,1,0,0,0)$ und $(0,0,0,0,1)$ der lexikograpisch erste).

5.26 Bemerkung. Step-by-Step-Decodierung erfordert statt Speicherung der Nebenklassenführer nur deren Gewichte (n Bits vs. maximal $\log(n)$ Bits). Andererseits sind maximal n Syndrom-Berechnungen notwendig gegenüber einer bei Syndrom-Decodierung. Time-Space-Trade-Off!

Zum Abschluss dieses Kapitels gehen wir noch auf den Begriff der Äquivalenz von Codes ein.

5.27 Definition (Isometrie)**.**
Eine invertierbare $n \times n$-Matrix A über K, $|K| = q$, heißt *Isometrie* (bzgl. der Hamming-Metrik) von K^n, falls für alle $u, v \in K^n$ gilt:

$$(*) \quad d(uA, vA) = d(u, v)$$

Wegen $d(uA, vA) = d(uA - vA, 0) = d((u-v)A, 0) = wt((u-v)A)$ ist $(*)$ äquivalent zu

$$(**) \quad wt(uA) = wt(u) \text{ für alle } u \in K^n$$

Beachte: Die Isometrien bilden bezüglich $\cdot$ eine Gruppe.
Es ist leicht zu zeigen:

5.28 Satz.
Jede Isometrie ist von der Form $A = D \cdot P_\pi$, wobei

$$D = \begin{pmatrix} a_1 & & 0 \\ & \ddots & \\ 0 & & a_n \end{pmatrix}$$

eine invertierbare $n \times n$-Diagonalmatrix ist und P_π eine $n \times n$-Permutationsmatrix (zur Permutation π auf $\{1, \ldots, n\}$), das heißt

$$P_\pi = (p_{ij}), \quad p_{ij} = \begin{cases} 1 & : \quad i = \pi(j) \\ 0 & : \quad sonst \end{cases}$$

($P_\pi \cdot P_{\pi'} = P_{\pi \cdot \pi'}$, $P_\pi^{-1} = P_{\pi^{-1}}$)

5.29 Definition.
Zwei Codes $\mathcal{C}, \mathcal{C}' \subseteq K^n$, $|K| = q$, heißen *äquivalent*, $\mathcal{C} \approx \mathcal{C}'$, falls eine Isometrie A existiert mit $\mathcal{C}A = \mathcal{C}'$, das heißt $\{xA \mid x \in \mathcal{C}\} = \mathcal{C}'$

Das bedeutet: $\mathcal{C}'$ entsteht aus $\mathcal{C}$ durch Multiplikation der Komponenten von $\mathcal{C}$ durch Skalare $a_1, \ldots, a_n \neq 0$ und Permutation der Komponenten.

5.30 Bemerkung.

(a) Äquivalente Codes haben die gleichen Parameter (d.h. gleiche Länge, gleiche Anzahl von Elementen und gleichen Minimalabstand).

(b) $\mathcal{C} \approx \mathcal{C}'$, $\mathcal{C}A = \mathcal{C}'$, $\mathcal{C}, \mathcal{C}'$ linear, dann gilt:

$$\begin{aligned} &G \text{ Erzeugermatrix für } \mathcal{C} \ \Rightarrow GA \text{ Erzeugermatrix für } \mathcal{C}' \\ &H \text{ Kontrollmatrix für } \mathcal{C} \ \Rightarrow H(A^t)^{-1} \text{ Kontrollmatrix für } \mathcal{C}' \end{aligned}$$

(GA Erzeugermatrix für $\mathcal{C}'$ ist klar.
$Hx^t = 0 \Leftrightarrow H(A^t)^{-1}A^t x^t = 0 \Leftrightarrow H(A^t)^{-1}(xA)^t = 0$).

(c) Elementare Zeilenumformungen an einer Erzeugermatrix bzw. an einer Kontrollmatrix eines linearen Codes ändern den Code nicht. Spaltenvertauschungen und Multiplikation von Spalten mit Skalaren $\neq 0$ (bei der Erzeuger- oder Kontrollmatrix) ändern den Code zu einem äquivalenten Code.

Insbesondere: Alle Hamming-Codes mit gleichen Parametern sind äquivalent.

(d) Seien $\mathcal{C}$ und $\mathcal{C}'$ lineare Codes der Dimension k über K^n. Angenommen, es existiert eine bijektive K-lineare Abbildung $\varphi : \mathcal{C} \longrightarrow \mathcal{C}'$ mit $wt(c) = wt(\varphi(c))$ für alle $c \in \mathcal{C}$. Dann ist $\mathcal{C} \approx \mathcal{C}'$, d.h. es existiert eine Isometrie A von K^n mit $\mathcal{C}A = \mathcal{C}'$. (Beachte: Umkehrung ist trivial). Dies ist ein Satz von MacWilliams (vgl. Willems, Satz 4.1.2) .

5.31 Satz.
Sei $\mathcal{C}$ ein $[n,k]$-Code über K, $|K| = q$. Dann existiert ein zu $\mathcal{C}$ äquivalenter Code $\mathcal{C}'$, dessen Erzeugermatrix G' in Standard-Form ist, das heißt
$G' = (I_k \mid B)$.

Beweis.
Sei G die Erzeugermatrix von $\mathcal{C}$. Die k Zeilen von G sind linear unabhängig.
„Zeilenrang = Spaltenrang"$\Rightarrow$ G hat k linear unabhängige Spalten.
Durch geeignete Spaltenvertauschungen erhält man eine Matrix $\tilde{G}$, in der die ersten k Spalten linear unabhängig sind.
Durch elementare Zeilenumformungen kann man $\tilde{G}$ auf die Form
$G' = (I_k \mid B)$ bringen.
G und $\tilde{G}$ erzeugen äquivalente Codes und $\tilde{G}$ und G' den gleichen Code. □

5.32 Beispiel.
Die folgende Matrix sei die Erzeugermatrix eines binären $[5,3]$-Codes $\mathcal{C}$:

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

$\mathcal{C}$ hat keine Erzeugermatrix in Standardform, denn jedes Element in $\mathcal{C}$ hat an den ersten beiden Stellen den gleichen Eintrag. Vertausche zuerst die 2te und 4te Spalte und wende elementare Zeilenumformungen an:

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

G' ist in Standardform und erzeugt einen zu $\mathcal{C}$ äquivalenten Code $\mathcal{C}'$.

6 Allgemeine Konstruktionsprinzipien linearer Codes

6.1 Definition.
$|K| = q$. Abbildung $<,>: K^n \times K^n \to K$ ist definiert durch

$$< u, v >:= \sum_{i=1}^{n} u_i v_i = uv^t \quad \text{(vgl. euklidsches Skalarprodukt im } \mathbb{R}^n\text{)}.$$

6.2 Bemerkung.

(a) $< u + v, w > = < u, w > + < v, w >$

(b) $< \lambda u, v > = \lambda < u, v >$

(c) $< u, v > = < v, u >$

Diese Eigenschaften machen $< \cdot, \cdot >$ zu einer *symmetrischen Bilinearform.*

(d) $< 0, v > = < v, 0 > = 0$

(e) Ist $u \in K^n$ mit $< u, v >= 0$ für alle $v \in K^n$, so ist $u = 0$.

$< \cdot, \cdot >$ ist somit eine *nicht-ausgeartete* symmetrische Bilinearform.

Beweis:
a)-d) $\surd$
e) $v = e_i$. $0 =< u, e_i >= u_i$, also $u = 0$. □

6.3 Definition (Dualer Code)**.**
Sei $\mathcal{C} \subseteq K^n$ ($\mathcal{C}$ braucht kein Unterraum zu sein). Dann heißt

$$\mathcal{C}^\perp = \{y \in K^n : < y, x > = 0 \text{ für alle } x \in \mathcal{C}\}$$

der *duale Code* zu $\mathcal{C}$. $\mathcal{C}^\perp$ ist immer ein Unterraum, also ein linearer Code, selbst wenn $\mathcal{C}$ nicht linear ist. $\mathcal{C}^\perp = \langle \mathcal{C} \rangle_K^\perp$.

6.4 Satz.
Sei $\mathcal{C}$ ein $[n, k]$-Code über K, $|K| = q$.

(a) $\mathcal{C}^\perp$ ist ein $[n, n-k]$-Code.

(b) $(\mathcal{C}^\perp)^\perp = \mathcal{C}$

(c) G ist Erzeugermatrix von $\mathcal{C}$ $\Leftrightarrow$ G ist Kontrollmatrix von $\mathcal{C}^\perp$
H ist Kontrollmatrix von $\mathcal{C}$ $\Leftrightarrow$ H ist Erzeugermatrix von $\mathcal{C}^\perp$

(d) Ist $(I_k \mid A)$ eine Erzeugermatrix von $\mathcal{C}$ (in Standardform), so ist $(-A^t \mid I_{n-k})$ eine Erzeugermatrix von $\mathcal{C}^\perp$, also Kontrollmatrix von $\mathcal{C}$.

Beweis.

(a) Sei G Erzeugermatrix von $\mathcal{C}$, $y \in K^n$. Dann gilt:

$$\begin{aligned} y \in \mathcal{C}^\perp &\Leftrightarrow < x, y > = 0 \ \forall \ x \in \mathcal{C} \\ &\Leftrightarrow < x_i, y > = 0 \text{ für eine Basis } x_1, \ldots, x_k \text{ von } \mathcal{C} \\ &\Leftrightarrow Gy^t = 0 \text{ (Zeilen von } G \text{ bilden Basis)} \end{aligned}$$

Damit ist G Kontrollmatrix von $\mathcal{C}^\perp$ mit $\dim(\mathcal{C}^\perp) = n$ - $\mathrm{rg}(G) = n - k$.

(b) Klar, dass $\mathcal{C} \subseteq (\mathcal{C}^\perp)^\perp$. Für die Dimensionen gilt:

$$\dim(\mathcal{C}^\perp)^\perp \overset{(a)}{=} n - (n-k) = \dim(\mathcal{C}).$$

Also gilt die Gleichheit.

(c) G Erzeugermatrix von $\mathcal{C} \Rightarrow G$ Kontrollmatrix von $\mathcal{C}^\perp$, siehe Beweis a).
Umgekehrt: G Kontrollmatrix von $\mathcal{C}^\perp$, dann:
x Zeile von G, so $xy^t = 0$ für alle $y \in \mathcal{C}^\perp$, das heißt $x \in (\mathcal{C}^\perp)^\perp \overset{b)}{=} \mathcal{C}$.
Zeilen von G linear unabhängig, $\dim(\mathcal{C}) = n - (n-k) =$ Anzahl Zeilen von $G \Rightarrow G$ Erzeugermatrix von $\mathcal{C}$. Zweite Äquivalenz folgt aus b).

(d) Es gilt:

$$(-\underbrace{A^t}_{(n-k)\times k} \mid I_{n-k})(I_k \mid \underbrace{A}_{k \times n-k})^t = \underbrace{(-A^t \mid I_{n-k})}_{(n-k)\times k} \underbrace{\begin{pmatrix} I_k \\ \hline A^t \end{pmatrix}}_{n \times k} = -A^t I_k + I_{n-k} A^t = 0$$

Daraus folgt, dass $(-A^t \mid I_{n-k})$ eine Kontrollmatrix von $\mathcal{C}$ ist und nach (c) somit eine Erzeugermatrix für $\mathcal{C}^\perp$.

□

6.5 Beispiel.
Wir beschreiben die zu den Hamming-Codes dualen Codes:

Hamming-Code ist $[\underbrace{\frac{q^\ell - 1}{q-1}}_{n}, n - \ell]$-Code über K, $|K| = q$.

Kontrollmatrix H enthält aus jedem 1-dimensionalen Unterraum von K^l einen Vektor $\neq 0$.
Sei $\mathcal{C}$ der duale Code zum Hamming-Code. Nach 6.4c) ist H Erzeugermatrix von $\mathcal{C}$. $\mathcal{C}$ ist also ein $[\frac{q^\ell - 1}{q-1}, \ell]$-Code. Er heißt *Simplex-Code*.
Es gilt:

Jedes Codewort $\neq 0$ in $\mathcal{C}$ hat Gewicht $q^{\ell-1}$, d.h. der Abstand zwischen je zwei verschiedenen Codeworten ist immer konstant $q^{\ell-1}$;
insbesondere $\mathrm{d}(\mathcal{C}) = q^{\ell-1}$:
Seien $z_1, \ldots, z_\ell$ die Zeilen von H, $0 \neq x = (x_1, \ldots, x_n) \in \mathcal{C}$.
Da die Zeilen von H eine Basis von $\mathcal{C}$ bilden, gibt es $a_i \in K$ mit

$x = \sum_{i=1}^{\ell} a_i z_i = \sum_{i=1}^{\ell} a_i (z_{i1}, \ldots, z_{in})$. Jedes $\begin{pmatrix} z_{1j} \\ \vdots \\ z_{lj} \end{pmatrix}$ ist Spalte von H.

Setze $a = (a_1, \ldots, a_\ell) \in K^l$.

$U = \{b = \begin{pmatrix} b_1 \\ \vdots \\ b_\ell \end{pmatrix} : \ b_i \in K, \sum a_i b_i = 0\} = < a >^{\perp}$ hat Dimension $\ell - 1$ nach 6.4a).

U enthält also $\frac{q^{\ell-1}-1}{q-1}$ der Spalten h_j von H.
Es gilt: $x_j = 0 \Leftrightarrow \sum_{i=1}^{\ell} a_i z_{ij} = 0 \Leftrightarrow (z_{1j} \ldots z_{\ell j})^t \in U$.
Damit sind in x genau $\frac{q^{\ell-1}-1}{q-1}$ viele Komponenten $x_j = 0$. Und für das Gewicht gilt:

$$wt(x) = n - \frac{q^{\ell-1}-1}{q-1} = \frac{q^{\ell-1}(q-1)}{q-1} = q^{\ell-1}.$$

Der Simplex-Code ist nicht perfekt, von kleiner Dimension, hat aber großen Minimalabstand $d(\mathcal{C}) = q^{\ell-1}$.

Speziell: Dualer Code zum $[7,4]$-Hamming-Code über $\mathbb{Z}_2$:
Simplex-Code $\mathcal{C}$: $[7,3]$-Code mit Minimalabstand $d(\mathcal{C}) = 4$

$$\text{Erzeugermatrix} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} = \text{Kontrollmatrix des Hamming-Codes}$$

$$\begin{aligned} \mathcal{C} = \{&(0000000), (1001011), (0101101), (0010111), \\ &(1100110), (1011100), (0111010), (1110001)\} \end{aligned}$$

Wenn man die Gewichte sämtlicher Codewörter eines linearen Codes $\mathcal{C}$ kennt, so sind die Gewichte von $\mathcal{C}^{\perp}$ eindeutig bestimmt und im Prinzip berechenbar. Dazu benötigen wir den Begriff des Gewichtzählers:

6.6 Definition.
Sei $\mathcal{C}$ linearer $[n,k]$-Code über K, $|K|=q$.
Für $0 \le w \le n$ sei A_w die Anzahl der Codewörter von Gewicht w.
Dann heißt das Polynom

$$\sum_{w=0}^{n} A_w z^w \in \mathbb{R}[z]$$

der *Gewichtszähler* (weight enumerator) von $\mathcal{C}$.

Nun gilt folgender wichtiger Satz von MacWilliams:

6.7 Satz (MacWilliams-Identität)**.**
Sei $\mathcal{C}$ ein $[n,k]$-Code über K, $|K|=q$, mit Gewichtszähler $A(z)=\sum_{w=0}^{n} A_w z^w$.
Dann hat $\mathcal{C}^\perp$ den Gewichtszähler

$$B(z) = \frac{1}{q^k}(1+(q-1)z)^n A\left(\frac{1-z}{1+(q-1)z}\right).$$

Für den Beweis verweisen wir auf Lütkebohmert, Satz 1.4.5.
In binären Fall siehe auch Justesen, Høholdt, Theorem 1.4.1

6.8 Beispiel.
Wir berechnen den Gewichtszähler der binären Hamming-Codes. Sei $\mathcal{H}=\mathcal{C}^\perp$, wobei $\mathcal{C}$ der $[2^\ell-1,\ell]$-Simplex-Code ist (Beispiel 6.5 und 6.4b)).
Die Gewichtsverteilung von $\mathcal{C}$ ist nach 6.5 einfach:

$$A_0=1, \quad A_{2^{\ell-1}}=2^\ell-1, \quad A_i=0 \text{ für } i\ge 1, i\neq 2^{\ell-1} \text{ d.h. } A(z)=1+(2^\ell-1)z^{2^{\ell-1}}.$$

Nach 6.7 hat der binäre $[2^\ell-1, 2^\ell-1-\ell]$-Hamming-Code den Gewichtszähler

$$\begin{aligned} B(z) &= \frac{1}{2^\ell}(1+z)^{2^\ell-1} A\left(\frac{1-z}{1+z}\right) \\ &= \frac{1}{2^\ell}(1+z)^{2^\ell-1}\left(1+(2^\ell-1)\frac{(1-z)^{2^{\ell-1}}}{(1+z)^{2^{\ell-1}}}\right) \\ &= \frac{1}{2^\ell}((1+z)^{2^\ell-1}+(2^\ell-1)(1-z)^{2^{\ell-1}}(1+z)^{2^{\ell-1}-1}) \end{aligned}$$

Durch Ausmultiplizieren (binomische Formel) und Zusammenfassen der Potenzen von z erhält man dann den Gewichtszähler für $\mathcal{H}$. Wir führen dies für den $[7,4]$- und $[15,11]$-Hamming-Code durch (also für $l=3,4$).

$\ell = 3$:

$$\begin{aligned}
B(z) &= \frac{1}{8}((1+z)^7 + 7\cdot(1-z)^4(1+z)^3) \\
&= \frac{1}{8}((1+z)^7 + 7\cdot(1-z^2)^3(1-z)) \quad \text{(beachte: } (1-z)(1+z) = 1-z^2\text{)} \\
&= \frac{1}{8}((1+7z+\binom{7}{2}z^2+\binom{7}{3}z^3+\binom{7}{4}z^4+\binom{7}{5}z^5+7z^6+z^7) \\
&\quad +7\cdot(1-3z^2+3z^4-z^6)(1-z)) \\
&= \frac{1}{8}(1+7z+21z^2+35z^3+35z^4+21z^5+7z^6+z^7 \\
&\quad +7-21z^2+21z^4-7z^6-7z+21z^3-21z^5+7z^7) \\
&= 1+7z^3+7z^4+z^7
\end{aligned}$$

Der $[7,4]$-Hamming-Code über $\mathbb{Z}_2$ enthält also außer 0 ein Codewort von Gewicht 7 (1111111), 7 Codewörter von Gewicht 3 und 7 Codewörter von Gewicht 4.
(Das hätte man natürlich auch direkt nachprüfen können.)

Für $\ell = 4$ müsste man dafür schon $2^{11} = 2048$ Codewörter durchprüfen. Hier ist die MacWilliams-Identität der bessere Weg, obwohl die Rechnungen auch etwas mühsam sind (Computer-Algebra-Systeme führen solche Auswertungen automatisch durch).

$[15,11]$-Hamming-Code: $B(z) = \frac{1}{16}((1+z)^{15} + 15\cdot(1-z)^8(1+z)^7)$
Ausrechnen liefert:

$$\begin{aligned}
B(z) &= 1+35z^3+105z^4+168z^5+280z^6+435z^7+435z^8+280z^9 \\
&\quad +168z^{10}+105z^{11}+35z^{12}+z^{15}
\end{aligned}$$

Weitere Konstruktionen:

6.9 Definition.
$\mathcal{C}$ ein $[n,k]$-Code über K, $|K| = q$. Wir nennen

$$\hat{\mathcal{C}} = \{(c_1,\ldots,c_{n+1}) : c_i \in K, (c_1,\ldots,c_n) \in \mathcal{C}, \sum_{i=1}^{n+1} c_i = 0\}$$

Erweiterung von $\mathcal{C}$ (durch parity-check-symbol).

6.10 Satz.
Sei $\mathcal{C}$ ein $[n,k]$-Code über K.

(a) $\widehat{\mathcal{C}}$ *ist ein linearer* $[n+1,k]$*-Code über* K *mit*

$$d(\mathcal{C}) \leq d(\widehat{\mathcal{C}}) \leq d(\mathcal{C}) + 1$$

(b) Ist $\mathcal{C}$ *binär* ($K = \mathbb{Z}_2$)*, dann gilt*

$$d(\widehat{\mathcal{C}}) = d(\mathcal{C}) + 1 \Leftrightarrow d(\mathcal{C}) \text{ ungerade.}$$

(c) Ist H *Kontrollmatrix von* $\mathcal{C}$ *, so ist*

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

eine Kontrollmatrix von $\widehat{\mathcal{C}}$*.*

Beweis.

(a) Klar, dass $\widehat{\mathcal{C}}$ ein linearer Code der Länge $n+1$ ist.
Da $|\widehat{\mathcal{C}}| = |\mathcal{C}| = q^k$, folgt $\dim(\widehat{\mathcal{C}}) = k$.
Die Aussage über die Minimaldistanz folgt aus

$$wt(x) = wt(x_1, \dots, x_n) + wt(x_{n+1})$$

für $x = (x_1, \dots, x_n, x_{n+1}) \in \widehat{\mathcal{C}}$.

(b) $\Leftarrow$: Sei $0 \neq \hat{x} = (c_1, \dots, c_{n+1}) \in \widehat{\mathcal{C}}$, so dass $wt(\hat{x}) = d(\widehat{\mathcal{C}})$.
Setze $x = (c_1, \dots, c_n)$:

1. $c_{n+1} = 0 \Rightarrow wt(\hat{x}) = wt(x)$ ist gerade.
   Da $d(\mathcal{C})$ ungerade ist, existiert ein $0 \neq y = (d_1, \dots, d_n) \in \mathcal{C}$ mit $d(\mathcal{C}) = wt(y) < wt(x)$.
   Dann ist $\hat{y} = (d_1, \dots, d_n, 1) \in \widehat{\mathcal{C}}$ mit $wt(\hat{y}) = wt(y) + 1 \leq wt(x) = wt(\hat{x}) = d(\widehat{\mathcal{C}})$.
   Also $d(\widehat{\mathcal{C}}) = wt(\hat{y}) = wt(y) + 1 = d(\mathcal{C}) + 1$.
2. $c_{n+1} \neq 0 \Rightarrow wt(x) < wt(\hat{x}) = d(\widehat{\mathcal{C}})$.
   Damit ist nach (a) $d(\mathcal{C}) = d(\widehat{\mathcal{C}}) + 1$.

$\Rightarrow$: Sei $x \in \mathcal{C}$ mit $wt(x) = d(\mathcal{C})$. Angenommen $d(\mathcal{C})$ wäre gerade.
Dann wäre $\hat{x} = (c_1, \dots, c_n, 0) \in \widehat{\mathcal{C}}$.
Daraus folgt: $d(\widehat{\mathcal{C}}) \leq wt(\hat{x}) = wt(x) = d(\mathcal{C})$ und mit (a) $d(\widehat{\mathcal{C}}) = d(\mathcal{C})$.
Widerspruch.

(c) $(c_1, \ldots, c_n, c_{n+1}) \in \widehat{\mathcal{C}} \Leftrightarrow H(c_1, \ldots, c_n)^t = 0$ und $\sum_{i=1}^{n+1} c_i = 0$

$$\Leftrightarrow \begin{pmatrix} 1 & \ldots & 1 \\ & H & 0 \\ & & 0 \end{pmatrix} \begin{pmatrix} c_1 \\ \vdots \\ c_{n+1} \end{pmatrix} = 0.$$

□

6.11 Beispiel.
Sei $\mathcal{C}$ der $[7,4]$-Hamming-Code über $\mathbb{Z}_2$ (wie in 4.5); dann ist $d(\mathcal{C}) = 3$.
$\widehat{\mathcal{C}}$ ist ein binärer $[8,4]$-Code mit $d(\widehat{\mathcal{C}}) = 4$ nach 6.10.b). Da $\hat{C}$ Länge 8 (1 Byte) hat, wird er häufig anstelle von $\mathcal{C}$ eingesetzt.

Erzeugermatrix von $\hat{\mathcal{C}}$: $\hat{G} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$ (vergleiche 4.5)

6.12 Definition.
Sei $\mathcal{C}$ ein $[n,k]$-Code über K, $|K| = q$, $n \geq 2$ und $1 \leq i \leq n$.

(a) $\check{\mathcal{C}} = \check{\mathcal{C}}_i = \{(c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_n) : \ (c_1, \ldots, c_{i-1}, 0, c_{i+1}, \ldots, c_n) \in \mathcal{C}\}$
heißt die *Verkürzung von $\mathcal{C}$ an der Stelle i.*

(b) $\overset{\circ}{\mathcal{C}} = \overset{\circ}{\mathcal{C}}_i = \left\{ (c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_n) : \begin{array}{c} \exists c_i \in K \text{ mit} \\ (c_1, \ldots, c_i, \ldots, c_n) \in \mathcal{C} \end{array} \right\}$
heißt die *Punktierung von $\mathcal{C}$ an der Stelle i.*

6.13 Satz.
Sei $\mathcal{C}$ ein $[n,k]$-Code, $n \geq 2$, $k \geq 1$.

(a) $\check{\mathcal{C}}_i$ *und* $\overset{\circ}{\mathcal{C}}_i$ *sind lineare Codes der Länge* $n-1$,
$\check{\mathcal{C}}_i \subseteq \overset{\circ}{\mathcal{C}}_i$,
$k - 1 \leq \dim(\check{\mathcal{C}}_i) \leq \dim(\overset{\circ}{\mathcal{C}}_i) \leq k$,
$d(\check{\mathcal{C}}_i) \geq d(\overset{\circ}{\mathcal{C}}_i)$ *(falls* $\check{\mathcal{C}}_i \neq \{0\}$*).*

(b) $\dim(\check{\mathcal{C}}_i) = k - 1 \Leftrightarrow \exists (c_1, \ldots, c_i, \ldots, c_n) \in \mathcal{C}$ *mit* $c_i \neq 0$.

(c) $\dim(\overset{\circ}{\mathcal{C}}_i) = k - 1 \Leftrightarrow \exists (0, \ldots, 0, c_i, 0, \ldots, 0) \in \mathcal{C}$ *mit* $c_i \neq 0$.
(Insbesondere: Ist $d(\mathcal{C}) \geq 2$*, so ist* $\dim(\overset{\circ}{\mathcal{C}}_i) = \dim(\mathcal{C}) = k$*)*

(d) $\check{\mathcal{C}}_i = \overset{\circ}{\mathcal{C}}_i \Leftrightarrow$ *(i)* $\exists (0, \ldots, 0, c_i, 0, \ldots, 0) \in \mathcal{C}$ *mit* $c_i \neq 0$ *oder*
(ii) $\forall (c_1, \ldots, c_n) \in \mathcal{C} : \ c_i = 0$.

(e) *Ist* $\check{\mathcal{C}}_i \neq \{0\}$, *so ist* $d(\check{\mathcal{C}}_i) \geq d(\mathcal{C})$.
(Ist $k > 1$, *so ist* $\check{\mathcal{C}}_i \neq \{0\}$*)*

(f) *Ist* $\dim(\overset{\circ}{\mathcal{C}}_i) = k$ *(d.h. nach (c)* $\mathcal{C} \cap \{0, \ldots, 0, c, 0, \ldots, 0) : c \in K\}) = \{0\}$*),*
so ist $d(\mathcal{C}) - 1 \leq d(\overset{\circ}{\mathcal{C}}_i) \leq d(\mathcal{C})$.

Beweis.

(a) Klar bis auf die Dimensionsaussage:
Klar: $\dim(\check{C}_i) \leq \dim(\overset{\circ}{\mathcal{C}}_i) \leq k$.
(i) Ist $c_i = 0$ für alle $(c_1, \ldots, c_n) \in \mathcal{C}$, so ist $\dim(\check{\mathcal{C}}_i) = \dim(\overset{\circ}{\mathcal{C}}_i) = k$.
(ii) Sei $x_1 = (c_1, \ldots, c_n) \in \mathcal{C}$ mit $c_i \neq 0$.
Dann existiert eine Basis $x_1, \ldots, x_k$ von $\mathcal{C}$.
Es gibt $\alpha_2, \ldots, \alpha_k \in K$, so dass $x_2 - \alpha_2 x_1, \ldots, x_k - \alpha_k x_1$ an der i-ten Stelle eine Null haben. Klar: $x_2 - \alpha_2 x_1, \ldots, x_k - \alpha_k x_1$ sind linear unabhängig. Also $\dim(\check{\mathcal{C}}_i) \geq k - 1$.

(b) $\dim(\check{\mathcal{C}}_i) = k - 1 \overset{(a)}{\Leftrightarrow} |\check{\mathcal{C}}_i| < |\mathcal{C}| \Leftrightarrow \exists (c_1, \ldots, c_n) \in \mathcal{C}$ mit $c_i \neq 0$.

(c) $\overset{\circ}{\mathcal{C}}_i$ ist Bild der Projektion π von $\mathcal{C}$ auf die Komponenten $\neq i$.
$(\pi(c_1, \ldots, c_n) = (c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_n))$
Kern $\pi = \mathcal{C} \cap \{(0, \ldots, 0, c, 0, \ldots, 0) : c \in K\}$.
Daraus folgt c)

(d) $\check{\mathcal{C}}_i = \overset{\circ}{\mathcal{C}}_i \Leftrightarrow \dim(\check{\mathcal{C}}_i) = \dim(\overset{\circ}{\mathcal{C}}_i) \Leftrightarrow \dim(\overset{\circ}{\mathcal{C}}_i) = k - 1$ oder $\dim(\check{\mathcal{C}}_i) = k$.
Behauptung folgt aus c).

(e) Sei $0 \neq x = (c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_n) \in \check{\mathcal{C}}_i,\ wt(x) = d(\check{\mathcal{C}}_i)$.
Dann ist $\hat{x} = (c_1, \ldots, c_{i-1}, 0, c_{i+1}, \ldots, c_n) \in \mathcal{C}$,
$d(\mathcal{C}) \leq wt(\hat{x}) = wt(x) = d(\check{\mathcal{C}}_i)$.

(f) Sei $0 \neq y = (c_1, \ldots, c_n) \in \mathcal{C},\ wt(y) = d(\mathcal{C})$ (Beachte: $\mathcal{C} \neq \{0\}$, da $k \geq 1$).
Dann ist $\overset{\circ}{y} = (c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_n) \in \overset{\circ}{\mathcal{C}}_i$ und $\overset{\circ}{y} \neq 0$, da sonst
$\mathcal{C} \cap \{(0, \ldots, 0, c, 0, \ldots, 0) :\ c \in K\} \neq \{0\}$. Also:

$$d(\overset{\circ}{\mathcal{C}}_i) \leq wt(\overset{\circ}{y}) \leq wt(y) = d(\mathcal{C}).$$

Sei $0 \neq x = (d_1, \ldots, d_{i-1}, d_{i+1}, \ldots, d_n) \in \overset{\circ}{\mathcal{C}},\ wt(x) = d(\overset{\circ}{\mathcal{C}}_i)$.
Dann existiert $\hat{x} = (d_1, \ldots, d_{i-1}, d_i, d_{i+1}, \ldots, d_n) \in \mathcal{C}$ und $\hat{x} \neq 0$. Somit:

$$d(\mathcal{C}) \leq wt(\hat{x}) \leq wt(x) + 1 = d(\overset{\circ}{\mathcal{C}}) + 1.$$

□

6.14 Beispiele.

(a) Sei $\mathcal{C} = \{(\underbrace{0,\ldots,0}_{n}), (1,\ldots,1), (0,1,\ldots,1), (1,0,\ldots,0)\}$, $n \geq 2$.

$\mathcal{C}$ ist linearer $[n,2]$-Code über $\mathbb{Z}_2$ mit $d(\mathcal{C}) = 1$.

$$\check{\mathcal{C}}_1 = \{(\underbrace{1,\ldots,1}_{n-1}), (\underbrace{(0,\ldots,0)}_{n-1})\}$$

und $d(\check{\mathcal{C}}_1) = n-1$.
(In 6.13.e) gibt es also keine Konstante c mit $d(\check{\mathcal{C}}_i) \leq d(\mathcal{C}) + c$).

(b)

$$\begin{aligned}
\mathcal{C} &= \{(1101), (0110), (1011), (0000)\},\\
&\mathcal{C} \text{ ist linearer } [4,2]\text{-Code über } \mathbb{Z}_2,\ d(\mathcal{C}) = 2\\
\overset{\circ}{\mathcal{C}}_1 &= \{(101), (110), (011), (000)\},\ d(\overset{\circ}{\mathcal{C}}_1) = 2\\
\overset{\circ}{\mathcal{C}}_2 &= \{(101), (010), (111), (000)\},\ d(\overset{\circ}{\mathcal{C}}_2) = 1
\end{aligned}$$

Dies zeigt, dass beide Fälle in 6.13.f) auftreten können.

Eine letzte Konstruktionsmöglichkeit:

6.15 Satz (Plotkin-Konstruktion)**.**
Für $i = 1, 2$ seien $\mathcal{C}_i$ $[n, k_i]$-Codes mit $d(\mathcal{C}_i) = d_i$ über K, $|K| = q$. Dann ist

$$\mathcal{C}_1 \propto \mathcal{C}_2 := \{(x_1, x_1 + x_2) :\ x_i \in \mathcal{C}_i\} \subseteq K^{2n}$$

ein $[2n, k_1 + k_2]$-Code über K mit $d(\mathcal{C}) = min\{2d_1, d_2\}$.

Beweis.
i) Klar, dass $\mathcal{C} \subseteq K^{2n}$. D.h. $\mathcal{C}$ ist von der Länge $2n$.
ii) Betrachte die Abbildung:

$$\alpha : \begin{cases} \mathcal{C}_1 \oplus \mathcal{C}_2 \mapsto \mathcal{C} \\ (x_1, x_2) \mapsto (x_1, x_1 + x_2) \end{cases}$$

Die Abbildung α ist linear, injektiv und surjektiv, also ein Vektorraumisomorphismus. Damit gilt für die Dimension:

$$\dim(\mathcal{C}) = \dim(\mathcal{C}_1 \oplus \mathcal{C}_2) = k_1 + k_2.$$

iii) Was ist d($\mathcal{C}$)? Ist $\mathcal{C} = \{0\}$, so ist $d_1 = d_2 = 0$.
Sei also $\mathcal{C} \neq \{0\}$ und $0 \neq x = (x_1, x_1 + x_2) \in \mathcal{C}$. Definiere für $z = (z_1, \ldots, z_n) \in K^n$

$$Tr(z) := \{j :\ z_j \neq 0\}$$

als den *Träger* von z (also $wt(z) = |Tr(z)|$).

$$\begin{aligned} wt(x) &= wt(x_1) + wt(x_1 + x_2) \\ &\geq wt(x_1) + wt(x_1) + wt(x_2) - 2|Tr(x_1) \cap Tr(x_2)| \\ &\geq wt(x_2) = d_2 \\ &(\text{da } wt(x_1) \geq |Tr(x_1) \cap Tr(x_2)| \ ) \end{aligned}$$

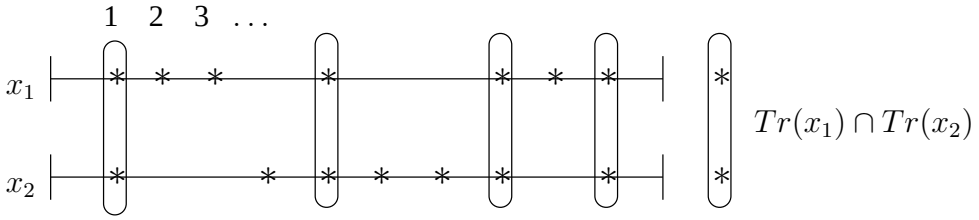


Abbildung 8: Der Träger von x_1 und x_2.

Also: Ist $x_2 \neq 0$, so $d(\mathcal{C}) = wt(x) \geq wt(x_2) \geq d_2$.

Ist $x_2 = 0$, so $d(\mathcal{C}) = wt(x) = 2wt(x_1) \overset{x_1 \neq 0}{\geq} 2d_1$.
Also: $d(\mathcal{C}) \geq \min\{2d_1, d_2\}$.
Mit $x_1 = 0$ und x_2 von minimalem Gewicht in $\mathcal{C}_2$ oder $x_2 = 0$ und x_1 von minimalem Gewicht in $\mathcal{C}_1$ wird das Minimum auch angenommen. □

6.16 Beispiel.
$\mathcal{C}_1$= binärer Hamming $[7, 4]$-Code, $d(\mathcal{C}_1)$=3.
$\mathcal{C}_2$= binärer $[7, 1]$-Code, $d(\mathcal{C}_2) = 7$.

$\mathcal{C} = \mathcal{C}_1 \propto \mathcal{C}_2 = \{(x_1, x_1), \underbrace{(x_1, \bar{x}_1)}_{\text{Gewicht 7}} : x_1 \in \mathcal{C}_1\}$, $d(\mathcal{C}) = \min\{7, 6\} = 6$.

$C' = \mathcal{C}_2 \propto \mathcal{C}_1 = \{(0, \ldots, 0, x_1), (1, \ldots, 1, \bar{x}_1) : x_1 \in \mathcal{C}_1\}$, $d(\mathcal{C}') = 3$.

6.17 Bemerkung.
$\mathcal{C}_1 \oplus \mathcal{C}_2$ ist direkte Summe von Codes, ergibt
$[n_1 + n_2, k_1 + k_2, \min(d_1, d_2)]$-Code.

7 Reed-Muller-Codes und Majority-Logic-Decodierung

Binäre Reed-Muller-Codes (1954, Muller; Majority-Logic-Decodierung für RM-Codes durch Reed, 1954)
$K = \mathbb{Z}_2$ (RM-Codes lassen sich auch auf andere endliche Körper verallgemeinern.)

7.1 Definition.
$0 \leq r \leq m$, $r, m \in \mathbb{N}_0$.
Definiere:

$$\begin{aligned} RM(0,m) &= [2^m, 1] - \text{Wiederholungscode} \\ &= \{\underbrace{(0,\ldots,0)}_{2^m}, \underbrace{(1,\ldots,1)}_{2^m}\} \\ RM(m,m) &= (\mathbb{Z}_2)^{2^m} \end{aligned}$$

(Also insb.: $RM(0,0) = \mathbb{Z}_2^1$, $RM(0,1) = \{(0,0),(1,1)\}$, $RM(1,1) = \mathbb{Z}_2^2$)
$m = 2, 3, \ldots$, $1 \leq r \leq m-1$:

$$RM(r,m) = RM(r, m-1) \propto RM(r-1, m-1)$$

(rekursive Definition) *Binärer Reed-Muller-Code r-ter Ordnung.*

7.2 Satz.
$0 \leq r \leq m$, $r, m \in \mathbb{N}_0$.

(a) $RM(r,m)$ *ist binärer* $\left[2^m, \sum_{i=0}^{r} \binom{m}{i}\right]$*-Code*

(b) $d(RM(r,m)) = 2^{m-r}$.

Beweis:

(a) Länge von $RM(r,m) = 2^m$, falls $r = 0$ oder $r = m$.
Ansonsten per Induktion (nach $r+m$):
Länge von $RM(r,m) = 2^{m-1} + 2^{m-1} = 2^m$.

Dimension:

$$\dim(RM(0,m)) = 1 = \sum_{i=0}^{0} \binom{m}{i}$$

$$\dim(RM(m,m)) = 2^m = \sum_{i=0}^{m} \binom{m}{i}$$

Ansonsten per Induktion (nach $r+m$):

$$\begin{aligned}\dim(RM(r,m)) &\overset{6.15}{=} \dim(RM(r,m-1)) + \dim(RM(r-1,m-1)) \\ &= \sum_{i=0}^{r}\binom{m-1}{i} + \sum_{i=0}^{r-1}\binom{m-1}{i} \\ &= 1 + \sum_{i=0}^{r-1}\left[\binom{m-1}{i+1} + \binom{m-1}{i}\right] \\ &= 1 + \sum_{i=0}^{r-1}\binom{m}{i+1} = \sum_{i=0}^{r}\binom{m}{i}.\end{aligned}$$

(b) $d(RM(0,m)) = 2^m = 2^{m-0}$
$d(RM(m,m)) = 1 = 2^{m-m}$
Ansonsten per Induktion:
$d(RM(r,m)) \overset{6.15}{=} \min\{2 \cdot 2^{m-1-r}, 2^{(m-1)-(r-1)}\} = 2^{m-r}$.

□

7.3 Beispiele.

(a) $\mathcal{C} = RM(1,2)$ ist ein $[4,3]$-Code über $\mathbb{Z}_2$ mit $d(\mathcal{C}) = 2$:

$$\begin{aligned}\mathcal{C} &= RM(1,1) \propto RM(0,1) \\ &= \mathbb{Z}_2^2 \propto \{(00),(11)\} \\ &= \{(0000),(0011),(1111),(1100), \\ &\quad (0101),(0110),(1010),(1001)\}\end{aligned}$$

(b) $\mathcal{C} = RM(1,3)$ ist ein $[8,4]$-Code über $\mathbb{Z}_2$ mit $d(\mathcal{C}) = 4$, $|\mathcal{C}| = 16$.
$\mathcal{C} = RM(1,2) \propto RM(0,2) = RM(1,2) \propto \{(0,0,0,0),(1,1,1,1)\}$.

(c) $\mathcal{C} = RM(1,4)$ ist $[16,5]$-Code über $\mathbb{Z}_2$ mit $d(\mathcal{C}) = 8$.

(d) $\mathcal{C} = RM(1,5)$ ist ein $[32,6]$-Code über $\mathbb{Z}_2$ mit $d(\mathcal{C}) = 16$.
$RM(1,5)$ wurde bei den Mariner Expeditionen 6, 7 und 9 zum Mars in den Jahren 1969 - 1972 eingesetzt, um Bilder zur Erde zu übertragen. Die $2^6 = 64$ Codewörter entsprachen der Helligkeit (Grautöne) eines Bildpunktes. Wegen $7 \leq \frac{d(\mathcal{C})-1}{2} = \frac{15}{2}$ konnten bis zu 7 Fehler in einem Codewort (der Länge 32) korrigiert werden. Ein Bild wurde in ein 600×600 Gitter zerlegt und jeder der 360.000 Bildpunkte in ein Codewort codiert. [1]

[1] vgl. www.jpl.nasa.gov bzw. www.nasa.gov

7.4 Satz.

(a) $RM(r-1,m) \subseteq RM(r,m)$

(b) $(\underbrace{1,\ldots,1}_{2^m}) \in RM(r,m)$ *für alle* $0 \le r \le m$

(c) $x \in RM(1,m),\ x \neq (0,\ldots,0),(1,\ldots,1)$ *, so* $wt(x) = 2^{m-1}$
(vgl. Simplex-Codes)

(d) $RM(m-1,m) = \{x \in \mathbb{Z}_2^{2^m} : wt(x)\ gerade\}$.

Beweis.

(a) Induktion nach m:
$m = 1:\ RM(0,1) \subseteq RM(1,1)$ klar.
$m-1 \to m$:
$r = m:\ RM(m-1,m) \subseteq RM(m,m) = \mathbb{Z}_2^{2^m}$
$r < m$:

$$\begin{aligned} RM(r-1,m) &= RM(r-1,m-1) \propto RM(r-2,m-1) \\ &\overset{Ind.}{\subseteq} RM(r,m-1) \propto RM(r-1,m-1) \\ &= RM(r,m). \end{aligned}$$

(b) Folgt aus (a), da $RM(0,m) \subseteq RM(r,m)$.

(c) Induktion nach m.
$m = 1:\ RM(1,1) = \{(00),(01),(10),(11)\}$ und $wt(01) = wt(10) = 1 = 2^{1-1}$

$$\begin{aligned} m > 1:\ RM(1,m) &= RM(1,m-1) \propto RM(0,m-1) \\ &= \{(x,x),(x,\bar{x}) : x \in RM(1,m-1)\} \end{aligned}$$

(i) $x = (0,\ldots,0)$ oder $(1,\ldots,1)$: $wt(x,\overline{x}) = 2^{m-1}$.

(ii) $x \neq (0,\ldots,0),(1,\ldots,1)$: $\overset{Ind.}{\Rightarrow} wt(x) = wt(\overline{x}) = 2^{m-2}$
$\Rightarrow wt(x,x) = wt(x,\overline{x}) = 2^{m-1}$.

(d) Induktion nach m.
$m = 1:\ RM(0,1) = \{(00),(11)\}$ klar.
$m-1 \to m$:

$$\begin{aligned} RM(m-1,m) &= RM(m-1,m-1) \propto RM(m-2,m-1) \\ &\overset{Ind.}{=} \{(x,x+y) :\ x \in \mathbb{Z}_2^{2^{m-1}},\ y \in \mathbb{Z}_2^{2^{m-1}},\ wt(y)\ \text{gerade}\} \end{aligned}$$

Sei $c = |Tr(x) \cap Tr(y)|$. Dann ist

$$wt(x+y) = wt(x) + wt(y) - 2c.$$

Daher ist $wt(x, x+y) = wt(x)+wt(x+y) = 2wt(x)+wt(y)-2c$ gerade.
Ferner ist

$$\dim(RM(m-1,m)) = \sum_{i=0}^{m-1} \binom{m}{i} = 2^m - 1.$$

Genau $2^{2^m-1} = \frac{1}{2}2^{2^m}$ viele Elemente des $\mathbb{Z}_2^{2^m}$ haben gerades Gewicht.
Daraus folgt die Behauptung.

□

7.5 Beispiele.
$\mathcal{C} = RM(1,m)$ ist $[2^m, m+1]$- Code über $\mathbb{Z}_2$ mit $d(\mathcal{C}) = 2^{m-1}$.

(a) $\check{\mathcal{C}}_i$ ist $[2^m-1, m]$-Code über $\mathbb{Z}_2$ mit $d(\check{\mathcal{C}}_i) = 2^{m-1}$; jedes Codewort $\neq 0$ hat Gewicht 2^{m-1}:

$m = 1:$ $\mathcal{C} = RM(1,1) = \{(0,0),(0,1),(1,0),(1,1)\}$
$\check{\mathcal{C}}_i = \{(0),(1)\}$, $i = 1,2$.

$m > 1:$ $RM(1,m) = RM(1,m-1) \propto RM(0,m-1)$
$(1,\ldots,1) \overset{7.4.b)}{\in} RM(1,m-1)$
$\Rightarrow (1,\ldots,1,0,\ldots,0),(0,\ldots,0,1,\ldots,1) \in RM(1,m)$
$\overset{6.13.b)}{\Rightarrow} \dim(\check{\mathcal{C}}_i) = m$ $(= \dim(\mathcal{C}) - 1)$.

Jedes Codewort $\neq (0,\ldots,0),(1,\ldots,1)$ in $\mathcal{C}$ hat nach 7.4c) Gewicht 2^{m-1}. Also liefert die Verkürzung außer $(0,\ldots,0)$ nur Codeworte von Gewicht 2^{m-1}. Damit ist $d(\check{\mathcal{C}}_i) = 2^{m-1}$. $\check{\mathcal{C}}_i$ hat die gleichen Parameter wie der binäre Simplex-Code der Länge 2^{m-1} (dualer Code zum Hamming $[2^m-1, 2^m-1-m]$-Code);vergleiche 6.5.
Man kann zeigen, dass $\check{\mathcal{C}}_i$ der Simplex-Code ist.

(b) $\overset{\circ}{\mathcal{C}}_i$ ist $[2^m-1, m+1]$-Code mit $d(\overset{\circ}{\mathcal{C}}_i) = 2^{m-1}-1$ (für $m \geq 2$):
$\mathcal{C}$ enthält keine Codewörter vom Gewicht 1 (da $m \geq 2$).
Nach 6.13.c): $\dim(\overset{\circ}{\mathcal{C}}_i) = \dim(\mathcal{C}) = m+1$.
Die Punktierung von $(1,\ldots,1,0,\ldots,0)$ bzw. $(0,\ldots,0,1,\ldots,1)$ liefert ein Codewort in $\overset{\circ}{\mathcal{C}}_i$ vom Gewicht $2^{m-1}-1$.
Also nach 6.13.f): $d(\overset{\circ}{\mathcal{C}}_i) = 2^{m-1}-1$.

Wir geben jetzt noch zwei weitere Beschreibungsmöglichkeiten für die Reed-Muller-Codes an, die wir auch für die Darstellung der Majority-Logic-Decodierung bei diesen Codes benötigen.

7.6. Boole'sche Funktionen
Betrachte Funktion $f : \mathbb{Z}_2^m \to \mathbb{Z}_2$. Anzahl $2^{|Z_2^m|} = 2^{2^m}$.

Beschreibung jeder solchen Funktion $f(x_1, \ldots, x_m)$ durch Wertetabelle:

x_m	0	0	0	0	$\ldots$	1	1	1	1
x_{m-1}	0	0	0	0		1	1	1	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$	$\vdots$
x_3	0	0	0	0		1	1	1	1
x_2	0	0	1	1		0	0	1	1
x_1	0	1	0	1	$\ldots$	0	1	0	1
$f(x_1, \ldots, x_m)$	$*$	$*$	$*$	$*$	$\ldots$	$*$	$*$	$*$	$*$

Werte von f, d.h. $* \in \mathbb{Z}_2$

Diese Anordnung der Vektoren in $\mathbb{Z}_2^m$ behalten wir im folgenden bei.

Diese Funktionen bilden einen Ring mit punktweiser Addition und Multiplikation.
Jede solche Funktion ist eine Polynomfunktion:
Eine *Polynomfunktion* ist z.B. von der Form $f(x_1, x_2, x_3) = 1 + x_1 + x_1x_2 + x_2 + x_3$ (Addition und Multiplikation über $\mathbb{Z}_2$).
Beachte: $x_i^2 = x_i$, $x_ix_j = x_jx_i$ (als Polynomfunktionen).
Jede Polynomfunktion ist Linearkombination der *Monome*

$$1,\ x_1,\ x_2,\ \ldots,\ x_m,\ x_1x_2,\ x_1x_3,\ \ldots,\ x_1x_m,$$
$$x_2x_3,\ \ldots,\ x_{m-1}x_m, x_1x_2x_3,\ \ldots\ ,\ x_1x_2\ldots x_m$$

Davon gibt es 2^m viele (=Anzahl der Teilmengen von $\{x_1, \ldots, x_m\}$) und sie sind über $\mathbb{Z}_2$ linear unabhängig.
Also gibt es 2^{2^m} viele Polynomfunktionen, und dies sind aus Anzahlgründen alle Funktionen $\mathbb{Z}_2^m \to \mathbb{Z}_2$.
($+$ entspricht XOR; $\cdot$ entspricht $\wedge$; daher auch Bezeichnung *Boole'sche Funktionen*).
Der *Grad* eines Monoms $x_{i_1} \ldots x_{i_k}$, $i_1 < \ldots < i_k$, ist k.
Grad einer Polynomfunktion f ist Maximum der Grade der in f auftretenden Monome.

Allgemeine Berechnungsmöglichkeit der Darstellung einer Boole'schen Funktion als Polynomfunktion:

$$f = \bigvee_{\{a \in \mathbb{Z}_2^m:\ f(a)=1\}} \Big(\bigwedge_{\{j:\ a_j=1\}} x_j \wedge \bigwedge_{\{k:\ a_k=0\}} \bar{x}_k \Big)$$

$$
\begin{aligned}
\cdot &= \wedge \\
1 + g &= \bar{g} \\
f + g + fg &= f \vee g \\
g + g &= 0 \\
g \cdot g &= g \\
g \cdot \bar{g} &= 0 \\
g + \bar{g} &= 1
\end{aligned}
$$

Beispiel:

x_3	0	0	0	0	1	1	1	1
x_2	0	0	1	1	0	0	1	1
x_1	0	1	0	1	0	1	0	1
$f(x_1, x_2, x_3)$	0	0	0	1	1	0	0	0
				$\uparrow$	$\uparrow$			

$$
\begin{aligned}
f &= x_1 x_2 \overline{x}_3 \vee \bar{x}_1 \bar{x}_2 x_3 \\
&= x_1 x_2 \overline{x}_3 + \bar{x}_1 \bar{x}_2 x_3 + \underbrace{x_1 x_2 \overline{x}_3 \bar{x}_1 \bar{x}_2 x_3}_{=0} \\
&= x_1 x_2 + x_1 x_2 x_3 + (1 + x_1)(1 + x_2) x_3 \\
&= x_1 x_2 + x_1 x_2 x_3 + (1 + x_1 + x_2 + x_1 x_2) x_3 \\
&= x_1 x_2 + x_1 x_2 x_3 + x_3 + x_1 x_3 + x_2 x_3 + x_1 x_2 x_3 \\
&= x_3 + x_1 x_2 + x_1 x_3 + x_2 x_3
\end{aligned}
$$

Wir ordnen jeder Funktion $f : \mathbb{Z}_2^m \to \mathbb{Z}_2$ (also jedem Boole'schen Polynom) den Vektor $\underline{f} \in \mathbb{Z}_2^{2^m}$ zu, der die unterste Zeile in der Wertetabelle von f ist (also die Folge der Funktionswerte).

Beispiel oben: $m = 3$

$$
\begin{aligned}
\underline{f} &= (00011000) \text{ (aus obigem Beispiel)} \\
\underline{x_1} &= (01010101) \\
\underline{x_3} &= (00001111) \\
\underline{x_1}\,\underline{x_2} &= (00010001) \\
\underline{x_1}\,\underline{x_2} + \underline{x_3} &= (00011110)
\end{aligned}
$$

7.7 Definition.
Für $0 \leq r \leq m$ definiere Code $\widetilde{R}(r, m)$ der Länge 2^m als Menge aller Vektoren $\underline{f}$ in $\mathbb{Z}_2^{2^m}$, wobei f Polynomfunktion vom Grad $\leq r$ auf $\mathbb{Z}_2^m$ ist.

7.8 Beispiel.

(a) $\widetilde{R}(0,m) = \{(0,\ldots,0),(1,\ldots,1)\}$

(b) $\widetilde{R}(1,2)$

$$\begin{aligned} \underline{0} &= (0,0,0,0) & \underline{1} &= (1,1,1,1) \\ \underline{x_1} &= (0,1,0,1) & \underline{x_1}+\underline{1} &= (1,0,1,0) \\ \underline{x_2} &= (0,0,1,1) & \underline{x_2}+\underline{1} &= (1,1,0,0) \\ \underline{x_1}+\underline{x_2} &= (0,1,1,0) & \underline{x_1}+\underline{x_2}+\underline{1} &= (1,0,0,1) \end{aligned}$$

$$\left(\begin{array}{c|cccc} x_2 & 0 & 0 & 1 & 1 \\ x_1 & 0 & 1 & 0 & 1 \\ \hline f(x_1,x_2) & * & * & * & * \end{array}\right)$$

7.3a): $\widetilde{R}(1,2) = RM(1,2)$.

Dies gilt allgemein:

7.9 Satz.
$\widetilde{R}(r,m) = RM(r,m)$.

Beweis.
Es ist $\widetilde{R}(0,m) = RM(0,m)$ und $\widetilde{R}(m,m) = \mathbb{Z}_2^{2^m} = RM(m,m)$.
Sei nun $1 \le r \le m$, $\underline{f} \in \widetilde{R}(r,m)$, $f = f(x_1,\ldots,x_m)$ Polynomfunktion vom Grad $\le r$.
Es ist

$$\begin{aligned} f(x_1,\ldots,x_m) &= g(x_1,\ldots,x_{m-1}) + x_m h(x_1,\ldots,x_{m-1}) \\ &\qquad \text{grad}(g) \le r,\ \text{grad}(h) \le r-1\ . \end{aligned}$$

Seien $\underline{g}$ und $\underline{h}$ die Vektoren der Länge 2^{m-1} in $\widetilde{R}(r,m-1)$ bzw. $\widetilde{R}(r-1,m-1)$, $\underline{g} = (c_1,\ldots,c_{2^{m-1}})$, $\underline{h} = (d_1,\ldots,d_{2^{m-1}})$.

Als Polynom in $x_1,\ldots,x_m$ betrachtet liefert g dann das Codewort

$$(c_1,\ldots,c_{2^{m-1}},c_1,\ldots,c_{2^{m-1}})$$

und

$$\begin{aligned} x_m h &= (0,\ldots,0,1,\ldots,1)(d_1,\ldots,d_{2^{m-1}},d_1,\ldots,d_{2^{m-1}}) \\ &= (0,\ldots,0,d_1,\ldots,d_{2^{m-1}}). \end{aligned}$$

Also ist $\underline{f} = (c_1,\ldots,c_{2^{m-1}},c_1+d_1,\ldots,c_{2^{m-1}}+d_{2^{m-1}})$.

Damit $\widetilde{R}(r,m) \subseteq \widetilde{R}(r,m-1) \propto \widetilde{R}(r-1,m-1)$.

Umgekehrt liefert jedes Wort aus $\widetilde{R}(r,m-1) \propto \widetilde{R}(r-1,m-1)$ ein Codewort aus $\widetilde{R}(r,m)$.
Also: $\widetilde{R}(r,m) = \widetilde{R}(r,m-1) \propto \widetilde{R}(r-1,m-1)$.
Damit folgt die Behauptung. □

7.10 Satz.
Für $0 \leq r \leq m-1$ *ist*

$$RM(r,m)^{\perp} = RM(m-r-1,m).$$

(Insbesondere ist $RM(r,2r+1)$ *ein selbstdualer Code der Dimension* 2^{2r} *und der Länge* 2^{2r+1}*).*

Beweis.
Sei $\underline{f} = (a_1, \ldots, a_{2^m}) \in RM(r,m)$ und $\underline{g} = (b_1, \ldots, b_{2^m}) \in RM(m-r-1,m)$, wobei $f(x_1, \ldots, x_m)$ vom Grad $\leq r$, $g(x_1, \ldots, x_m)$ vom Grad $\leq m-r-1$.
Dann ist grad $(f \cdot g) \leq m-r-1+r = m-1$. Damit ist $\underline{f \cdot g} \in RM(m-1,m)$ und hat nach 7.4.d) gerades Gewicht.
Daher ist $< f, g >= 0$, denn $< f, g >$ ist die Summe der Einträge in $\underline{f} \cdot \underline{g}$ (gerechnet in $\mathbb{Z}_2$).

Also: $\quad RM(m-r-1,m) \subseteq RM(r,m)^{\perp}$.

Es ist

$$\begin{aligned}\dim(RM(r,m)^{\perp}) &= 2^m - \dim(RM(r,m)) \\ &= 2^m - \sum_{i=0}^{r} \binom{m}{i} = \sum_{i=r+1}^{m} \binom{m}{i} = \sum_{i=0}^{m-r-1} \binom{m}{i} \\ &= \dim(RM(m-r-1,m)).\end{aligned}$$

Daraus folgt die Behauptung. □

7.11 Bemerkung.
Mit Hilfe von 7.10 kann man zeigen, dass $RM(m-2,m) = \hat{\mathcal{H}}_m$, $\mathcal{H}_m$ der binäre Hamming-Code. (Man zeigt: $RM(1,m) = \hat{\mathcal{H}}_m^{\perp}$ und verwendet dann 7.10)

7.12 Bemerkung.

a) Ist $x = (x_1, \ldots, x_m) \in \mathbb{Z}_2^m$, so ordne x die Zahl

$$z(x) = \sum_{i=1}^{m} x_i 2^{i-1} \in [0, \ldots, 2^m - 1]$$

zu. In der Wertetabelle einer Boole'schen Funktion sind also die Vektoren $x \in \mathbb{Z}_2^m$ gerade in aufsteigender Reihenfolge der $z(x)$ angeordnet.
$z(x)$=Binärzahl die durch $x_m \ldots x_1$ repräsentiert wird.

b) Es gibt eine bijektive Beziehung zwischen

1) Boole'sche Funktionen $\mathbb{Z}_2^m \longrightarrow \mathbb{Z}_2$ (= Boole'sche Polynome)
2) Vektoren des $(\mathbb{Z}_2)^{2^m}$
3) Teilmengen des $\mathbb{Z}_2^m$

$1 \to 2$: $f \longmapsto \underline{f}$ (injektiv, also bijektiv)
$2 \to 3$: $a = (a_0, \ldots, a_{2^m-1}) \longmapsto$
$M = \{x \in \mathbb{Z}_2^m : \ a_{z(x)} = 1\}$ (injektiv)
$3 \to 1$: $M \subseteq \mathbb{Z}_2^m \longmapsto$ *charakteristische Funktion* χ_M:

$$\chi_M(x) = \begin{cases} 1, & x \in M \\ 0, & x \notin M \end{cases}, x \in \mathbb{Z}_2^m \quad \text{(injektiv)}$$

Beispiel:

$f : \mathbb{Z}_2^3 \longrightarrow \mathbb{Z}_2$ gegeben durch folgende Wertetabelle:

x_3	0	0	0	0	1	1	1	1
x_2	0	0	1	1	0	0	1	1
x_1	0	1	0	1	0	1	0	1
$f(x_1, x_2, x_3)$	0	0	1	0	1	1	0	0

$$\underline{f} = (\underbrace{0}_{\underline{f}_0} 010110 \underbrace{0}_{\underline{f}_7})$$

$$\underline{f} \mapsto \{x \in \mathbb{Z}_2^m : \ \underline{f}_{z(x)} = 1\} = \{(010), (001), (101)\} =: M$$

$$f = \chi_M$$

Wann ist $\underline{\chi}_M$ ein Boole'sches Polynom vom Grad $\leq r$, liegt also in $RM(r, m)$ (nach 7.9)?
Wir werden nur eine hinreichende Bedingung angeben, die für unsere Zwecke ausreicht:

7.13 Satz.
Sei M ein t-dimensionaler Unterraum von $\mathbb{Z}_2^m$, $t \geq m - r$.
Dann ist $\underline{\chi}_M \in RM(r, m)$.

Beweis:
Als t-dimensionaler Unterraum lässt sich M als Lösungsmenge eines homogenen Gleichungssystems in m Unbekannten mit $m-t$ Gleichungen beschreiben (Koeffizientenmatrix = Kontrollmatrix von M).

$$\begin{array}{rcl} a_{11}x_1 & + \cdots + & a_{1m}x_m = 0 \\ \vdots & & \vdots \quad \vdots \\ a_{m-t,1}x_1 & + \cdots + & a_{m-t,m}x_m = 0 \end{array}$$

Setzte $p_i(x_1, \ldots, x_m) = a_{i1}x_1 + \ldots + a_{im}x_m + 1$, Boole'sches Polynom vom Grad ≤ 1; $p(x_1, \ldots, x_m) = \prod_{i=1}^{m-t} p_i(x_1, \ldots, x_m)$, Boole'sches Polynom vom Grad $\leq m - t \leq r$, das heißt $\underline{p} \in RM(r, m)$.
Nun gilt für $(y_1, \ldots, y_m) \in \mathbb{Z}_2^m$:

$$\begin{aligned} p(y_1, \ldots, y_m) = 1 &\iff p_i(y_1, \ldots y_m) = 1,\ i = 1, \ldots, m-t \\ &\iff a_{i1}y_1 + \ldots + a_{im}y_m = 0,\ i = 1, \ldots, m-t \\ &\iff (y_1, \ldots, y_m) \in M. \end{aligned}$$

Also: $\underline{\chi}_M = \underline{p} \in RM(r, m)$. □

7.14 Beispiel.
$m = 3$, $r = 1$
$M = \{\underbrace{(0,0,0)}_{=v_1}, \underbrace{(1,0,1)}_{=v_2}, \underbrace{(0,1,1)}_{=v_3}, \underbrace{(1,1,0)}_{=v_4}\}$
(2-dimensionale Unterräume über $\mathbb{Z}_2$ sind immer von der Form $\{0, a, b, a+b\}$)

$$z(v_1) = 0, \quad z(v_2) = 5, \quad z(v_3) = 6, \quad z(v_4) = 3$$

$\underline{\chi}_M = (10010110)$

$$(a_{11}, a_{12}, a_{13}) \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = 0$$

Kontrollmatrix für M:
(1×3)-Matrix

$$(a_{11}, a_{12}, a_{13}) \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} = 0$$

$$\begin{aligned} a_{11} + \qquad a_{13} &= 0 \\ a_{12} + a_{13} &= 0 \end{aligned}$$

Lösungsraum wird erzeugt von $(1, 1, 1)$.

Zugehöriges Polynom: $p(x_1, x_2, x_3) = x_1 + x_2 + x_3 + 1$, $\underline{p} = \underline{\chi}_M$
Also: $\underline{\chi}_M \in RM(1, 3)$.

Man hätte p natürlich auch nach dem Verfahren von 7.6 bestimmen können:

$$\begin{aligned}
&\bar{x}_1\bar{x}_2\bar{x}_3 \vee x_1\bar{x}_2x_3 \vee \bar{x}_1x_2x_3 \vee x_1x_2\bar{x}_3 = \\
&(\bar{x}_1\bar{x}_2\bar{x}_3 + x_1\bar{x}_2x_3 + \underbrace{\bar{x}_1\bar{x}_2\bar{x}_3x_1\bar{x}_2x_3}_{=0}) \vee (\bar{x}_1x_2x_3 + x_1x_2\bar{x}_3 + \underbrace{\bar{x}_1x_2x_3x_1x_2\bar{x}_3}_{=0}) \\
&= \bar{x}_1\bar{x}_2\bar{x}_3 + x_1\bar{x}_2x_3 + \bar{x}_1x_2x_3 + x_1x_2\bar{x}_3 + \underbrace{(\bar{x}_1\bar{x}_2\bar{x}_3 + x_1\bar{x}_2x_3)(\bar{x}_1x_2x_3 + x_1x_2\bar{x}_3)}_{=0} \\
&= (1+x_1)(1+x_2)(1+x_3) + x_1(1+x_2)x_3 + (1+x_1)x_2x_3 + x_1x_2(1+x_3) \\
&= (1+x_1)(1+x_2+x_3+x_2x_3) + x_1x_3 + x_1x_2x_3 + x_2x_3 + x_1x_2x_3 + x_1x_2 + x_1x_2x_3 \\
&= 1 + x_2 + x_3 + x_2x_3 + x_1 + x_1x_2 + x_1x_3 + x_1x_2x_3 + x_1x_3 + x_2x_3 + x_1x_2 + x_1x_2x_3 \\
&= 1 + x_1 + x_2 + x_3
\end{aligned}$$

7.15 Bemerkung.
Man kann zeigen, dass
$RM(r,m) = \langle \underline{\chi}_M \in \mathbb{Z}_2^{2^m} \ : \ M$ ist t-dimensionaler affiner Unterraum von $\mathbb{Z}_2^m, t \geq m-r \rangle$
Affiner Unterraum $= y + M$, M linearer Unterraum, $y \in \mathbb{Z}_2^m$
(Nebenklassen der linearen Unterräume)

Beweis: siehe Skript Codierungstheorie SS 2004, Satz 5.8, oder
MacWilliams, Sloane, Chapter 13, §6.

Damit beenden wir unsere Beschreibung der Reed-Muller-Codes und wenden uns dem Verfahren der Majority-Logic-Decodierung zu. Wir beschreiben dies zunächst für beliebige lineare Codes und zeigen dann, wie es für Reed-Muller-Codes besonders effizient anwendbar ist.

7.16 Definition.
Sei $\mathcal{C}$ ein $[n,k]$-Code über einem endlichen Körper K, $y^{(1)}, \ldots, y^{(\ell)} \in \mathcal{C}^\perp$ (d.h. $< y^{(i)}, x >= 0$ für $j = 1, \ldots, \ell$ und alle $x \in \mathcal{C}$), $y^{(j)} = (y_1^{(j)}, \ldots, y_n^{(j)})$.
$y^{(1)}, \ldots, y^{(\ell)}$ heißen *orthogonal bezüglich Position* i $(1 \leq i \leq n)$ falls gilt:

(1) $y_i^{(j)} = 1$ für alle $j = 1, \ldots, \ell$.

(2) Ist $r \neq i$, so ist $y_r^{(j)} \neq 0$ für höchstens ein j.

7.17. Majority-Logic-Decodierung

Geg : $[n,k]$-Code $\mathcal{C}$, Position i $(1 \leq i \leq n)$, $y^{(1)}, \ldots, y^{(\ell)} \in \mathcal{C}^\perp$ orthogonal bezüglich Position i.

Ang : $x = (x_1, \ldots, x_n) \in \mathcal{C}$ wird gesendet, $z = (z_1, \ldots, z_n) \in K^n$ wird empfangen, das t Fehler enthält, $t \leq \frac{1}{2}\ell$. Die fehlerhaften Stellen seien $z_{k_1}, \ldots, z_{k_t}$.

Frage : Ist z_i fehlerhaft oder nicht?

Es ist $< y^{(j)}, x >= y_1^{(j)} x_1 + \ldots + y_n^{(j)} x_n = 0$ für $j = 1, \ldots, \ell$.
Wir berechnen $< y^{(j)}, z >$ und betrachten 2 Fälle:

1.Fall : z_i ist fehlerhaft, etwa $z_i = z_{k_1}$.
Dann $y_i^{(j)} z_i \neq y_i^{(j)} x_i, \;\; j = 1, \ldots, \ell$.
Es gibt wegen Bedingung (2) aus 7.16 maximal $t-1$ viele $y^{(j)}$, die an einer der Stellen $k_2, \ldots, k_t$ einen Eintrag $\neq 0$ haben.
Für die übrigen $\ell - (t-1)$ vielen $y^{(j)}$ gilt also:
Ist $r \neq i$, so ist $y_r^{(j)} x_r = y_r^{(j)} z_r$
(denn entweder ist $x_r = z_r$ oder $y_r^{(j)} = 0$ (oder beides))
Also gilt für diese j: $< y^{(j)}, z >=< y^{(j)}, z > - \underbrace{< y^{(j)}, x >}_{=0} = y_i^{(j)} z_i - y_i^{(j)} x_i \neq 0$

2.Fall : z_i ist korrekt.
Es gibt maximal t viele $y^{(j)}$, die an einer Stelle $k_1, \ldots, k_t$ einen Eintrag $\neq 0$ haben.
Die übrigen $\ell - t$ vielen haben an diesen Stellen Eintrag 0.
Also gilt für diese $y^{(j)}$: $y_r^{(j)} x_r = y_r^{(j)} z_r$ für alle r,
das heißt $< y^{(j)}, z >=< y^{(j)}, x >= 0$.

Fasst man beide Fälle zusammen, so erhält man:

$$< y^{(j)}, z >\neq 0 \text{ für } \begin{cases} \leq t \text{ Werte von } j, \text{ falls } z_i \text{ korrekt} \\ \geq \ell - (t-1) \text{ Werte von } j, \text{ falls } z_i \text{ fehlerhaft} \end{cases}$$

Da $t \leq \frac{1}{2}\ell$, ist $2t \leq \ell$, das heißt $t \leq \ell - t$ und $\ell - (t-1) \geq (t+1)$; also folgt:
Ist z_i fehlerhaft, so ist die Mehrheit der Werte $< y^{(j)} z >\neq 0$
Ist z_i korrekt, so ist das nicht der Fall
(maximal $t \leq \frac{\ell}{2}$ viele $\neq 0$, mindestens $\ell - t \geq \frac{\ell}{2}$ viele $= 0$).

Damit :
Mehrheit der Werte $< y^{(j)}, z >\neq 0 \iff z_i$ ist fehlerhaft.
Das ist die Majority-Logic-Decodierung. Im binären Fall kann z_i natürlich korrigiert werden, wenn es fehlerhaft ist.

7.18 Beispiel.
Sei $\mathcal{C}$ der [7,3]-Simplex-Code (der duale Code zum [7,4]-Hamming-Code über $\mathbb{Z}_2$).

Erzeugermatrix von $\mathcal{C}^\perp$=Erzeugermatrix des Hamming-Codes

$$\tilde{H} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \text{ (vgl. Beispiel 5.8)}$$

Wir wählen:

$$\begin{aligned} y^{(1)} &= \text{erste Zeile} \\ y^{(2)} &= \text{dritte Zeile} \\ y^{(3)} &= \text{Summe der zweiten und vierten Zeile} \end{aligned}$$

$$\begin{aligned} y^{(1)} &= (1101000) \\ y^{(2)} &= (1010010) \\ y^{(3)} &= (1000101) \end{aligned}$$

$y^{(1)}, y^{(2)}, y^{(3)}$ sind orthogonal bezüglich Position 1.
Angenommen $z = (0100110)$ wird empfangen, maximal 1 Fehler sei aufgetreten. (Nach 6.5 hat jedes Codewort $\neq 0$ in $\mathcal{C}$ Gewicht $2^{3-1} = 4$. d.h. in z ist auch tatsächlich ein Fehler aufgetreten.)

$$\langle y^{(1)}, z\rangle = 1, \quad \langle y^{(2)}, z\rangle = 1, \quad \langle y^{(3)}, z\rangle = 1.$$

$\Rightarrow$ Stelle z_1 ist fehlerhaft.
Also wird zu $x = (1100110)$ decodiert.
(Tatsächlich $x \in \mathcal{C}$, da $\tilde{H}x = 0$: beachte $\tilde{H}$ ist Kontrollmatrix von $\mathcal{C}$.)

Angenommen $z = (0110010)$ wird empfangen, maximal 1 Fehler sei aufgetreten (dann wieder genau 1 Fehler)

$$\langle y^{(1)}, z\rangle = 1, \quad \langle y^{(2)}, z\rangle = 0, \quad \langle y^{(3)}, z\rangle = 0.$$

$\Rightarrow z_i$ korrekt.
(Über die übrigen Stellen lässt sich mit $y^{(1)}, y^{(2)}, y^{(3)}$ nichts aussagen!)

Wir verallgemeinern jetzt 7.17, der Einfachheit halber nur für binäre Codes.

7.19 Definition.
Sei $\mathcal{C}$ binärer $[n, k]$-Code, $y^{(1)}, \ldots, y^{(\ell)} \in \mathcal{C}^\perp$, $y^{(i)} = (y_1^{(i)}, \ldots, y_n^{(i)})$.
$y^{(1)}, \ldots, y^{(\ell)}$ heißen *orthogonal bezüglich der Positionen* $i_1, \ldots, i_s$,
falls gilt:

(1) $y_{i_1}^{(j)} = \ldots = y_{i_s}^{(j)} = 1$ für alle $j = 1, \ldots, \ell$.

(2) Ist $r \neq i_1, \ldots, i_s$, so ist $y_r^{(j)} \neq 0$ höchstens für ein j.

7.20 Satz (Multistep-Majority-Decodierung)**.**
bezeichnungen wie in 7.19. Wird ein Wort $x \in \mathcal{C}$ gesendet und z empfangen und sind maximal $t \leq \frac{1}{2}\ell$ Fehler aufgetreten, so gilt:
Die Anzahl der Fehler in z an den Stellen $i_1, \ldots, i_s$ ist ungerade
$\iff \sharp\{j : \langle y^{(j)}, z\rangle = 1\} \quad > \quad \sharp\{j : \langle y^{(j)}, z\rangle = 0\}$

Beweis:
1.Fall: Anzahl der Fehler in z an den Stellen $i_1, \ldots, i_s$ ist ungerade (also mindestens 1):

Dann maximal $t-1$ viele Fehler außerhalb $i_1, \ldots, i_s$.
Daher nur maximal $t-1$ viele $y^{(j)}$, die an einer dieser Stellen Eintrag 1 haben.
Für die übrigen gilt

$y_r^{(j)} z_r = y_r^{(j)} x_r \quad \forall r \neq i_1, \ldots, i_s$ (entweder $z_r = x_r$ oder $y_r^{(j)} = 0$ (oder beides))

Für diese $y^{(j)}$ gilt also:

$$\begin{aligned} < y^{(j)}, z > &= < y^{(j)}, z > - \underbrace{< y^{(j)}, x >}_{=0} \\ &= (z_{i_1} - x_{i_1}) + \ldots + (z_{i_s} - x_{i_s}) \text{ (Bedingung (1) von 7.19)} \\ &= \text{Anzahl der Fehler an den Stellen } i_1, \ldots, i_s \mod 2 = 1 \end{aligned}$$

Also: Für mindestens $\ell - (t-1)$ viele $y^{(j)}$ ist $< y^{(j)}, z >= 1$.

2. Fall : Anzahl der Fehler in z an den Stellen $i_1, \ldots, i_s$ ist gerade

Dann maximal t Fehler außerhalb $i_1, \ldots, i_s$, also maximal t viele $y^{(j)}$, die an einer dieser Stellen Eintrag 1 haben.
Für die übrigen $\ell - t$ vielen $y^{(j)}$ gilt

$$y_r^{(j)} z_r = y^{(j)} x_r \quad \forall r \neq i_1, \ldots, i_s$$

also

$$\begin{aligned} < y^{(j)}, z > &= (z_{i_1} - x_{i_1}) + \ldots + (z_{i_s} - x_{i_s}) \\ &= \text{Anzahl der Fehler an den Stellen } i_1, \ldots, i_s \mod 2 \\ &= 0 \end{aligned}$$

Also: Für mindestens $\ell - t$ viele $y^{(j)}$ ist $< y^{(j)}, z >= 0$, das heißt für höchstens t viele $y^{(j)}$ ist $< y^{(j)}, z >= 1$
Da $\ell - (t-1) > t$, folgt die Behauptung wie in 7.17. □

Für Reed-Muller-Codes lässt sich die Multistep-Majority-Decodierung besonders einfach durchführen, falls maximal $t \leq \left\lfloor \frac{d(\mathcal{C})-1}{2} \right\rfloor = 2^{m-r-1} - 1$ Fehler bei der Übertragung aufgetreten sind. (Man findet dann nämlich genügend viele Vektoren in $RM(r,m)^{\perp}$, die orthogonal bezüglich gewisser Positionen sind.)

Wir benötigen folgendes Lemma:

7.21 Lemma.
Sei M ein p-dimensionaler Unterraum von $\mathbb{Z}_2^m$, $p < r$.
Dann gibt es genau $2^{m-p} - 1$ $(p+1)$-dimensionale Unterräume von $\mathbb{Z}_2^m$, die M enthalten.
Sind M_1, M_2 zwei solche (verschiedenen), so ist $M_1 \cap M_2 = M$.

Beweis:
Es ist $|M| = 2^p$. Ist $v \in \mathbb{Z}_2^m \setminus M$, so ist $\langle M \cup \{v\} \rangle$ ein $(p+1)$-dimensionaler Unterraum, der M enthält. Er enthält $2^{p+1} - 2^p = 2^p$ viele Vektoren $\notin M$. Jeder von diesen spannt also mit M den gleichen Unterraum auf. Da es $2^m - 2^p$ viele Vektoren außerhalb von M gibt, ist die gesuchte Anzahl gerade $\frac{2^m - 2^p}{2^p} = 2^{m-p} - 1$. Es ist $M_1 \supsetneqq M_1 \cap M_2 \geq M$ und daher $p+1 > \dim(M_1 \cap M_2) \geq p$, also $\dim(M_1 \cap M_2) = p = \dim(M)$, $M_1 \cap M_2 = M$. □

7.22. Multistep-Majority-Decodierung für Reed-Muller-Codes

(1) $\mathcal{C} = RM(r,m)$, $r < m$, $n = 2^m$, $k = \sum_{i=0}^{r} \binom{m}{i}$, $d = 2^{m-r}$.

$x = (x_0, \ldots, x_{2^m-1})$ gesendet,
$z = (z_0, \ldots, z_{2^m-1})$ empfangen.

Annahme : z enthalte t Fehler, $t \leq 2^{m-r-1} - 1 = \left\lfloor \frac{d-1}{2} \right\rfloor$.

Ziel : Decodierverfahren, das z in x decodiert (und in der Regel schneller ist als Syndrom-Decodierung.)

Folgende Bezeichnung für die Elemente von $\mathbb{Z}_2^m$:

v_0	v_1	v_2	$\ldots$	v_{2^m-1}
0	0	0		1
$\vdots$	$\vdots$	$\vdots$		$\vdots$
$\vdots$	0	1		1
0	1	0		1

$M \subseteq \mathbb{Z}_2^m$: Positionen von $M = \{i : v_i \in M\} = \{i : \underline{\chi}_M$ hat 1 an Position $i\}$.
Wir benötigen jetzt:

(2) Sei M p-dimensionaler Unterraum von $\mathbb{Z}_2^m$, $p \leq r$.
$M_1, \ldots, M_w$, $w = 2^{m-p} - 1$, die sämtlichen $(p+1)$-dimensionalen Unterräume von $\mathbb{Z}_2^m$, die M enthalten (7.21). Dann gilt:

Die Anzahl der Fehler in z an den Positionen von M ist ungerade
$\Longleftrightarrow$
$\sharp\{j$: Anzahl der Fehler in z an den Positionen von M_j ist ungerade$\}$
$>$
$\sharp\{j$: Anzahl der Fehler in z an den Positionen von M_j ist gerade$\}$:

$M_i \cap M_j = M$ für $i \neq j$. Daher hat jedes $\underline{\chi}_{M_i}$ an den Positionen von M eine 1 und außerhalb von M haben $\underline{\chi}_{M_i}$ und $\underline{\chi}_{M_j}$ für $i \neq j$ an keiner Position eine 1 gemeinsam (vergleiche 7.19).
Angenommen, Anzahl der Positionen von M an denen z fehlerhaft ist, ist ungerade.
Maximal $t-1$ viele der $\underline{\chi}_{M_j}$ haben eine 1 an einer fehlerhaften Position außerhalb von M.
Der Rest, also mindestens $w-(t-1)$ viele, trifft mit seinen Einsen außer auf korrekte Positionen nur auf die fehlerhaften Positionen von M.
Also: Es gibt mindestens $w-(t-1)$ viele M_j, an deren Positionen z eine ungerade Anzahl von Fehlern hat.

Angenommen, Anzahl der Positionen von M, an denen z fehlerhaft ist, ist gerade.
Maximal t viele der $\underline{\chi}_{M_j}$ haben 1 an fehlerhaften Positionen von z außerhalb von M.
Also gibt es mindestens $w-t$ viele M_j, an deren Positionen z eine gerade Anzahl von Fehlern hat.
Das heißt, es gibt maximal t viele, an deren Positionen z eine ungerade Anzahl von Fehlern hat.
Es ist $2t \leq 2^{m-r} - 2 < 2^{m-p} - 1 = w$, das heißt $t < w - t$ und daher auch $w-(t-1) > t-1$. Behauptung folgt.

(3) Beschreibung des Decodierverfahrens
1. Schritt :
Wähle r-dimensionalen Unterraum von $\mathbb{Z}_2^m$; also $v_0 \in M$.
Seien $M_1, \ldots, M_\ell$ alle $(r+1)$-dimensionalen Unterräume von $\mathbb{Z}_2^m$, die M enthalten; $\ell = 2^{m-r} - 1$ nach 7.21. Nach 7.10 ist $RM(r,m)^\perp = RM(m-r-1, m)$.
Nach 7.13 liegen alle $\underline{\chi}_{M_j}$ in $RM(r,m)^\perp$, da $r+1 = m-(m-r-1)$.
Wegen $M_i \cap M_j = M$ für $i \neq j$ sind die $\underline{\chi}_{M_j}$ orthogonal bezüglich der Positionen von M (Definition 7.19)

Berechne alle $\langle \underline{\chi}_{M_j}, z \rangle$.

Mehrzahl der Werte $=1 \iff$ Anzahl der Fehler in z an den Positionen von M ist ungerade (7.20),

da $t \leq 2^{m-r-1} - 1 \leq \frac{1}{2}(\underbrace{2^{m-r} - 1}_{\ell})$.

2. Schritt :
Führe den 1. Schritt für alle r-dimensionalen Unterräume von $\mathbb{Z}_2^m$ durch.
Nach dem 2. Schritt :
Für jeden r-dimensionalen Unterraum ist bekannt, ob die Anzahl der Fehler von z an dessen Positionen gerade ober ungerade ist.

3. Schritt :
Wähle $(r-1)$-dimensionalen Unterraum N von $\mathbb{Z}_2^m$ und bestimme alle r-dimensionalen Unterräume von $\mathbb{Z}_2^m$, die N enthalten.
Aufgrund von Schritt 2 und (2) können wir bestimmen, ob die Anzahl der Fehler in z an den Positionen von N gerade oder ungerade ist.
Führe dies für alle $(r-1)$-dimensionalen Unterräume durch.

4. Schritt :
Führe den 3. Schritt für alle $(r-2), \ldots, 1, 0$-dimensionalen Unterräume durch.

0-dimensionaler Fall: $\{v_0 = 0\}$.
Wir wissen daher, ob z_0 korrekt ist oder nicht. Da wir auch wissen, ob die Anzahl der Fehler von z an den Positionen der 1-dimensionalen Unterräume $\{v_0, v_i\}$ gerade oder ungerade ist, können wir auch feststellen, ob z_i korrekt ist, $i = 1, \ldots, 2^m - 1$.

7.23 Bemerkung.
Ist $r \approx \frac{m}{2}$, so benötigt das geschilderte Verfahren $O(m2^{m^2+m})$ binäre Operationen.
Syndrom-Decodierung braucht $O(2^{2^{m-1}})$ viele Berechnungen von Nebenklassenführern und Vergleiche.
Für große m ist der Aufwand für Syndrom-Decodierung also erheblich höher.
Es gibt Modifikationen und effizientere Versionen des vorgestellten Verfahrens: MacWilliams, Sloane, Chapter 10.

7.24 Beispiel.
Multistep Majority-Decodierung des Reed-Muller-Codes $RM(1,4)$

$n = 2^m = 16, \quad k = 1 + 4 = 5, \quad d = 2^3 = 8$
$RM(1,4)$ ist 3-Fehler-korrigierend.

Wir wählen folgende Bezeichnungen für die Elemente des $\mathbb{Z}_2^4$:

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}	v_{15}
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Die durch v_i dargestellte Zahl in Binärschreibweise (niedrigstes Bit unten) gibt die Position in der Tabelle an (v_0 Position 0,...,v_{15} Position 15).

Angenommen, ein Wort $x = (x_0, \ldots, x_{15}) \in R(1,4)$ wird gesendet und $z = (1011010001001010)$ wird empfangen.
Wir nehmen an, dass maximal drei Fehler aufgetreten sind und wollen z mit Multistep Majority-Logic decodieren.

Da $r = 1$, bestimmen wir zunächst durch Betrachtung aller 1-dimensionalen Unterräume $\{v_0, v_i\}, i = 1, \ldots, 15$, des $\mathbb{Z}_2^4$ (beachte v_0 ist der Nullvektor), ob an den zugehörigen Positionen 0 und i von z eine gerade oder ungerade Anzahl von Fehlern aufgetreten ist.

Wir beginnen mit $M = \{v_0, v_1\}$ und müssen für alle 2-dimensionalen Unterräume M_j des $\mathbb{Z}_2^4$, die M enthalten, überprüfen, ob $\langle z, \underline{\chi}_{M_j} \rangle$ gleich 0 oder 1 ist.

Es gibt sieben solcher Unterräume, nämlich
$M_1 = \{v_0, v_1, v_2, v_3\}, M_2 = \{v_0, v_1, v_4, v_5\}, M_3 = \{v_0, v_1, v_6, v_7\}, M_4 = \{v_0, v_1, v_8, v_9\}$,
$M_5 = \{v_0, v_1, v_{10}, v_{11}\}, M_6 = \{v_0, v_1, v_{12}, v_{13}\}, M_7 = \{v_0, v_1, v_{14}, v_{15}\}$.
Man erhält sie, indem man neben v_0 und v_1 einen dritten Vektor v_s wählt; der vierte Vektor ist dann $v_1 + v_s$, also gerade v_{s+1}.

$\langle z, \underline{\chi}_{M_1} \rangle$ rechnet man aus, indem man die Einsen in z, die an den durch M_1 bestimmten Positionen 0,1,2,3 stehen, addiert (modulo 2), also $1 + 1 + 1 = 1$. Analog verfährt man mit den übrigen M_j. Dann erhält man:

j	1	2	3	4	5	6	7
$\langle z, \underline{\chi}_{M_j} \rangle$	1	0	1	0	1	0	0

Da die Mehrzahl der Werte von $\langle z, \underline{\chi}_{M_j} \rangle$ gleich 0 ist, enthält z an den Positionen 0,1 eine gerade Anzahl von Fehlern.

Diese Rechnung haben wir jetzt für alle $M = \{v_0, v_i\}$ durchzuführen. Die Ergebnisse sind in folgender Tabelle zusammengefasst:

M	M_j	$\langle z, \underline{\chi}_{M_j} \rangle$	
$\{v_0, v_1\}$ siehe oben			Anzahl der Fehler an den Positionen 0,1 gerade
$\{v_0, v_2\}$	$\{v_0, v_2, v_1, v_3\}$	1	Anzahl Nullen
	$\{v_0, v_2, v_4, v_6\}$	0	$>$
	$\{v_0, v_2, v_5, v_7\}$	1	Anzahl Einsen
	$\{v_0, v_2, v_8, v_{10}\}$	0	Daher:
	$\{v_0, v_2, v_9, v_{11}\}$	1	Anzahl der Fehler
	$\{v_0, v_2, v_{12}, v_{14}\}$	0	an den Positionen 0,2
	$\{v_0, v_2, v_{13}, v_{15}\}$	0	gerade
$\{v_0, v_3\}$	$\{v_0, v_3, v_1, v_2\}$	1	Anzahl Nullen
	$\{v_0, v_3, v_4, v_7\}$	0	$<$
	$\{v_0, v_3, v_5, v_6\}$	1	Anzahl Einsen
	$\{v_0, v_3, v_8, v_{11}\}$	0	Daher:
	$\{v_0, v_3, v_9, v_{10}\}$	1	Anzahl der Fehler
	$\{v_0, v_3, v_{12}, v_{15}\}$	1	an den Positionen 0,3
	$\{v_0, v_3, v_{13}, v_{14}\}$	1	ungerade
$\{v_0, v_4\}$	$\{v_0, v_4, v_1, v_5\}$	0	Anzahl Nullen
	$\{v_0, v_4, v_2, v_6\}$	0	$>$
	$\{v_0, v_4, v_3, v_7\}$	0	Anzahl Einsen
	$\{v_0, v_4, v_8, v_{12}\}$	0	Daher:
	$\{v_0, v_4, v_9, v_{13}\}$	0	Anzahl der Fehler
	$\{v_0, v_4, v_{10}, v_{14}\}$	0	an den Positionen 0,4
	$\{v_0, v_4, v_{11}, v_{15}\}$	1	gerade

Analog ergeben sich die übrigen Fälle:

M	Anzahl der Fehler
$\{v_0, v_5\}$	an den Positionen 0,5: gerade
$\{v_0, v_6\}$	an den Positionen 0,6: gerade
$\{v_0, v_7\}$	an den Positionen 0,7: ungerade
$\{v_0, v_8\}$	an den Positionen 0,8: gerade
$\{v_0, v_9\}$	an den Positionen 0,9: gerade
$\{v_0, v_{10}\}$	an den Positionen 0,10: gerade
$\{v_0, v_{11}\}$	an den Positionen 0,11: ungerade
$\{v_0, v_{12}\}$	an den Positionen 0,12: gerade
$\{v_0, v_{13}\}$	an den Positionen 0,13: gerade
$\{v_0, v_{14}\}$	an den Positionen 0,14: gerade
$\{v_0, v_{15}\}$	an den Positionen 0,15: gerade

Da die Mehrzahl der Fehleranzahlen in den Positionen 0, i gerade ist, ist die Fehleranzahl an der Position 0 gerade, d.h. die Position 0, also die erste Koordinate von z ist korrekt. Da wir wissen, ob die Fehleranzahl an einem Positionenpaar 0, i gerade oder ungerade ist, folgt jetzt auch, dass genau an den Positionen 3, 7, 11 Fehler aufgetreten sind.
z wird also decodiert zum Codewort $x = (1010010101011010)$.

8 Polynomcodierung und zyklische Codes

8.1 Exkurs (Polynomringe, I)**.**
$K[x]$ Polynomring über Körper K (formale Polynome).

(1) Addition:

$$\sum_{i=0}^{n} a_i x^i + \sum_{i=0}^{n} b_i x^i := \sum_{i=0}^{n} (a_i + b_i) x^i$$

Multiplikation mit Skalaren:

$$a \cdot \sum_{i=0}^{n} a_i x^i := \sum_{i=0}^{n} (a a_i) x^i$$

Damit ist $K[x]$ ein Vektorraum über K.
Für jedes $n \geq 0$ ist

$$K[x]_n := \{f \in K[x] : \ \mathrm{Grad}(f) < n\}$$

ein Untervektorraum von $K[x]$ ($\mathrm{Grad}(0) := -\infty$).

(2) Multiplikation von Polynomen:

$$\left(\sum_{i=0}^{n} a_i x^i\right) \cdot \left(\sum_{i=0}^{m} b_i x^i\right) = \sum_{i=0}^{n+m} c_i x^i$$

mit $c_i := a_0 b_i + a_1 b_{i-1} + \ldots + a_i b_0 = \sum_{j=0}^{i} a_j b_{i-j}$,
wobei $a_j = 0$ für $j > n$ und $b_j = 0$ für $j > m$ zu setzen ist.
Multiplikation mit Skalaren entspricht der Multiplikation mit Polynomen vom Grad ≤ 0.
Beispiel: $K = \mathbb{Z}_2$

$$(x^2 + x + 1)(x^3 + x^2 + 1) = x^5 + x + 1$$

(3) $K[x]$ ist mit der in (1) definierten Addition und der Multiplikation aus (2) ein kommutativer Ring mit Eins.

(4) In $K[x]$ gibt es *Division mit Rest*: Seien $f, g \in K[x]$ und $g \neq 0$. Dann gibt es eindeutig bestimmte Polynome $h, r \in K[x]$ mit

$$f = g \cdot h + r$$

und grad $r <$ grad g.
Bezeichnung: $r = f \mod g$.

Beispiel: $K = \mathbb{Z}_2$, $f = x^4 + x^2 + 1$, $g = x^2 + x + 1$

$$\begin{array}{l} x^4{+}x^2 + 1 : x^2 + x + 1 = x^2 - x + 1 \\ \underline{-(x^4 + x^3 + x^2)} \\ \quad x^3{+}1 \\ \quad \underline{-(-x^3 - x^2 - x)} \\ \qquad -x^2 + x{+}1 \\ \qquad \underline{-(x^2 + x + 1)} \\ \qquad\qquad 0 \end{array}$$

In obiger Notation ergibt sich $h = x^2 - x + 1$ und $r = 0$, und somit ist $f \mod g = 0$. Das heißt g teilt f; $g \mid f$.

$f = x^4 + x + 1$, $g = x^2 + 1$

$$\begin{array}{l} x^4{+}x + 1 : x^2 + 1 = x^2 - 1 \\ \underline{-(x^4 + x^2)} \\ \quad -x^2 + x{+}1 \\ \quad \underline{-(-x^2 - 1)} \\ \qquad\qquad x + 2 \end{array}$$

In obiger Notation ergibt sich $h = x^2 - 1$ und $r = x + 2$, also

$$f \mod g = x + 2.$$

Ist $K = \mathbb{Z}_2$, so ist $f \mod g = x$.

(5) Ein Polynom $f \in K[x]$ vom Grad ≥ 1 heißt *irreduzibel*, falls f nicht Produkt zweier Polynome vom Grad ≥ 1 ist. (Entspricht einer Primzahl in $\mathbb{Z}$).

(6) Jedes Polynom $f \in K[x]$ vom Grad ≥ 1 lässt sich eindeutig schreiben in der Form

$$f = a \cdot f_1 \cdot \ldots \cdot f_s$$

Dabei ist $a \in K$ und f_i sind irreduzible Polynome mit höchstem Koeffizienten 1. (Entspricht der eindeutigen Primfaktorzerlegung in $\mathbb{Z}$).

Man kann K^n identifizieren mit $K[x]_n$:

$$(a_0, \ldots, a_{n-1}) \to a_0 + a_1 x + \ldots + a_{n-1} x^{n-1}$$

Vektorraumisomorphismus. Polynomcodierung benutzt diese Identifikation.

8.2. Polynomcodierung
Sei $g(x) = a_0 + \ldots + a_{\ell-1}x^{\ell-1} + x^\ell$ fest gegebenes Polynom über K (mit höchstem Koeffizienten $a_\ell = 1$).
Klartexte (beliebiger Länge k) über K ($m = m_0 \ldots m_{k-1}$) werden identifiziert mit $m(x) = m_0 + m_1 x + \ldots + m_{k-1}x^{k-1}$.
Multipliziere $m(x)$ mit x^ℓ:

$$\begin{aligned} x^\ell m(x) &= m_0 x^\ell + m_1 x^{\ell+1} + \ldots + m_{k-1}x^{\ell+k-1} \in K[x]_{\ell+k} \\ &\quad (\leftrightarrow (\underbrace{0, \ldots, 0}_{\ell}, m_0, \ldots, m_{k-1})) \end{aligned}$$

Codiere $m(x)$ in

$$x^\ell m(x) - (x^\ell m(x) \mod g(x)) =: c(x) \quad (= \text{ Vielfaches von } g(x))$$

Ist $x^\ell m(x) \mod g(x) = b_0 + b_1 x_1 + \ldots + b_{\ell-1}x^{\ell-1}$, so

$$\begin{aligned} c(x) &= -b_0 - b_1 x_1 - \ldots - b_{\ell-1}x^{\ell-1} + m_0 x^\ell + m_1 x^{\ell+1} + \ldots + m_{k-1}x^{\ell+k-1} \\ &\quad \leftrightarrow (\underbrace{-b_0, -b_1, \ldots, b_{\ell-1}}_{\text{'Kontrollstellen'}}, \underbrace{m_0, \ldots, m_{k-1}}_{\text{Klartext}}) \end{aligned}$$

$g(x)$ heißt *Erzeugerpolynom* der Codierung.
Beachte:
Polynomcodierung in dieser Form liefert keinen Blockcode; es können Wörter beliebiger Länge codiert werden; die Länge des Codewortes ist dann um ℓ Stellen länger.

Wird $c(x)$ gesendet und $d(x)$ empfangen, so bildet der Empfänger $d(x) \mod g(x)$. Es ist $c(x) \mod g(x) = 0$.
Ist $d(x) \mod g(x) \neq 0$, so weiß man, dass Fehler aufgetreten sind. Ist $d(x) \mod g(x) = 0$, so nimmt man an, dass kein Fehler aufgetreten ist. $m(x)$ kann dann sofort an den letzten k Stellen (Koeffizienten von $x^\ell, \ldots, x^{\ell+k-1}$) abgelesen werden.

8.3 Bemerkung.
Polynomcodierung ist sinnvoll, wen es nur auf Fehlererkennung ankommt; z.B. wird sie verwendet um die Nachrichten eines Pakets im Ethernet LAN gegen Übertragungsfehler zu schützen. Die ersten ℓ Stellen ($-x^\ell m(x) \mod g(x)$) heißen dann *Ethernet-Trailer* oder *Frame Checking Sequence* (FCS).
Generatorpolynome sind dabei standardisiert:

CRC-CCITT-Polynom : $g(x) = x^{16} + x^{12} + x^5 + 1$
CRC-16-Polynom : $g(x) = x^{16} + x^{15} + x^2 + 1$

CCITT (Comité Consultativ International de Téléphonique et Télégraphique), Vorläuferorganisation der ITU, einem internationalen Gremium zur Vereinheitlichung und Normierung von Telekommunikationsstandards.

CRC=Cyclic Redundancy Code (zu dieser Bezeichnung später mehr)

8.4 Beispiel.
$g(x) = x^{16} + x^{15} + x^2 + 1$, $K = \mathbb{Z}_2$.
Nachricht $m = 10110101 \leftrightarrow m(x) = 1 + x^2 + x^3 + x^5 + x^7$ (normalerweise ist Nachricht länger).
$x^{16}m(x) = x^{16} + x^{18} + x^{19} + x^{21} + x^{23}$

$$x^{23} + x^{21} + x^{19} + x^{18} + x^{16} : x^{16} + x^{15} + x^2 + 1 = x^7 + x^6 + x^3 + 1$$
$$\underline{x^{23} + x^{22} + x^9 + x^7}$$
$$x^{22} + x^{21} + x^{19} + x^{18} + x^{16} + x^9 + x^7$$
$$\underline{x^{22} + x^{21} + x^8 + x^6}$$
$$x^{19} + x^{18} + {}^{16} + {}^{9} + x^8 + x^7 + x^6$$
$$\underline{x^{19} + x^{18} + x^5 + x^3}$$
$$x^{16} + x^9 + x^8 + x^7 + x^6 + x^5 + x^3$$
$$\underline{x^{16} + x^{15} + x^2 + 1}$$
$$x^{15} + x^9 + x^8 + x^7 + x^6 + x^5 + x^3 + x^2 + 1$$

$$x^{16}m(x) \mod g(x) = x^{15} + x^9 + x^8 + x^7 + x^6 + x^5 + x^3 + x^2 + 1$$

$$\begin{aligned} c(x) &= 1 + x^2 + x^3 + x^5 + x^6 + x^7 + x^8 + x^9 + x^{15} + x^{16} + x^{18} + x^{21} + x^{23} \\ &\leftrightarrow (\underbrace{\underline{1011011111000001}}_{\text{Ethernet-Trailer}} \mid \underbrace{\underline{10110101}}_{\text{Klartext}}) \end{aligned}$$

8.5. Polynomcodierung mit linearen Schieberegistern
Sei $g(x) = a_0 + \ldots + a_{\ell-1}x^{\ell-1} + x^\ell$,
$m(x) = m_0 + m_1 x + \ldots + m_{k-1}x^{k-1}$.
Wir haben $x^\ell \cdot m(x) \mod g(x) =: R(x^\ell \cdot m(x)) = b_0 + b_1 + \ldots + b_{\ell-1}x^{\ell-1}$ zu berechnen.
Wir benutzen das Hornerschema:
$x^\ell m(x) = m_0 x^\ell + x(m_1 x^\ell + x(m_2 x^\ell + \ldots + x(m_{k-1}x^\ell)\ldots))$
Dann:
$$R(x^\ell m(x)) = \underbrace{R(m_0 x^\ell + x\,\underbrace{R(m_1 x^\ell + xR(\ldots + x\,\underbrace{R(m_{k-1}x^\ell)}_{S^{(1)}(x)}\ldots))}_{S^{(k-1)}(x)})}_{S^{(k)}(x)}$$

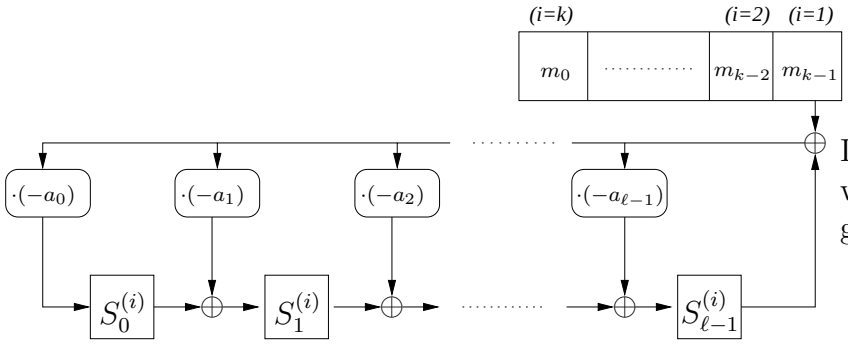
$S^{(0)}(x) := 0$
$S^{(i)}(x) = R(m_{k-i}x^\ell + xS^{(i-1)}(x)) \quad (*)$
Dann: $S^{(k)}(x) = x^\ell m(x) \mod g(x)$.

Alle Polynome $S^{(i)}$ haben Grad $\leq \ell - 1$, $S^{(i)}(x) = \sum_{j=0}^{\ell-1} S_j^{(i)} x^j$.
Reduktion mod $g(x)$ bedeutet, x^ℓ durch $-\sum_{i=0}^{\ell-1} a_i \cdot x^i$ zu ersetzen. Macht man das, so folgt aus $(*)$:

$$\begin{aligned} S_j^{(i)} &= S_{j-1}^{(i-1)} - a_j(m_{k-i} + S_{\ell-1}^{(i-1)}), \quad j = 0, \ldots, \ell-1 \quad (**) \\ & \quad (S_{-1}^{(i)} = 0 \quad \text{gesetzt}) \end{aligned}$$

Es ist dann $S_j^{(k)} = b_j, \quad j = 0, \ldots, \ell - 1$.

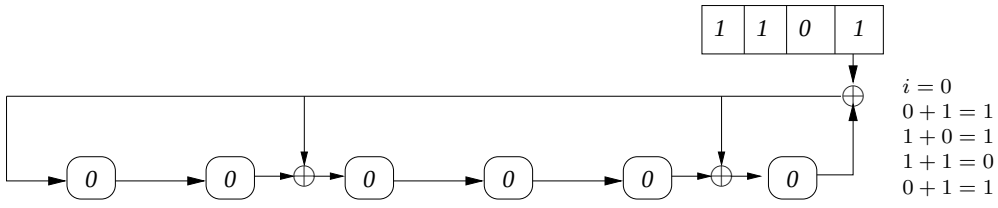
Die Berechnung von $(**)$ wird durch das folgende lineare Schieberegister realisiert:



Im i-ten Takt wird m_{k-i} eingegeben.

Am Anfang werden alle Register mit 0 belegt. Nach dem k-ten Takt sind die Register mit $b_0, \ldots, b_{\ell-1}$ belegt.

8.6 Beispiel.
$g(x) = x^6 + x^5 + x^2 + 1$, $m(x) = x^3 + x + 1 \quad (K = \mathbb{Z}_2) \quad (k = 4, \ell = 6)$



$i = 1:$	1	0	1	0	0	1
$i = 2:$	1	1	1	1	0	1
$i = 3:$	0	1	1	1	1	0
$i = 4:$	1	0	0	1	1	0

$x^6 \cdot m(x) \mod g(x) = 1 + x^3 + x^4$.

Man kann Polynomcodierung auch zur Erzeugung von Blockcodes verwenden (und so wird sie auch in der Regel eingesetzt).
Blockcode der Länge n über K.
Identifiziere K^n mit $K[x]_n$.
Wähle Erzeugerpolynom $g(x) = a_0 + a_1 x + \ldots + a_{n-k-1}x^{n-k-1} + x^{n-k}$ vom Grad $\ell = n - k$. Wir setzen $a_{n-k} = 1$.
Codiert werden Elemente des K^k, identifiziert mit $K[x]_k$.
$m(x) = m_0 + m_1 + \ldots + m_{k_1}x^{k-1}$

$$\rightarrow c(x) = \underbrace{-b_0 - b_1 x - \ldots - b_{n-k-1}x^{n-k-1}}_{=-x^{n-k}m(x) \mod g(x)} + m_0 x^{n-k} + \ldots + m_{k-1}x^{n-1}$$

(In der Regel ist n deutlich größer als $n - k$; z.B.: CRC-Polynom hat Grad $16 = n - k$; es wird üblicherweise für Blockcodes der Länge $n = 2^{15} - 1$ verwendet, das heißt $k = 2^{15} - 17$. Dazu später noch mehr.)
Mit diesen Bezeichnungen gilt dann:

8.7 Satz.

(a) Die Menge $\mathcal{C}$ aller Codewörter in $K[x]_n$ ist gerade

$$g(x) \cdot K[x]_k.$$

(b) $\mathcal{C}$ ist ein linearer $[n, k]$-Code über K;

$$\begin{pmatrix} a_0 & \ldots & \ldots & a_{n-k} & 0 & \ldots & 0 \\ 0 & \ddots & & & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & & & \ddots & 0 \\ 0 & \ldots & 0 & a_0 & \ldots & \ldots & a_{n-k} \end{pmatrix}$$

ist Erzeugermatrix von $\mathcal{C}$.

(c) Ist $x^i = g(x)q_i(x) + r_i(x)$ ($Grad(r_i) < Grad(g) = n - k$) für $i = n - k, \ldots, n - 1$ und

$$r_i(x) = r_0^{(i)} + r_1^{(i)}x + \ldots + r_{n-k-1}^{(i)}x^{n-k-1},$$

so ist

$$\begin{pmatrix} -r_0^{(n-k)} & \ldots & -r_{n-k-1}^{(n-k)} & 1 & & 0 \\ \vdots & & \vdots & & \ddots & \\ -r_0^{(n-1)} & \ldots & -r_{n-k-1}^{(n-1)} & 0 & & 1 \end{pmatrix}$$

Erzeugermatrix von $\mathcal{C}$, (Entspricht einer Standardform mit Einheitsmatrix am Ende.)

(In (b) und (c) beziehen sich die Erzeugermatrizen auf die Schreibweise der Elemente von $\mathcal{C}$ als n-Tupel.)

Beweis.

(a) Klar: $\mathcal{C} \subseteq g(x) \cdot K[x]_k$, denn ist
$\underbrace{x^{n-k}m(x)}_{\text{Grad } \leq n-1} = \underbrace{g(x)}_{\text{Grad } n-k} \cdot q(x) + r(x)$, so $\text{Grad}(q(x)) \leq k-1$
und $c(x) = g(x) \cdot q(x)$.

Umgekehrt: Ist $g(x) \cdot t(x) \in g(x) \cdot K[x]_k$, das heißt $\text{Grad}(t(x)) < k$,
$g(x)t(x) = c_0 + \ldots + c_{n-k-1}x^{n-k-1} + m_0x^{n-k} + \ldots + m_{k-1}x^{n-1}$,
so setze $m(x) = m_0 + \ldots + m_{k-1}x^{k-1} \in K[x]_k$.
Dann $g(x)t(x) = \underbrace{c_0 + \ldots + c_{n-k-1}x^{n-k-1}}_{=-x^{n-k}m(x) \mod g(x)} + x^{n-k}m(x) \in \mathcal{C}$

(b) Folgt aus a); Basis $g(x) \cdot 1, \ldots, g(x) \cdot x^{k-1}$ liefert Erzeugermatrix.

(c)

$$\begin{aligned} x^{n-k}m(x) &= m_0x^{n-k} + \ldots + m_{k-1}x^{n-1} \\ &= m_0r_{n-k}(x) + \ldots + m_{k-1}r_{n-1}(x) \\ &\quad + g(x)(m_0q_{n-k}(x) + \ldots + m_{k-1}q_{n-1}(x)) \end{aligned}$$

$\rightarrow m_0r_{n-k}(x) + \ldots + m_{k-1}r_{n-1}(x) = x^{n-k}m(x) \mod g(x)$

Basis von $\mathcal{C}$ ist

$$\begin{aligned} x^{n-k} \cdot 1 - (x^{n-k} \cdot 1 \mod g(x)) &= x^{n-k} - r_{n-k}(x) \\ x^{n-k} \cdot x - (x^{n-k} \cdot x \mod g(x)) &= x^{n-k+1} - r_{n-k+1}(x) \\ &\vdots \\ x^{n-k} \cdot x^{k-1} - (x^{n-k} \cdot x^{k-1} \mod g(x)) &= x^{n-1} - r_{n-1}(x) \end{aligned}$$

Daraus folgt die Behauptung.

□

Bemerkung.
Die Codierung $(m_0, \ldots, m_{k-1}) \rightarrow (m_0, \ldots, m_{k-1}) \cdot G$ mit der Erzeugermatrix aus 8.7(*c*) entspricht gerade der Polynomcodierung entsprechend 8.2.

Polynomcodierung wird vor allem für sogennante zyklische Codes verwendet.

8.8 Definition.
Sei $\mathcal{C}$ ein $[n,k]$-Code über K. $\mathcal{C}$ heißt *zyklisch*, falls gilt:
Ist $(c_1,\ldots,c_n)\in\mathcal{C}$, so auch $\sigma(c_1,\ldots,c_n):=(c_n,c_1,c_{n-1})\in\mathcal{C}$.
Dann auch $\sigma^i(c_1,\ldots,c_n)=(c_{n-i+1},\ldots,c_n,c_1,\ldots,c_{n-i})\in\mathcal{C}\quad\forall i=1,\ldots,n$.

8.9 Bemerkung.
Ist $\mathcal{C}$ ein zyklischer Code, $y^{(1)},\ldots,y^{(\ell)}\in\mathcal{C}^\perp$ orthogonal bezüglich Position 1,
$y^{(j)}=(y_1^{(j)},\ldots,y_n^{(j)})$ (vgl. 7.16).
Dann $\sigma^i(y^{(j)})=(y_{n-i+1}^{(j)},\ldots,y_n^{(j)},y_1^{(j)},\ldots,y_{n-i}^{(j)})\in\mathcal{C}^\perp$
[$u_1,\ldots,u_k$ Basis von $\mathcal{C}$, so auch $\sigma^i(u_1),\ldots,\sigma^i(u_k)$].
Außerdem: $\sigma^i(y^{(j)})$, $j=1,\ldots,\ell$, sind orthogonal bezüglich Position $i+1$.
Majority-Logic-Decodierung ist für zyklische Codes an allen Positionen möglich, wenn sie an einer Position möglich ist.

8.10 Beispiel.
Sei $\mathcal{C}$ der binäre Hamming $[7,4]$-Code aus Beispiel 4.5 bzw. 5.8 bzw. 5.12:

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

$(1110001)\in\mathcal{C}$
$\sigma(1110001)=(1111000)\notin\mathcal{C}$, da

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

Wir werden aber zeigen, dass es einen äquivalenten Hamming-$[7,4]$-Code gibt (Spaltenvertauschungen in H - und entsprechend in G), der zyklisch ist. Wir zeigen dies allgemeiner für alle binären Hamming-Codes.

8.11 Satz.
Für jedes $\ell\geq 2$ gibt es einen zyklischen $[2^\ell-1,2^\ell-1-\ell]$-Hamming-Code über $\mathbb{Z}_2$.
(Ohne Beweis: das gilt auch für die nicht-binären Hamming-Codes.)

Wir benötigen dazu:

8.12.
Sei K ein endlicher Körper, mit $|K|=q=p^m$, p Primzahl (5.2).

(a) Für alle $k \in K$ gilt: $\underbrace{k + \ldots + k}_{p} = 0$.

(b) Die multiplikative Gruppe $K^* = K \setminus \{0\}$ ist zyklisch, das heißt es existiert $\alpha \in K^*$ mit $K^* = \{1 = \alpha^0, \alpha^1, \ldots, \alpha^{q-2}\}$ $(\alpha^{q-1} = 1)$.

(Beweis z.B. Willems, 2.2.1 und 2.2.6)

Beweis von 8.11.
Es gibt nach 5.2 einen endlichen Körper K mit $|K| = 2^p$.
Dieser Körper enthält $\{0, 1\} = \mathbb{Z}_2$ als Teilkörper (da $1 + 1 = 0$ in K nach 8.12 a)).
Daher kann man K als ℓ-dimensionalen Vektorraum über $\mathbb{Z}_2$ betrachten.
Sei $k_1, \ldots, k_\ell$ eine Basis,
das heißt $K = \{\sum \epsilon_i \cdot k_i : \epsilon_i = 0, 1\} \leftrightarrow \{(\epsilon_1, \ldots, \epsilon_\ell)^t : \epsilon_i = 0, 1\}$.
$K^* = \{1, \alpha, \ldots, \alpha^{q-2}\}$ für ein geeignetes $\alpha \in K^*$ nach 8.12 b), $q = 2^\ell$.
Schreibe die $1 = \alpha^0, \alpha, \ldots, \alpha^{q-2}$ als $0 - 1-$Vektoren der Länge ℓ. Das liefert alle Elemente $\neq 0$ von K, das heißt alle Vektoren $\neq 0$ des $\mathbb{Z}_2$-Vektorraums K.
Bilde $H = (\alpha^0, \alpha^1, \ldots, \alpha^{q-2})$, α^i als Spaltenvektor der Länge ℓ aufgefasst.
Dann ist H Kontrollmatrix für $[2^\ell - 1, 2^\ell - 1 - \ell]$-Hamming-Code $\mathcal{H}_\ell$.

$$x = (x_1, \ldots, x_{q-1}) \in \mathcal{H}_\ell \iff Hx^t = 0 \iff \sum_{i=1}^{q-1} x_i \cdot \alpha^{i-1} = 0$$

$$x_{q-1}\alpha^0 + \sum_{i=1}^{q-2} x_i \alpha^i = \left(\sum_{i=1}^{q-1} x_i \alpha^{i-1} \right) \alpha = 0 \quad (\text{denn } \alpha^{q-1} = \alpha^0 = 1)$$

Also: $(x_{q-1}, x_1, \ldots, x_{q-1}) \in \mathcal{H}_\ell$, $\mathcal{H}_\ell$ ist zyklisch.

[Für den Fall $\ell = 3$ führt das z.B. auf folgende Kontrollmatrix für einen zyklischen Hamming $[7, 4]$-Code $\mathcal{H}_3$:

$$\tilde{H} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} \text{ und Erzeugermatrix } \tilde{G} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

$\sigma(1010001) = \ (1101000) \in \mathcal{H}_3$
$\sigma(1110010) = \ (0111001) = (1010001) + (1101000) \in \mathcal{H}_3$
$\sigma(0110100) = \ (0011010) = (1110010) + (1101000) \in \mathcal{H}_3$
$\sigma(1101000) = \ (0110100) \in \mathcal{H}_3$]

□

8.13 Exkurs (Polynomringe, II)**.**

(1) Sei R ein beliebiger kommutativer Ring (mit Eins).
$I \subseteq R$ heißt **Ideal**, falls gilt:
a) I ist bezüglich $+$ eine Untergruppe von R;
b) $ri \in I$ für alle $r \in R, i \in I$, d.h. $rI \subseteq I$ für alle $r \in R$.

(2) Bedeutung von Idealen:
Die Nebenklassen bezüglich Addition von I in R, also die Mengen $s+I$, $s \in R$, bilden bezüglich

$$(s_1 + I) + (s_2 + I) := (s_1 + s_2) + I \text{ und}$$

$$(s_1 + I) \cdot (s_2 + I) := (s_1 \cdot s_2) + I$$

einen Ring R/I, den **Faktorring** von R nach I.
(Wichtig: Da I ein Ideal ist, hängt die Definition der Verknüpfungen $+$ und $\cdot$ auf den Nebenklassen *nicht* von der Wahl der Vertreter ab.)

(3) Ist $a \in R$, so ist $aR = \{ar : r \in R\}$ ein Ideal (**Hauptideal**).

(4) In $K[x]$ ist jedes Ideal ein Hauptideal, d.h. von der Form $f(x)K[x]$ für ein $f(x) \in K[x]$.
Dies liegt im Wesentlichen an der Division mit Rest.
(Ein entsprechender Satz gilt für den Ring $\mathbb{Z}$.)

(5) $f_1(x)K[x] \subseteq f_2(x)K[x] \iff f_2(x) \mid f_1(x)$ ($f_2(x)$ teilt $f_1(x)$).
Ist $f(x)$ irreduzibel, so ist $K[x]/f(x)K[x]$ ein Körper.

(6) Die Ideale in $K[x]/f(x)K[x]$ sind genau die $g(x)K[x]/f(x)K[x]$ mit $g(x) \mid f(x)$.

(7) Sei der Grad von $f(x)$ gleich n. Dann bilden die Polynome vom Grad $< n$ ein Vertretersystem der Nebenklassen von $f(x)K[x]$ in $K[x]$:
$K[x]/f(x)K[x] = \{ \underbrace{g(x) + f(x)K[x]}_{\text{paarweise verschieden}} : \text{Grad}(g(x)) < n\}$
Addition und Multiplikation auf $K[x]/f(x)K[x]$ lassen sich auf $K[x]_n$ übertragen:
Addition $+$ = übliche Polynomaddition
Multiplikation $\odot$ = Multiplikation in $K[x]$ und dann Reduktion mod $f(x)$.
Dann sind $K[x]_n$ und $K[x]/f(x)K[x]$ isomorphe Ringe.
(Beachte: Die Multiplikation auf $K[x]_n$ hängt von $f(x)$ ab.)

(8) Beispiele:

a) $K = \mathbb{Z}_2$, $f(x) = x^3 + x^2 + 1$.
$K[x]/f(x)K[x] = \{\, g(x) + f(x)K[x] : \ \mathrm{Grad}(g(x)) < 3\}$
$K[x]_3 = \{0,\ 1,\ x,\ x+1,\ x^2,\ x^2+1,\ x^2+x,\ x^2+x+1\}$
Multiplikation: Was ist $(x^2+1) \odot (x^2+x+1)$?
$(x^2+1) \cdot (x^2+x+1) = x^4 + x^3 + x + 1$ in $K[x]$.
Reduktion mod $f(x)$:
$x^4 + x^3 + x + 1 = x \cdot (x^3 + x^2 + 1) + 1$, also
$(x^2+1) \odot (x^2+x+1) = 1$.
$x^2 + x + 1$ ist also das Inverse bezüglich der Multiplikation $\odot$ in $K[x]_3$ von $x^2 + 1$.
Da $f(x)$ ein irreduzibles Polynom in $K[x]$ ist, ist $K[x]_3$ bezüglich der angegebenen Addition und Multiplikation ein Körper, d.h. jedes Element $\neq 0$ hat bezüglich der Multiplikation $\odot$ ein Inverses.

b) $K = \mathbb{Z}_2$, $f(x) = x^3 + 1$.
Jetzt ist $f(x)$ nicht irreduzibel, denn $f(x) = (x+1)(x^2+x+1)$ (Zerlegung in irreduzible Faktoren in $K[x]$).
Nach (6) sind die Ideale von $K[x]/f(x)K[x]$:

$$\begin{aligned}
1K[x]/f(x)K[x] &= K[x]/f(x)K[x] \\
(x+1)K[x]/f(x)K[x] &= \{\, 0 + f(x)K[x],\ (x+1) + f(x)K[x], \\
&\qquad (x^2+x) + f(x)K[x],\ (x^2+1) + f(x)K[x] \,\} \\
(x^2+x+1)K[x]/f(x)K[x] &= \{\, 0 + f(x)K[x],\ (x^2+x+1) + f(x)K[x] \,\} \\
f(x)K[x]/f(x)K[x] &= \{\, 0 + f(x)K[x] \,\}
\end{aligned}$$

Da $f(x)$ jetzt von dem in a) verschieden ist, ist auch die Multiplikation $\odot$ in $K[x]_3$ von der in a) verschieden.
Was ist jetzt $(x^2+1) \odot (x^2+x+1)$?
$(x^2+1) \cdot (x^2+x+1) = x^4 + x^3 + x + 1$ in $K[x]$.
Reduktion mod $f(x)$:
$x^4 + x^3 + x + 1 = (x+1) \cdot (x^3+1)$, d.h.
$(x^2+1) \odot (x^2+x+1) = 0$.
Daher haben auch x^2+1 und x^2+x+1 (ebenso wie $x+1$ und x^2+x) keine Inverse bezüglich dieser Multiplikation $\odot$ in $K[x]_3$. $K[x]_3$ ist jetzt nur ein kommutativer Ring mit Eins, aber kein Körper.

8.14 Bemerkung.
Sei K ein endlicher Körper.
Identifiziere K^n mit $K[x]_n$ $((a_0, \ldots, a_n) \leftrightarrow \sum_{i=0}^{n-1} a_i \cdot x^i)$.
Mache entsprechend 8.13 $(K[x]_n, +, \odot)$ zu einem Ring, indem man Multiplikation $\mod x^n - 1$ rechnet, das heißt $(K[x]_n, +, \odot) \widetilde{=} K[x]/(x^n-1)K[x]$.
Dann entspricht dem Shift $\sigma(a_0, \ldots, a_{n-1}) = (a_{n-1}, a_0, \ldots, a_{n-2})$ in K^n gerade die Multiplikation mit x in $(K[x]_n, +, \odot)$.

Beweis:

$$\begin{aligned} x \odot \left(\sum_{i=0}^{n-1} a_i x^i \right) &= ? \\ x \cdot \sum_{i=0}^{n-1} a_i x^i &= \sum_{i=0}^{n-1} a_i x^{i+1} = \sum_{i=0}^{n-2} a_i x^{i+1} + a_{n-1} x^n \\ x^n &\equiv 1 \mod (x^n - 1) \\ \text{Also: } x \odot \left(\sum_{i=0}^{n-1} a_i x^i \right) &= a_{n-1} + \sum_{i=0}^{n-2} a_i x^{i+1} \leftrightarrow (a_{n-1}, a_0, \ldots, a_{n-2}). \end{aligned}$$

□

Charakterisierung zyklischer Codes.

8.15 Satz.
Sei K ein endlicher Körper, $\mathcal{C}$ ein linearer $[n,k]$-Code, das heißt $\mathcal{C}$ ist ein k-dimensionaler Unterraum von K^n.
Identifiziere K^n mit $(K[x]_n, +, \odot) \cong K[x]/(x^n-1)K[x]$ wie in 8.14.
Sei $\tilde{\mathcal{C}} \subseteq K[x]_n$ der entsprechende Unterraum in $K[x]_n$.

Dann: $\mathcal{C}$ ist zyklisch $\iff$ $\tilde{\mathcal{C}}$ ist Ideal in $K[x]_n$.

Beweis:
$\Leftarrow$:

$(c_0, \ldots, c_{n-1}) \in \mathcal{C} \Rightarrow \sum_{i=0}^{n-1} c_i x^i \in \tilde{\mathcal{C}}$

$\Rightarrow c_{n-1} + \sum_{i=0}^{n-2} c_i x^{i+1} \overset{8.14}{=} x \odot \left(\sum_{i=0}^{n-1} c_i x^i \right) \in \tilde{\mathcal{C}}$, da $\tilde{\mathcal{C}}$ Ideal ist.

$\Rightarrow$: wie in $\Leftarrow$ sieht man:

$\sum_{i=0}^{n-1} c_i x^i \in \tilde{\mathcal{C}} \Rightarrow x \odot \left(\sum_{i=0}^{n-1} c_i x^i \right) \in \tilde{C}$

$\overset{\text{iterieren}}{\Rightarrow} x^j \odot \left(\sum_{i=0}^{n-1} c_i x^i \right) \in \tilde{C}$ für alle $j = 0, 1, \ldots, n-1$

$\overset{\tilde{c} \text{ ist UR}}{\Rightarrow} \left(\sum_{j=0}^{n-1} b_j x^j \right) \odot \left(\sum_{i=0}^{n-1} c_i x^i \right) \overset{\text{Distributiv-gesetz}}{=} \sum_{j=0}^{n-1} b_j \left(x^j \odot \sum_{i=0}^{n-1} c_i x^i \right) \in \tilde{\mathcal{C}},$

$\tilde{\mathcal{C}}$ ist Ideal. □

8.16 Satz.

(a) Sei $\mathcal{C}$ ein zyklischer $[n,k]$-Code über einem endlichen Körper K, identifiziert mit $\tilde{\mathcal{C}} \subseteq (K[x]_n, +, \odot) \widetilde{=} K[x]/(x^n-1)K[x]$ entsprechend 8.15. Dann existiert ein eindeutig bestimmtes Polynom $g(x)$ mit höchstem Koeffizienten 1 vom Grad $n-k$ in $K[x]$ und $g(x) \mid (x^n-1)$, so dass

$$\tilde{\mathcal{C}} = \{ \underbrace{m(x) \cdot g(x)}_{\substack{\text{normale Polynom-} \\ \text{Multiplikation}}} : \mathit{Grad}(m(x)) < k\} = g(x) \cdot K[x]_k$$

$g(x)$ ist also Erzeugerpolynom von $\tilde{\mathcal{C}}$ (bzw. von $\mathcal{C}$) [vgl, 8.7].

(b) Ist in der Situation von (a) $x^n - 1 = g(x) \cdot h(x)$, also $\mathit{Grad}(h) = k$, so ist $\tilde{\mathcal{C}} = \{f(x) \in K[x]_n : f(x) \odot h(x) = 0\}$.
$h(x)$ heißt Kontrollpolynom *von $\tilde{\mathcal{C}}$ (bzw. von $\mathcal{C}$).*

(c) Jedes Polynom $g(x)$ (mit höchstem Koeffizienten 1) vom $\mathit{Grad}(n-k)$ und $g(x) \mid x^n - 1$ liefert durch

$$\tilde{\mathcal{C}} = \{m(x) \cdot g(x) : \mathit{Grad}(m(x)) < k\}$$

einen zyklischen $[n,k]$-Code $\tilde{\mathcal{C}}$.

Beweis.

(a) Nach 8.15 ist $\tilde{\mathcal{C}}$ ein Ideal in $(K[x]_n, +, \odot)$, $\odot$ bezüglich $x^n - 1$.
Es ist $(K[x]_n, +, \odot) \widetilde{=} K[x]/(x^n-1)K[x]$. Ideale von $K[x]/(x^n-1)K[x]$ sind nach 8.13 (6) von der Form $g(x)K[x]/(x^n-1)K[x]$, $g(x) \mid x^n - 1$. Dabei ist $g(x)$ bis auf skalare Vielfache eindeutig festgelegt (folgt aus 8.13 (5)).
Fordert man, dass der höchste Koeffizient von $g(x)$ gleich 1 ist, so ist $g(x)$ eindeutig bestimmt.
Dem Ideal $g(x)K[x]/(x^n-1)K[x]$ entspricht in $(K[x]_n, +, \odot)$ das Ideal $g(x) \odot K[x]_n = \{g(x) \odot f(x) : f(x) \in K[x]_n\}$.
Ist $\mathrm{Grad}(f) < k$, so $g(x) \odot f(x) = g(x) \cdot f(x)$.
Da $|\{g(x) \cdot m(x) : \mathrm{Grad}(m(x)) < k\}| = |K|^k = |\tilde{\mathcal{C}}|$, folgt die letzte Behauptung unter (a).

(b) Es ist $g(x) \odot h(x) = 0$ in $K[x]_n$, also $c(x) \odot h(x) = 0$ für alle $c(x) \in \tilde{\mathcal{C}}$ nach (a).
Ist umgekehrt $f(x) \odot h(x) = 0$ $(f(x) \in K[x]_n)$, so $g(x) \cdot h(x) = x^n - 1 \mid f(x) \cdot h(x)$, also $g(x) \mid f(x) \Rightarrow f(x) \in g(x) \cdot K[x]_n = \tilde{\mathcal{C}}$ nach (a).

(c) Wie in (a) sieht man, dass $\tilde{\mathcal{C}}$ Ideal in $(K[x]_n, +, \odot)$ ist.
Behauptung folgt aus 8.15. $|\tilde{\mathcal{C}}| = |K|^k$, also $\dim(\tilde{\mathcal{C}}) = k$.

8.17. Folgerung
Ist $\mathcal{C}$ ein zyklischer $[n,k]$-Code,
$g(x) = \sum_{i=0}^{n-k} g_i x^i$, $(g_{n-k} = 1)$, das Erzeugerpolynom,
$h(x) = \sum_{i=0}^{k} h_i x^i$, $(h_k = 1)$, so ist die $k \times n$-Matrix

$$G = \begin{pmatrix} g_0 & \dots & \dots & g_{n-k} & 0 & \dots & 0 \\ 0 & \ddots & & & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & & & \ddots & 0 \\ 0 & \dots & 0 & g_0 & \dots & \dots & g_{n-k} \end{pmatrix}$$

Erzeugermatrix von $\mathcal{C}$ und die $(n-k) \times n$-Matrix

$$H = \begin{pmatrix} h_k & \dots & \dots & h_0 & 0 & \dots & 0 \\ 0 & \ddots & & & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & & & \ddots & 0 \\ 0 & \dots & 0 & h_k & \dots & \dots & h_0 \end{pmatrix}$$

Kontrollmatrix von $\mathcal{C}$.

Beweis:
Aussage über G folgt aus 8.7 b) und 8.16 a).
Es ist $x^n - 1 = g \cdot h = \sum_{i=0}^{n} \left(\sum_{j=0}^{i} g_j h_{i-j} \right) x^i = \sum_{i=0}^{n} \left(\sum_{j=0}^{n-k} g_j h_{i-j} \right) x^i$
(alle nicht definierten h_j, g_j gleich 0 zu setzen).
Koeffizientenvergleich: $\sum_{j=0}^{n-k} g_j h_{i-j} = 0$ für alle $i = 1, \dots, n-1$. Also $GH^t = 0$,
das heißt $HG^t = 0$.
$g_0 h_0 = -1 \Rightarrow rg(H) = n - k \Rightarrow H$ ist Kontrollmatrix. □

8.18 Folgerung.
Es gibt genauso viele zyklische Codes der Länge n über einem endlichen Körper K wie es Teiler von $x^n - 1$ in $K[x]$ gibt.

8.19 Beispiele.

(a) $n = 4$, $K = \mathbb{Z}_2$.
Zyklische Codes der Länge 4 über K.

$$x^4 - 1 = (x-1)^4 \text{ in } K[x]$$

Teiler:

1	$\mathcal{C} = \mathbb{Z}_2^4$
$x-1$	Erzeugermatrix $\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$, $\dim(\mathcal{C}) = 3$
$(x-1)^2 = x^2 - 1$	Erzeugermatrix $\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$ $\dim(\mathcal{C}) = 2$
$(x-1)^3 = x^3 + x^2 + x + 1$	Erzeugermatrix $\begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix}$, $\dim(\mathcal{C}) = 1$
$(x-1)^4$	$\mathcal{C} = \{0\}$

(b) Für das CRC-CCITT Polynom $g(x) = x^{16} + x^{12} + x^5 + 1$ und das CRC-16-Polynom $\tilde{g}(x) = x^{16} + x^{15} + x^2 + 1$ (siehe 8.3) gilt, dass sie $x^{2^{15}-1} - 1$ teilen. Die entsprechenden Blockcodes der Länge $2^{15} - 1$ (vgl. 8.6) sind also zyklisch. Diese Codes haben darüberhinaus besonders gute Fehlererkennungseigenschaften (vgl. Friedrichs, Kanalcodierung, Kapitel 5).

8.20 Satz.
Ist $\mathcal{C}$ ein zyklischer Code, so auch $\mathcal{C}^\perp$.
Ist $h(x) = h_0 + h_1 x + \ldots + h_k x^k$ ($h_k = 1$) das Kontrollpolynom von $\mathcal{C}$ (beachte $h_0 \neq 0$), so ist

$$h^*(x) = h_0^{-1}(h_k + h_{k-1}x + \ldots + h_0 x^k)$$

das Erzeugerpolynom von $\mathcal{C}^\perp$.
Entsprechend erhält man das Kontrollpolynom von $\mathcal{C}^\perp$ aus dem Erzeugerpolynom von $\mathcal{C}$.

Beweis:
$h^*(x)$ hat höchsten Koeffizient 1, $h^*(x) = h_0^{-1} x^k h(\frac{1}{x})$.
$g(x)h(x) = x^n - 1 \Rightarrow g(\frac{1}{x})h(\frac{1}{x}) = \frac{1}{x^n} - 1$.
$\underbrace{x^{n-k} g(\frac{1}{x})}_{\text{Polynom}} \underbrace{x^k h(\frac{1}{x})}_{\text{Polynom}} = 1 - x^n = -(x^n - 1)$, also $h^*(x) \mid x^n - 1$
Sei $\mathcal{C}^* = \{h^*(x) \cdot m(x) : \text{Grad}(m) < n - k\}$, zyklischer $[n, n-k]$-Code nach 8.16 c) mit Erzeugerpolynom $h^*(x)$.
Eine Erzeugermatrix von $\mathcal{C}^*$ ist nach 8.17

$$G^* = h_0^{-1} \begin{pmatrix} h_k & \ldots & \ldots & h_0 & 0 & \ldots & 0 \\ 0 & \ddots & & & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & & & \ddots & 0 \\ 0 & \ldots & 0 & h_k & \ldots & \ldots & h_0 \end{pmatrix}$$

Dies ist nach 8.17 eine Kontrollmatrix von $\mathcal{C}$, also eine Erzeugermatrix von $\mathcal{C}^\perp$. Daher $\mathcal{C}^* = \mathcal{C}^\perp$. □

Eine wichtige Klasse zyklischer Codes werden wir im nächsten Kapitel kennenlernen.

Zyklische Codes sind häufig gut geeignet nicht nur zufällige Fehler gut zu erkennen (und zu korrigieren), sondern auch Fehlerbündel (mehrere nahe beieinander liegende Bits gestört), ein typischer Fall bei Speicherung auf CDs oder bei Übertragungen.

8.21 Definition.

(a) Ein Vektor $(0, \ldots, 0, \overbrace{a, * \ldots *, \tilde{a}}^{b \text{ Stellen}}, 0, \ldots, 0) \in K^n$ (K endlicher Körper) mit $a, \tilde{a} \neq 0$ heißt *Bündel* der Länge b ('burst').

(b) Wird $c \in \mathcal{C}$ gesendet, $v \in K^n$ empfangen und ist $f = v - c$ (der Fehlervektor) ein Bündel der Länge b, so heißt f *Fehlerbündel* der Länge b.

8.22 Satz.
Sei $\mathcal{C}$ ein zyklischer $[n,k]$-Code. Dann enthält $\mathcal{C}$ keine Bündel der Länge $\leq n-k$.
Daher entdeckt ein zyklischer $[n,k]$-Code Fehlerbündel der Länge $\leq n-k$.

Beweis.
Angenommen $(0, \ldots, 0, a_i, \ldots, a_{i+b-1}, 0, \ldots, 0) \in \mathcal{C}$, $a_i, a_{i+b-1} \neq 0$, $b \leq n-k$.
Dann auch $c = (a_i, \ldots, a_{i+b-1}, \underbrace{0, \ldots, 0}_{\geq k \text{ Stellen}}) \in \mathcal{C}$.
Die b-te Zeile der Kontrollmatrix H aus 8.17 ist

$$h = (\underbrace{0, \ldots, 0}_{b-1 \text{ Stellen}}, \underbrace{h_k}_{=1}, \ldots, h_0, 0, \ldots, 0) \qquad \text{(Beachte: } (b-1) + (k-1) \leq n\text{)}$$

.
$hc^t = h_k \cdot a_{i+b-1} \neq 0$, also $Hc^t \neq 0$. Widerspruch.
Ist $v = c + f$, f Fehlerbündel der Länge $\leq n-k$, so

$Hv^t = Hc^t + Hf^t \overset{c \in \mathcal{C}}{=} Hf^t \neq 0$, da $f \notin \mathcal{C}$. □

8.23 Bemerkung.
Für einen linearen $[n,k,d]$-Code gilt nach 5.15:

$$d \leq n - k + 1$$

Also ist ein linearer Code maximal $(n-k)$-Fehler-erkennend, und die Schranke wird gerade für MDS-Codes angenommen.
Demgegenüber erkennt jeder zyklische Code Fehlerbündel der Länge $n-k$, gleichgültig wie groß d ist.

9 Reed-Solomon-Codes und Anwendungen

Reed-Solomon-Codes gehören zu den wichtigsten Blockcodes.

9.1. Konstruktion der Reed-Solomon-Codes (1960)
Sei K ein endlicher Körper, $|K| = q$, $q \geq 3$.
Setzte $n = q - 1$ und wähle d mit $2 \leq d \leq n - 1$.
Nach 8.12 ist K^* zyklisch, das heißt es existiert $\alpha \in K^*$ mit
$K^* = \{\alpha^0 = 1, \alpha, \ldots, \alpha^{n-1}\}$ $(\alpha^n = 1)$.
Sei H die folgende $(d-1) \times n$-Matrix über K:

$$H = \begin{pmatrix} 1 & \alpha & \alpha^2 & \ldots & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^4 & \ldots & \alpha^{2(n-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \alpha^{d-1} & \alpha^{(d-1)2} & \ldots & \alpha^{(d-1)(n-1)} \end{pmatrix}$$

Der *Reed-Solomon-Code* über K ist der lineare Code mit Kontrollmatrix H.
Bezeichnung $RS_q(d)$.

9.2 Satz.
$q \geq 3$ Primzahlpotenz, $n = q - 1$, $2 \leq d \leq n$.

(a) $\dim(RS_q(d)) = n - d + 1$.

(b) Die Minimaldistanz von $RS_q(d)$ ist d.

(c) $RS_q(d)$ ist ein zyklischer Code.

(d) $RS_q(d)$ ist ein MDS-Code.

Beweis:

(a) Wähle $d-1$ Spalten von H, etwa zu den Potenzen $\alpha^{i_1}, \ldots, \alpha^{i_{d-1}}$ in der ersten Zeile.

$$\det \begin{pmatrix} \alpha^{i_1} & \ldots & \alpha^{i_{d-1}} \\ \vdots & & \vdots \\ \alpha^{i_1(d-1)} & \ldots & \alpha^{i_{d-1}(d-1)} \end{pmatrix} =$$

$$\alpha^{i_1+\ldots+i_{d-1}} \cdot \begin{pmatrix} 1 & \ldots & 1 \\ \alpha^{i_1} & \ldots & \alpha^{i_{d-1}} \\ \vdots & & \vdots \\ \alpha^{i_1(d-2)} & \ldots & \alpha^{i_{d-1}(d-2)} \end{pmatrix} \overset{Vandermonde\ Det.}{=}$$

$$\alpha^{i_1+\ldots+i_{d-1}} \prod_{\substack{j,k \in \{1,\ldots,d-1\} \\ j<k}} (\alpha^{i_k} - \alpha^{i_j}) \neq 0$$

Also sind je $d-1$ Spalten von H linear unabhängig und $\mathrm{rg}(H) = d-1$. Daher ist $\dim(RS_q(d)) = n-(d-1) = n-d+1$.

(b) Da $\mathrm{rg}(H) = d-1$ sind je d Spalten linear abhängig; nach (a) sind je $d-1$ Spalten linear unabhängig. Mit 5.13 folgt (b).

(c)

$$\begin{aligned}
& x = (x_0, \ldots, x_{n-1}) \in RS_q(d) \\
\iff\quad & Hx^t = 0 \\
\iff\quad & \sum_{j=0}^{n-1} \alpha^{i\cdot j} x_j = 0 \quad \text{für alle } i = 1, \ldots, d-1 \\
\iff\quad & 0 = \alpha^i \sum_{j=0}^{n-1} \alpha^{i\cdot j} x_j = \sum_{j=0}^{n-1} \alpha^{i\cdot(j+1)} x_j = \\
& = x_{n-1} + \alpha^i x_0 + \alpha^{2i} x_1 + \ldots + \alpha^{(n-1)i} x_{n-2} \ \text{ für alle } i = 1, \ldots, d-1 \\
\iff\quad & (x_{n-1}, x_0, \ldots, x_{n-2}) \in RS_q(d).
\end{aligned}$$

(d) Dies folgt aus (a) und (b).

□

Bemerkung.
Da $RS_q(d)$ ein MDS-Code ist, ist die Einzelfehlerkorrektur genauso gut wie die Bündelkorrektur entsprechend 8.22.

9.3 Beispiel.
$q = 5$, $n = 4$, $d = 3$, $k = \dim(RS_q(3)) = 2$.
$\mathbb{Z}_5^* = \{1, 2, 3, 4\}$ $\quad \alpha = 2$ oder $\alpha = 3$ möglich.
$\alpha = 2$

$$H = \begin{pmatrix} 1 & 2 & 4 & 3 \\ 1 & 4 & 1 & 4 \end{pmatrix}$$

$$H \to \begin{pmatrix} 1 & 2 & 4 & 3 \\ 0 & 2 & 2 & 1 \end{pmatrix}$$

$$\text{Erzeugermatrix } G = \begin{pmatrix} 3 & 2 & 0 & 1 \\ 3 & 4 & 1 & 0 \end{pmatrix}$$

zyklisch: $(1320) = 2 \cdot (3410)$
$(0341) = (3201) + 4 \cdot (3410)$

9.4 Satz.
Das Erzeugerpolynom von $RS_q(d)$ ist $g(x) = \prod_{j=1}^{d-1}(x-\alpha^j)$ und das Kontrollpolynom ist $h(x) = (x-1)\prod_{j=d}^{n-1}(x-\alpha^j) = \prod_{j=d}^{n}(x-\alpha^j)$.

Beweis:
Sei $g(x) = \sum_{i=0}^{d-1} a_i x^i$ das Erzeugerpolynom von $RS_q(d)$.
(Beachte: $n - k = n - (n - d + 1) = d - 1$)
Dann ist $c = (a_0, a_1, \ldots, a_{d-1}, 0, \ldots, 0) \in RS_q(d)$.
Aus $Hc^t = 0$ folgt

$$\begin{aligned} a_0 + a_1\alpha + \ldots + a_{d-1}\alpha^{d-1} &= 0 \\ a_0 + a_1\alpha^2 + \ldots + a_{d-1}\alpha^{2(d-1)} &= 0 \\ &\vdots \\ a_0 + a_1\alpha^{d-1} + \ldots + a_{d-1}\alpha^{(d-1)(d-1)} &= 0 \end{aligned}$$

Daraus folgt, dass $\alpha, \alpha^2, \ldots, \alpha^{d-1}$ Nullstellen von $g(x)$ sind, also
$g(x) = \prod_{j=1}^{d-1}(x - \alpha^j)$.
Für alle $\beta \in K^*$ gilt $\beta^{q-1} = \beta^n = 1$, das heißt $\alpha^0, \alpha^1, \ldots, \alpha^{n-1}$ sind die Nullstellen von $x^n - 1 \in K[x]$. Folglich: $x^n - 1 = \prod_{j=1}^{n}(x - \alpha^j)$. Daraus folgt die Behauptung für $h(x)$. □

Es gibt viele Decodierverfahren für Reed-Solomon-Codes, die besser sind als die für alle linearen Codes mögliche Syndrom-Decodierung. Wir beschreiben eines dieser Verfahren.

9.5. Peterson-Gorenstein-Ziesler-Decodierer
(Peterson 1960; Gorenstein-Ziesler 1961)

Gegeben: $\mathcal{C} := RS_q(d)$, $|K| = q$, $H = \begin{pmatrix} 1 & \alpha & \alpha^2 & \ldots & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^4 & \ldots & \alpha^{2(n-1)} \\ \vdots & & & \vdots & \\ 1 & \alpha^{d-1} & \alpha^{(d-1)2} & \ldots & \alpha^{(d-1)(n-1)} \end{pmatrix}$

1. Sei $c = (c_0, \ldots, c_{n-1}) \in \mathcal{C}$, $v = c + f$ mit Fehlervektor $f = (f_0, \ldots, f_{n-1}) \in K^n$.

   Wir machen folgende Annahme :

   Es sind maximal $\lfloor \frac{d-1}{2} \rfloor$ Fehler aufgetreten, das heißt $wt(f) = t \leq \lfloor \frac{d-1}{2} \rfloor$
   (In diesem Fall gibt es eindeutige Hamming-Decodierung).

   Bezeichnungen :

   (a) $(s_1, \ldots, s_{d-1})^t = Hv^t = Hf^t$ Syndrom von v.
   (b) $F = Tr(f) = \{i_1, \ldots, i_t\}$ Menge der Fehlerpositionen.
   (c) $q(x) = \prod_{\ell \in F}(1 - \alpha^\ell x) = q_0 + q_1 x + \ldots + q_t x^t \in K[x]$ $(q_0 = 1)$, *Fehlerortungspolynom.*

Beachte : $\ell \in F \iff q(\alpha^{-\ell}) = 0 \qquad (\alpha^{-\ell} = \alpha^{n-\ell})$.

Der Algorithmus besteht aus 2 Hauptschritten:

I) Bestimme $q(x)$ und damit F.

II) Bestimme mit Hilfe von F den Fehlervektor f.

2. Wir zeigen zunächst zwei Gleichungen für die Komponenten s_i des Syndroms $(s_1, \ldots, s_{d-1})^t$.

(a) $s_i = \sum\limits_{\ell \in F} \alpha^{\ell i} f_\ell$, $i = 1, \ldots, d-1$:

$$s_i = 1 \cdot f_0 + \alpha^i f_1 + \alpha^{2i} \cdot f_2 + \ldots + \alpha^{(n-1)i} f_{n-1} = \sum_{\ell \in F} \alpha^{\ell i} f_\ell.$$

(b) $s_i = -\sum\limits_{j=1}^{t} q_j s_{i-j}$, $i = t+1, \ldots, d-1$:

$$\begin{aligned} \sum_{j=0}^{t} q_j s_{i-j} &\overset{a)}{=} \sum_{j=0}^{t} q_j \sum_{\ell \in F} \alpha^{\ell(i-j)} f_\ell \\ &= \sum_{\ell \in F} \alpha^{\ell i} f_\ell \sum_{j=0}^{t} q_j \alpha^{-\ell j} \\ &= \sum_{\ell \in F} \alpha^{\ell i} f_\ell \, q(\alpha^{-\ell}) = 0 \end{aligned}$$

Da $q_0 = 1$, folgt die Behauptung.

3. Aus 2) folgt, dass man bei Kenntnis der Fehlerpositionen $F = \{i_1, \ldots, i_t\}$ den Fehlervektor berechnen kann. Dazu muss man natürlich nur $f_{i_1}, \ldots, f_{i_t}$ bestimmen. Dazu betrachten wir die ersten t Gleichungen in $2(a)$ und schreiben diese in Matrixform:

$$\underbrace{\begin{pmatrix} \alpha^{i_1} & \cdots & \alpha^{i_t} \\ \alpha^{2i_1} & \ldots & \alpha^{2i_t} \\ \vdots & & \vdots \\ \alpha^{ti_1} & \cdots & \alpha^{t \cdot i_t} \end{pmatrix}}_{\text{invertierbar, vgl. Beweis 9.2.a)}} \begin{pmatrix} f_{i_1} \\ \vdots \\ \vdots \\ f_{i_t} \end{pmatrix} = \begin{pmatrix} s_1 \\ \vdots \\ \vdots \\ s_t \end{pmatrix}$$

Gleichungssystem hat eindeutige Lösung, z.B Cramer Regel.

4. Wie bestimmt man F? Dazu bestimmt man $q(x)$.
Wir zeigen zunächst wie man $q(x)$ bestimmen kann, wenn man $|F| = t$, die Anzahl der Fehler, kennt:

Bilde für $1 \leq r \leq \left\lfloor \frac{d-1}{2} \right\rfloor$ die Matrix

$$S_r = \begin{pmatrix} s_1 & s_2 & \cdots & s_r \\ s_2 & s_3 & \cdots & s_{r+1} \\ \vdots & & & \vdots \\ s_r & s_{r+1} & \cdots & s_{2r-1} \end{pmatrix}$$

Die ersten t Gleichungen in $2(b)$ besagen:

$$S_t \cdot \begin{pmatrix} q_t \\ \vdots \\ q_1 \end{pmatrix} = - \begin{pmatrix} s_{t+1} \\ \vdots \\ s_{2t} \end{pmatrix} \qquad (*)$$

Ist t die Anzahl der Fehler, so gilt:

$$S_t = \begin{pmatrix} \alpha^{i_1} & \cdots & \alpha^{i_t} \\ \alpha^{2i_1} & \cdots & \alpha^{2i_t} \\ \vdots & & \vdots \\ \alpha^{ti_1} & \cdots & \alpha^{t \cdot i_t} \end{pmatrix} \cdot \begin{pmatrix} f_{i_1} & 0 & \cdots & 0 \\ 0 & f_{i_2} & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & f_{i_t} \end{pmatrix} \cdot \begin{pmatrix} 1 & \alpha^{i_1} & \cdots & \alpha^{(t-1)i_1} \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ 1 & \alpha^{i_t} & \cdots & \alpha^{(t-1)i_t} \end{pmatrix}$$

wie man leicht mit $2(a)$ nachrechnet.
Alle rechts auftretenden Matrizen sind invertierbar, also auch S_t.
Damit: Kennt man t, so kann man aus $(*)$ $q_t, \ldots, q_1$ eindeutig bestimmen.

5. Wie bestimmt man $t = |F|$?
   Sei $e = \left\lfloor \frac{d-1}{2} \right\rfloor$.
   Es ist $\det(S_t) \neq 0$. Ist $t < r \leq e$, so ist $\det(S_r) = 0$:
   Die erste Behauptung haben wir schon in (4) gezeigt.
   Die Gleichungen in $2(b)$ besagen, dass die r-te Spalte von S_r von den unmittelbar davor stehenden t Spalten linear abhängig ist. Also ist $\mathrm{rg}(S_r) < r$, das heißt $\det(S_r) = 0$.

6. Decodieralgorithmus

   (Korrigiert korrekt, wenn maximal $\left\lfloor \frac{d-1}{2} \right\rfloor$ Fehler aufgetreten sind. Ansonsten falsche Korrektur oder Fehlermeldung.)
   (Bezeichnungen wie in 1.)

   1) Berechne das Syndrom $(s_1, \ldots, s_t)^t = Hv^t$.
      Sind alle $s_i = 0$, so setze $c = v$, fertig.
   2) Bestimme $t = \max\{r \mid 1 \leq r \leq \left\lfloor \frac{d-1}{2} \right\rfloor, \det(S_r) \neq 0\}$.
      Sind alle Determinanten gleich Null, so Ausgabe von Fehlermeldung, stop.

3) Bestimme Fehlerortungspolynom $q(x)$ aus dem Gleichungssystem

$$S_t \cdot \begin{pmatrix} q_t \\ \vdots \\ q_1 \end{pmatrix} = - \begin{pmatrix} s_{t+1} \\ \vdots \\ s_{2t} \end{pmatrix}$$

4) Bestimme die Nullstellen von $q(x)$, etwa durch Einsetzen der Elemente $\alpha^0, \alpha^1, \ldots, \alpha^{n-1}$. Ist $q(\alpha^{n-\ell}) = 0$, so ist ℓ Fehlerposition. Dadurch erhält man $F = \{i_1, \ldots, i_t\}$.

5) Berechne den Fehler $f = (f_1, \ldots, f_n)$ aus dem Gleichungssystem

$$\begin{pmatrix} \alpha^{i_1} & \cdots & \alpha^{i_t} \\ \alpha^{2i_1} & \ldots & \alpha^{2i_t} \\ \vdots & & \vdots \\ \alpha^{ti_1} & \cdots & \alpha^{t \cdot i_t} \end{pmatrix} \begin{pmatrix} f_{i_1} \\ \vdots \\ \vdots \\ f_{i_t} \end{pmatrix} = \begin{pmatrix} s_1 \\ \vdots \\ \vdots \\ s_t \end{pmatrix}$$

6) $c = v - f$.

7. Der Decodieralgorithmus decodiert also nicht in jedem Fall (keine Hamming-Decodierung), sondern wenn maximal $\left\lfloor \frac{d-1}{2} \right\rfloor$ Fehler aufgetreten sind (und dann korrekt): BMD-Decodierung (begrenzte Minimaldistanz).
Sind mehr als $\left\lfloor \frac{d-1}{2} \right\rfloor$ Fehler aufgetreten, so ist einer von 3 Fällen möglich:
   - Fehlermeldung.
   - Decodierung in ein Wort, das kein Codewort ist (feststellbar mit Kontrollmatrix).
   - Decodierung in ein falsches Codewort (decoder error).

8. Besonders aufwändig in 6. sind die Schritte 2,3,5.
Hier gibt es Algorithmen, die (z.B. aufgrund der speziellen Form der auftretenden Koeffizientenmatrix) diese Schritte effizienter ausführen.

Stichworte:
- Verwendung des euklidschen Algorithmus in $K[x]$.
- Berlekamp-Massey-Algorithmus.

Literatur:
- Willems, Codierungstheorie, Kapitel 9.
- Friedrichs, Kanalcodierung, Kapitel 7.
- Justeson, Høholdt, A Course in Error-Correcting-Codes Kapitel 5.

9.6 Beispiel.
Wir benutzen den Code aus 9.3 $RS_5(3)$, $d = 3$, $\alpha = 2$, $\left\lfloor \frac{d-1}{2} \right\rfloor = 1$.

$$H = \begin{pmatrix} 1 & 2 & 4 & 3 \\ 1 & 4 & 1 & 4 \end{pmatrix} \quad G = \begin{pmatrix} 3 & 2 & 0 & 1 \\ 3 & 4 & 1 & 0 \end{pmatrix}$$

$c = (3201)$, $v = (3221)$, also $f = (0, 0, 2, 0)$

1. $\begin{pmatrix} 1 & 2 & 4 & 3 \\ 1 & 4 & 1 & 4 \end{pmatrix} \begin{pmatrix} 3 \\ 2 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} s_1 \\ s_2 \end{pmatrix}$
2. $S_1 = (s_1) = (3) \Rightarrow t = 1$.
3. $s_1 q_1 = -s_2$
   $3 \cdot q_1 = -2 = 3$, $q_1 = 1$
   $q(x) = 1 + x$
4. Nullstelle von $q(x)$: $4 = 2^2$, Fehlerposition $i_1 = 2$.
5. $\alpha^2 \cdot f_2 = s_1$
   $4 \cdot f_2 = 3$, $f_2 = 2$
   $f = (0, 0, 2, 0)$
6. $c = (3221) - (0020) = (3201)\surd$

Reed-Solomon-Codes haben viele Anwendungen (In der militärischen Nachrichtenübertragung mehrerer NATO-Länder wird $RS_{2^5}(17)$ verwendet). Sie wurden (und werden) außerdem in der Raumfahrt eingesetzt, z.B.

Voyager 2 (Jupiter, Uranus)

Galileo (Jupiter)

Pathfinder (Mars)

und zwar in der Regel ein $RS_{2^8}(33)$ Code (Länge 255, Dimension 233 über K, $|K| = 2^8$) in Verbindung mit einem Faltungscode (vgl. Kapitel 10). Diese beiden Codes werden durch sogennantes Interleaving miteinander verbunden, eine Technik, die wir gleich vorstellen werden.
Sie wird nämlich auch benutzt zur Codierung von Daten auf Audio-CD's (mittels Reed-Solomon-Codes), die wir uns jetzt genauer ansehen.

9.7 Definition.
Sei $\mathcal{C}$ ein Code in K, $|K| = q$, q Primzahlpotenz. *Interleaving* (Spreizung) dieses Codes bis zur Tiefe s liefert einen Code $\mathcal{C}(s) \subseteq K^{ns}$ auf folgende Weise:

$$\begin{aligned} \mathcal{C}(s) \quad = \quad & \{(c_{11}, \ldots, c_{s1}, \ldots, c_{1n}, \ldots, c_{sn}) : \\ & (c_{11}, c_{12}, \ldots, c_{1n}), \ldots, (c_{s1}, \ldots, c_{sn}) \in \mathcal{C}\} \end{aligned}$$

Das heißt, je s Codewörter $(c_{11}, c_{12}, \ldots, c_{1n}), \ldots, (c_{s1}, \ldots, c_{sn})$ werden als Zeilen in eine Matrix geschrieben:

$$\begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ \vdots & \vdots & & \vdots \\ c_{s1} & c_{s2} & \cdots & c_{sn} \end{pmatrix};$$

diese wird dann spaltenweise in einen Vektor der Länge sn ausgelesen.

9.8 Satz.
Sei $\mathcal{C}$ ein $[n,k]$-Code mit $d(\mathcal{C}) = d$. Dann gilt:

a) $\mathcal{C}(s)$ ist ein $[ns, ks]$-Code mit $d(\mathcal{C}(s)) = d$.

b) Ist $\mathcal{C}$ zyklisch, so auch $\mathcal{C}(s)$.

c) Kann $\mathcal{C}$ Fehlerbündel der Länge b entdecken (korrigieren), so kann $\mathcal{C}(s)$ Bündelfehler der Länge bs entdecken (korrigieren).

Beweis.

a) Ist $(c_{11}, c_{12}, \ldots, c_{1n}), \ldots, (c_{k1}, \ldots, c_{kn})$ eine Basis von $\mathcal{C}$, so sind die Codewörter die aus den Matrizen

$$\begin{pmatrix} 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ c_{i1} & \cdots & \cdots & \cdots & \cdots & \cdots & c_{in} \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \end{pmatrix} \leftarrow \ell\text{-te Zeile} \qquad \begin{matrix} i = 1, \ldots, k, \\ \ell = 1, \ldots, s \end{matrix}$$

gebildet werden, linear unabhängig und erzeugen $\mathcal{C}(s)$, $\dim(\mathcal{C}(s)) = ks$. Klar: $d(\mathcal{C}(s)) = d$.

b) Man kann leicht zeigen: Ist $g(x)$ Erzeugerpolynom von $\mathcal{C}$, so ist $\tilde{g}(x) := g(x^s)$ Erzeugerpolynom von $\mathcal{C}(s)$, $\tilde{g}(x) \mid x^{ns} - 1$.
(siehe z.B. Lüthkebohmert, Satz 4.2.1)

c) Ein Fehlerbündel der Länge $\leq bs$ in $\mathcal{C}(s)$ betrifft maximal b Stellen in Codewörtern von $\mathcal{C}$, da die Komponenten eines Codewortes aus $\mathcal{C}$ in $\mathcal{C}(s)$ Abstand s haben. Daraus folgt die Behauptung.

□

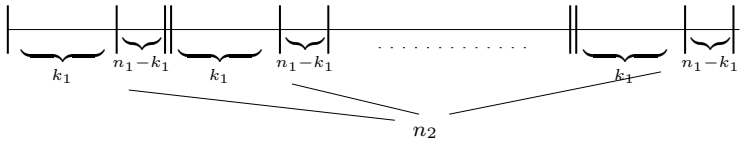
Für die CD-Codierung benötigt man eine Variante des Interleaving:

9.9 Definition.
Seinen $\mathcal{C}_1, \mathcal{C}_2$ Codes der Länge n_1 bzw. n_2 über K, $|K| = q$. Es sei $\dim(\mathcal{C}_1) = k_1$ und $\mathcal{C}_1$ besitze eine Erzeugermatrix (E_{k_1}, A), A $k_1 \times (n_1 - k_1)-$Matrix.
Das *Cross-Interleaving* des *inneren Codes* $\mathcal{C}_1$ mit dem *äußeren Code* $\mathcal{C}_2$ ist der Code, der aus folgenden Codewörtern der Länge $n_1 \cdot n_2$ besteht:

Wähle irgendwelche Codewörter $c_1, \ldots, c_{k_1} \in \mathcal{C}_2 \subseteq K^{n_2}$ und bilde die $n_2 \times n_1$-Matrix

$$\begin{aligned} M &:= (c_1^t, \ldots, c_{k_1}^t) \cdot (E_{k_1}, A) \\ &= (c_1^t, \ldots, c_{k_1}^t, a_{11}c_1^t + \cdots + a_{k_1,1}c_{k_1}^t, \ldots, a_{1,n_1-k_1}c_1^t + \ldots + a_{k_1,n_1-k_1}c_{k_1}^t) \end{aligned}$$

M wird zeilenweise ausgelesen und bildet ein Codewort der Länge $n_1 \cdot n_2$.



Die n_2 Stücke der Länge k_1 hintereinandergelesen bilden gerade eine Codeverschachtelung von $\mathcal{C}_2$ zur Tiefe k_1.
Die n_2 Zeilen von M bilden Codewörter aus $\mathcal{C}_1$ (Multiplizieren der Zeilen von $(c_1^t, \ldots, c_{k_1}^t)$ mit Erzeugermatrix von $\mathcal{C}_1$), die n_1 Spalten sind Codewörter aus $\mathcal{C}_2$.
Selbst wenn $\mathcal{C}_1$ und $\mathcal{C}_2$ keine besonders guten Korrektureigenschaften haben, führt das Cross-Interleaving zu deutlich besserem Korrekturverhalten:

9.10 Bemerkung.
Bezeichnungen wie in 9.9.
$M = (c_1^t, \ldots, c_{k_t}^t) \cdot (E_{k_1}, A)$ wird gesendet (zeilenweise ausgelesen),
$n_2 \times n_1$-Matrix $\widetilde{M}$ wird empfangen (rekonstruiert aus Vektor der Länge $n_1 \cdot n_2$).
Mit Hilfe einer Kontrollmatrix von $\mathcal{C}_1$ wird überprüft, ob Zeilen von $\widetilde{M}$ fehlerhaft sind oder zu $\mathcal{C}_1$ gehören (ggf. wird eine kleine Fehleranzahl korrigiert). Die fehlerhaften Zeilen werden ausgelöscht.
Wenn k_1, abhängig von den Fehlereigenschaften des Kanals, geeignet gewählt wurde, sind nur wenige Zeilen, also jeweils nur wenige Positionen in den Spalten von $\widetilde{M}$ betroffen. Dies ist insbesondere bei Bursts der Fall.
Die Fehler (Auslöschungen) in den Spalten werden dann durch die Korrekturmöglichkeiten von $\mathcal{C}_2$ korrigiert.

Beachte :
Nur an den 'ausgelöschten' Stellen können Fehler aufgetreten sein, das heißt

man kennt die Fehlerpositionen, bzw. ihre Anzahl ist deutlich eingeschränkt. Das verbessert die Korrekturmöglichkeiten von $\mathcal{C}_2$ im Vergleich zu Fehlern, deren Positionen man nicht kennt (und erspart Rechenarbeit, vergleiche Peterson-Gorenstein-Ziesler-Decodierer).

Problem: Man muss alle n_2 Zeilen der Matrix M empfangen haben (also alle k_1 Codewörter von $\mathcal{C}_2$), bevor man mit der Decodierung beginnen kann.

Entsprechend der vorangegangenen Diskussion definieren wir:

9.11 Definition.
Ein Fehler in einem Codewort heißt *Auslöschung*, wenn man seine Position kennt.

9.12 Satz.
Sei $\mathcal{C}$ ein $[n,k,d]$-Code über K. Dann kann $\mathcal{C}$ in einem Codewort a Auslöschungen und zusätzlich (an den übrigen Positionen) t Fehler korrigieren, falls $d \geq 2t+a+1$.
(Speziell : Ist $a \leq d-1$, so kann $\mathcal{C}$ a Auslöschungen korrigieren, aber nur $\lfloor \frac{d-1}{2} \rfloor$ Fehler, wenn man deren Positionen nicht kennt.)

Beweis:
Sei $c \in \mathcal{C}$ gesendet, v empfangen. Es seien t Fehler und a Auslöschungen aufgetreten mit $d \geq 2t+a+1$. Die Auslöschungen in v seien durch ? ersetzt. Sei I die Menge der Koordinaten, an denen Auslöschungen aufgetreten sind, $M = \{m \in K^n : Tr(m) \subseteq I\}$.
Nach Voraussetzung existiert $m \in M$ mit $d(c, v(m)) \leq t$, wobei $v(m)$ aus v entsteht, indem man die Positionen mit ? durch die Einträge von m an den entsprechenden Positionen ersetzt.
Angenommen es existiert $c' \in \mathcal{C}$, $m' \in M$ mit $d(c', v(m')) \leq t$.
Dann $d(c,c') \leq d(c, v(m)) + d(v(m), v(m')) + d(c, v(m')) \leq 2t + a < d$, Widerspruch.
Also gibt es nur ein Codewort, nämlich c, und nur ein $m \in M$ mit $d(c, v(m)) \leq t$. c kann also bestimmt werden. □

9.13 Bemerkung.
Bezeichnungen wie in 9.12. $\mathcal{C}$ sei ein binärer Code.
Hat man für $\mathcal{C}$ ein Decodierverfahren, dass bis zu $\lfloor \frac{d-1}{2} \rfloor$ Fehler korrigiert, so kann man dieses Verfahren auch einsetzen, um a Auslöschungen und t Fehler zu korrigieren, falls $d \geq 2t+a+1$.

Beweis:
Bilde v_0, indem man in v alle Auslöschungen ? durch 0 ersetzt,
v_1, indem man alle Auslöschungen ? durch 1 ersetzt.

In einem der beiden v_i ist mindestens die Hälfte der ersetzen Stellen korrekt. Dieser Vektor enthält also höchstens $\lfloor \frac{a}{2} \rfloor + t \leq \lfloor \frac{d-1}{2} \rfloor$ Fehler. Es lässt sich mit dem Decodierverfahren zu c decodieren.
(Decodiere, sofern möglich, beide und teste wo die Anzahl der Korrekturen $\leq \lfloor \frac{d-1}{2} \rfloor$ war.) □

Für die Codierung auf Audio-CD's wird ein Cross-Interleaving mit mehrfach verkürzten Reed-Solomon-Codes vorgenommen.

9.14 Satz.
Sei $\mathcal{C}$ ein $[n, k, d]$-MDS-Code über K (das heißt $d = n - k + 1$), $k > 1$. Dann entsteht durch Verkürzung (an einer Stelle i) ein $[n-1, k-1, d]$-MDS-Code $\check{\mathcal{C}}_i$.

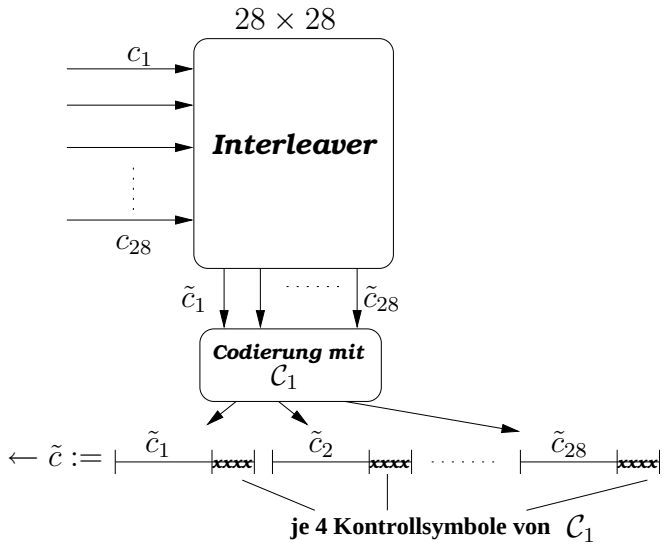
Beweis:
Sei $k' = \dim(\check{\mathcal{C}}_i)$ und $d' = d(\check{\mathcal{C}}_i)$.
Nach 6.13.a) e) ist $k - 1 \leq k' \leq k$ und $d' \geq d$ (denn $k > 1$).
Somit gilt: $(n-1) - d' + 1 \leq (n-1) - d + 1 = n - d = k - 1 \leq k'$.
Aus der Singleton-Schranke (5.15) folgt Gleichheit, also $d' = d$, $k' = k - 1$ und $\check{\mathcal{C}}_i$ ist MDS-Code. □

9.15. Cross-Interleaving bei Audio-CD's

(a) Starte mit $RS_{2^8}(5)$. Dies ist ein $[255, 251, 5]$-Code über $K = \mathbb{F}_{2^8}$ (d.h. $|K| = 2^8$).
Verkürze $RS_{2^8}(5)$ um (die letzten) 223 bzw. 227 Stellen. Dies liefert nach 9.14 einen $[32, 28, 5]$-Code $\mathcal{C}_1$ über $\mathbb{F}_{2^8}$ und einen $[28, 24, 5]$-Code $\mathcal{C}_2$ über $\mathbb{F}_{2^8}$, die beide MDS-Codes sind. Indem man gegebenenfalls zu einem äquivalenten Code übergeht, kann man annehmen, dass $\mathcal{C}_1$ eine Erzeugermatrix der Form (E_{28}, A) besitzt, wobei A eine 28×4-Matrix über F_{2^8} ist.

Jetzt Cross-Interleaving mit $\mathcal{C}_1$ als innerem und $\mathcal{C}_2$ als äußerem Code.
Bezeichnung: CIRC : Cross-Interleaved-Reed-Solomon-Code.

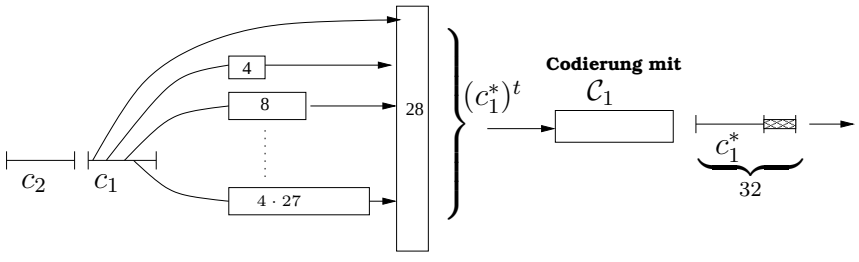
Bei Audio-CD's wird eine Folge von Codewörtern $c_1, c_2, \ldots \in \mathcal{C}_2$ produziert.
Für das in 9.9 beschriebene Cross-Interleaving mit $\mathcal{C}_1$ werden jeweils 28 ($= \dim(\mathcal{C}_1)$) Codewörter verarbeitet (diese haben Länge 28).



Zeilenweise einlesen, spaltenweise ausgeben. $28 \cdot 28 = 784$ Symbole aus $\mathbb{F}_{2^8}$ werden zwischengespeichert.

Dann die nächsten 28 Codewörter von $\mathcal{C}_2$, etc.
Angenommen bei der Speicherung (Übertragung) von $\tilde{c}$ tritt ein Fehlerbündel der Länge $\leq 3 \cdot 32 + 1 = 97$ auf. Dann sind in $\tilde{c}$ maximal 4 nebeneinanderliegende $\tilde{c}_j, \tilde{c}_{j+1}, \tilde{c}_{j+2}, \tilde{c}_{j+3}$ betroffen. Durch Multiplikation jedes der $\tilde{c}_i$ mit der Kontrollmatrix von $\mathcal{C}_1$ werden diese $\tilde{c}_j$ erkannt und dann ausgelöscht (vgl. 9.10). Schreibt man die $\tilde{c}_i$ untereinander, so stehen in den Spalten Wörter von $\mathcal{C}_2$, jedes von ihnen hat maximal 4 Auslöschungen. Da $d(\mathcal{C}_2) = 5$ können diese nach 9.12/9.13 korrigiert werden. Damit Bursts der Länge ≤ 97 korrigierbar. Es geht aber noch besser.

(b) Dazu wird sogenanntes *Cross-Interleaving mit 4-stufiger Verzögerung* angewandt.
Der Interleaver besteht aus 28 Speichervektoren der Länge $0, 4, 8, \ldots, 4 \cdot 27$ (insgesamt $4 \cdot \frac{27 \cdot 28}{2} = 1512$ Symbole aus $\mathbb{F}_{2^8}$ werden zwischengespeichert). Diese sind mit Nullen vorbesetzt.



Jetzt werden die c_j einzeln eingelesen, und zwar das i-te Symbol von c_j von links in den i-ten Speicher (der Länge $4 \cdot (i-1)$), dafür wird das

rechts stehende Symbol im jeweiligen Speicher ausgegeben. Das erste Bit von c_j wird direkt (ohne Zwischenspeicherung) ausgegeben.
Das entstehende Ausgabewort der Länge 28 (=Länge von $\mathcal{C}_2$) wird jetzt mit $\mathcal{C}_1$ kodiert (liefert Länge 32). Wichtig hier: Länge von $\mathcal{C}_2 = \dim(\mathcal{C}_1)$.
Auf diese Weise werden nacheinander (ohne Verzögerung wie bei (a)), die Codewörter $c_1, c_2, \ldots$ verarbeitet.

Seien

$$\begin{aligned} c_1 &= (c_{1,1}, \ldots, c_{1,28}) \\ c_2 &= (c_{2,1}, \ldots, c_{2,28}) \\ &\vdots \end{aligned}$$

Dann

$$\begin{aligned} c_1^* &= (c_{1,1}, 0, \ldots, 0 \mid ****) \\ c_2^* &= (c_{2,1}, 0, \ldots, 0 \mid ****) \\ &\vdots \\ c_5^* &= (c_{5,1}, c_{1,2}, 0, \ldots, 0 \mid ****) \\ c_6^* &= (c_{6,1}, c_{2,2}, 0, \ldots, 0 \mid ****) \\ &\vdots \\ c_9^* &= (c_{9,1}, c_{5,2}, c_{1,3}, 0, \ldots, 0 \mid ****) \\ &\vdots \\ c^*_{\underbrace{27 \cdot 4 + 1}_{109}} &= (c_{27\cdot 4+1,1}, c_{26\cdot 4+1,2}, \ldots, c_{5,27}, c_{1,28} \mid ****) \\ c^*_{\underbrace{27 \cdot 4 + 2}_{110}} &= (c_{110,1}, c_{106,2}, \ldots, c_{6,27} c_{2,28} \mid ****) \end{aligned}$$

In der Folge der c_j^* liegen zwischen den einzelnen Komponenten eines Codewortes $c_i \in C_2$ jeweils $4 \cdot 32$ andere Symbole. c_i ist auf $27 \cdot 4 + 1 = 109$ aufeinander folgende Codewörter c_j^* verteilt.
Tritt jetzt ein Burst der Länge $\leq 15 \cdot 32 + 1 = 481$ auf, so sind maximal 16 aufeinanderfolgende c_j^* betroffen und damit maximal 4 aufeinanderfolgende Komponenten eines Codewortes c_i.
Macht man das Interleaving durch entsprechendes Deinterleaving wieder rückgängig, so können wegen $d(\mathcal{C}_2) = 5$ diese als Auslöschungen markierten Positionen korrigiert werden. Also: Bündelfehler der Länge ≤ 481 korrigierbar, keine Verzögerung beim Interleaving/Deinterleaving. Allerdings höherer Speicherbedarf.

9.16. Datenspeicherung auf Audio-CD's

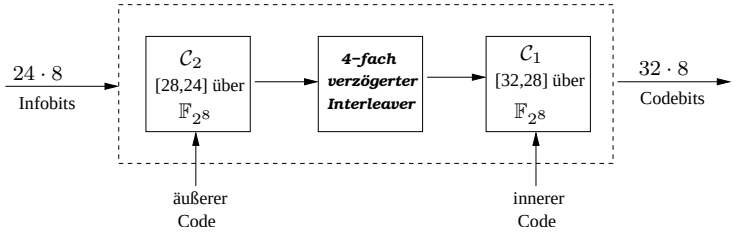
(a) Das analoge Tonsignal wird 44100 mal pro Sekunde abgetastet; diese diskreten Werte (Amplituden der Druckwelle des Schalls), die sogenannten Audiosamples, genügen, um nach dem Shannon'schen Abtastheorem al-

le Frequenzen bis 22 kHZ exakt rekonstruieren zu können.

Diesen Audiosamples wird ein Wert auf einer Skala von 0 bis $2^{16} - 1 = 65535$ zugeordnet. Man macht das für linken und rechten Kanal getrennt.

Jedes Audiosample entspricht also einem Paar von Werten in $\mathbb{Z}_2^{16}$. Identifiziert man $\mathbb{Z}_2^8$ mit $\mathbb{F}_{2^8}$, so entspricht jedem Audiosample ein Element in $\mathbb{F}_{2^8}^4$. Jeweils 6 Audiosamples entsprechen dann einem Wort in $\mathbb{F}_{2^8}^{24}$; es enthält $24 \cdot 8$ Infobits.

(b) Nun erfolgt die CIRC-Codierung entsprechend 9.15. Jedes Infowort in $\mathbb{F}_{2^8}^{24}$ wird mit dem äußeren verkürzten Reed-Solomon $[28, 24]$-Code $\mathcal{C}_2$ in ein $\mathcal{C}_2$-Codewort der Länge 28 über $\mathbb{F}_{2^8}$ codiert; diese werden dann mit einem 4-fach verzögerten Interleaver (9.15.b)) und anschließender Codierung mit dem inneren verkürzten Reed-Solomon $[32, 28]$-Code $\mathcal{C}_1$ in Worte der Länge 32 über $\mathbb{F}_{2^8}$ codiert. Da die Elemente des $\mathbb{F}_{2^8}$ durch 8 Bits repräsentiert werden, entstehen Codeworte mit $32 \cdot 8 = 256$ Bits.



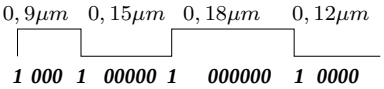
Die Rate der Codierung beträgt also $\frac{24 \cdot 8}{32 \cdot 8} = \frac{3}{4}$. Bursts der Länge ≤ 481 (Symbole in $\mathbb{F}_{2^8}$), das heißt von maximal $8 \cdot 481 = 3848$ Bits, können dabei korrigiert werden.

(c) Ein Codewort von $256 = 32 \cdot 8$ Codebits wird nun in einen sogenannten *Rahmen* von 588 sogenannten *Kanalbits* umgesetzt; diese werden dann auf der CD gespeichert (siehe (d)).
Dazu werden einem Codewort zunächst 8 Bits hinzugefügt, die Displayinformationen (Zeit,Titelnummer) enthalten (*Subcode*).
Ein solcher Bit-Vektor kann nicht unmittelbar auf die CD übertragen werden. Aus Gründen, die wir in (d) erläutern, darf zwischen 2 Einsen höchstens 10 mal eine Null stehen und es müssen zwischen 2 Einsen mindestens 2 Nullen stehen. Berechnet man die Anzahl $\alpha(n)$ der $\{0, 1\}$-Folgen der Länge n, die diese beiden Bedingungen erfüllen, so stellt man fest, dass $n = 14$ die kleinste Zahl mit $\alpha(n) > 2^8$ ist; es ist $\alpha(n) = 267$.
Jedes erweiterte Codewort der Länge 264 (über $\mathbb{Z}_2$) wird in 33 8-Bit-Vektoren zerlegt, und jeder dieser wird in einen 14-Bit-Vektor umgesetzt, so dass die obigen Bedingungen erfüllt sind (EFM-Verfahren:

Eight-to-Fourteen Modulation.
Damit auch die Übergänge zwischen zwei 14-Bit-Vektoren die obige 0,1-Bedingung erfüllen, werden hinter jeden nochmals 3 Bits eingefügt (Koppelbits, merging Bits). Dafür gibt es mehrere Möglichkeiten. Wie diese Koppelbits jeweils gewählt werden, erläutern wir in (d).
Am Ende des so entstandenen Blocks werden noch 24 Bits zur Synchronisation (Identifikation der nächsten Codesequenz) und 3 Puffer-Bits angehängt. Insgesamt führt jedes Codewort mit 256 Bits dann zu $33 \cdot 8 \cdot \frac{14}{8} + 33 \cdot 3 + 24 + 3 = 588$ Bits des Rahmens, die zu ursprünglich 6 Audiosamples gehören. Pro Sekunde werden also $\frac{44100 \cdot 588}{6} = 4.321.800$ Kanalbits produziert, die beim Abspielen auch in derselben Zeit wieder in Audiosignale umgesetzt werden müssen: Kanalbitrate von $4,3218$ MBit/sec.

(d) Physikalische Speicherung auf CD:

Spiralförmig nach innen verlaufende Spur. Dort Vertiefungen $(0,12\mu m)$ (Pits) und Nicht-Vertiefungen (Lands).
Spurbreite: $0,6\mu m$; Abstand zweier Spuren ca. $1\mu m$.
Übergänge Pit/Land bzw. Land/Pit:1, alles andere 0. Kanalbit auf einer Spur: Länge $0,3\mu m$.



Bei den Pits wird der Laserstrahl (Durchmesser ca. $1\mu m$) wegen Interferenzen weniger stark reflektiert als bei den Lands.
Auflösung erfordert, dass zwischen zwei Einsen mindestens 2 Nullen stehen. Synchronisation und Spurhaltung erfordern zwischen zwei Einsen höchstens 10 Nullen (vgl. (c)).
Synchronistion: 'Bit-Uhr' wird bei jedem Übergang neu synchronisiert und zählt die aufeinanderfolgenden Nullen.
Darüberhinaus sollte die Gesamtlänge der Pits und Lands ungefähr gleich sein. Dies dient ebenfalls der Fokussierung des Lasers und zur Hell-Dunkel-Unterscheidung (Dies wird durch die Wahl der Koppelbits sichergestellt; vgl (c)).

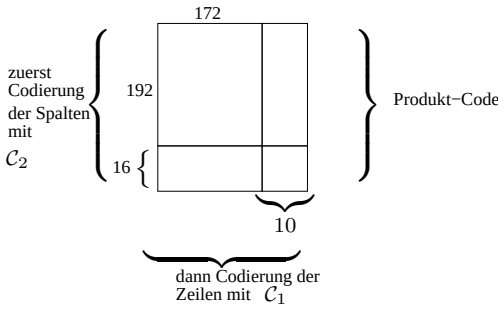
Abspieldauer einer CD ≈ 74 min.
Erfordert $74 \cdot 60 \cdot 4.321.800 \approx 19 \cdot 10^9$ Bits, die abgespeichert werden müssen (vgl. (c)).
Erfordert Spurlänge von $19 \cdot 10^9 \cdot 0,3 \cdot 10^{-6} = 5,7$ km.
Lesegeschwindigkeit: $\frac{5700}{60 \cdot 74} \approx 1,2 - 1,3$ m/sec.

(e) Decodierung wie in (b) beschrieben kann Bursts von 3848 Code-Bits korrigieren, diese sind in $3848 \cdot \frac{588}{256} \approx 8850$ Kanalbits enthalten. Lange eines korrigierbaren Burstfehlers ist also circa $8850 \cdot 0,3\mu m \approx 2,65mm$ Spurlänge.
Durch besondere Abspeicherung der codierten 6 Audiosamples innerhalb eines Rahmens lassen sich besonders günstig durch Interpolation Daten ersetzen, die verloren gegangen sind. Dies führt zusammen mit anderen technischen Details am Ende zu einer Korrekturfähigkeit von Burstfehlern bis zu einer Länge von 7,5 mm.
Näheres: Standard ECMA-130, http://www.ecma-international.org
(früher: European Computer Manufacturers Association).

9.17. Daten-CDs und DVDs

(a) Bei Daten-CD's ist keine Interpolation möglich. Es gibt verschiedene 'Modes', in denen Daten auf CD's codiert werden.
Entweder nur Fehlererkennung: Dazu wird CRC-Code verwendet mit $g(x) = (x^{16} + x^{15} + x^2 + 1)(x^{16} + x^2 + x + 1)$.
Oder auch zur Fehlerkorrektur: Dann werden sogenannte RS-Produktcodes verwendet, ähnlich wie bei DVD.

(b) DVD verwendet sogenannte RS-Produktcodes; wie CIRC mit unverzögertem Interleaver (9.15b)).
Man verwendet einen verkürzten $\mathcal{C}_1$ $[172, 162, 11]$-RS-Code über $\mathbb{F}_{2^8}$ und einen verkürzten $\mathcal{C}_2$ $[208, 192, 17]$-RS-Code über $\mathbb{F}_{2^8}$, beide mit Erzeugermatrix in Standardform.

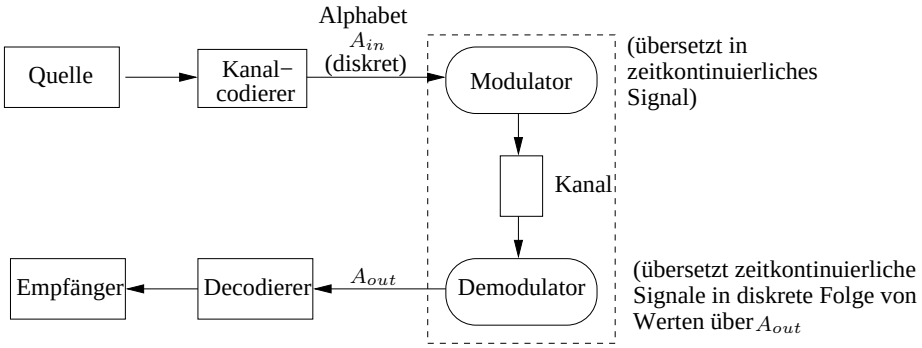


Danach werden allerdings noch weitere Permutationen und Umformungen vor der Speicherung vorgenommen.

Details siehe: http://www.ecma-international.org oder
http://www.tecchannel.de/special/957/index.html

10 Faltungscodes

Faltungscodes (Convolutional Codes) bilden neben den Blockcodes die zweite wichtige Klasse von Codes zur Kanalcodierung (D. Elias, 1955).
Ihr wichtigster Vorteil ist, dass sie besonders geeignet sind zur Verarbeitung von Ausgaben von *Soft-Decision-Demodulatoren*. Was ist damit gemeint? Vergleiche Schaubild zu Beginn der Vorlesung.



Bei *Hard-Decision* gibt der Demodulator wieder Folgen über dem Eingangsalphabet A_{in} aus, das heißt $A_{in} = A_{out}$, das heißt es muss den zeitkontinuierlichen Signalen wieder Folgen von Werten über dem Eingangsalphabet A_{in} zuordnen (Schätzung!). Das ist die typische Situation die wir bei Blockcodes beobachtet haben.
Bei *Soft-Decision* ist A_{out} größer als A_{in} (im Extremfall $\mathbb{R}$). Damit kann der Demodulator dem Decodierer zusätzliche Informationen übermitteln (zum Zustand des Kanals, Sicherheit seiner Entscheidung etc.).
Typischer Fall: $A_{in} = \{0, 1\}$ $|A_{out}| = 8$ (3-Bit-Quantifizierung).

Wir werden auf diesen wichtigen Aspekt allerdings im Folgenden nur am Rande eingehen.
Faltungscodes sind im Wesentlichen dadurch gekennzeichnet, dass der Datenstrom (über $\mathbb{Z}_2$) in kleine Infoblöcke unterteilt wird, die Codierung dieser Infoblöcke aber auch von den vorhergehenden Infoblöcken abhängt.

10.1 Definition.
$K = \mathbb{Z}_2$
Ein *Faltungscode der Rate* $\frac{k}{n}$ wird durch einen Codierer mit Gedächtnis beschrieben:
Datenstrom wird in Infoblöcke der *Länge* k unterteilt: $u_0, u_1, u_2, \ldots$.
Jeder Infoblock wird in einen *Codeblock* der *Länge* n codiert: $c_0, c_1, c_2, \ldots$
Dabei hängt die Codierung von u_r in c_r von u_r und den m vorangegangenen Infoblöcken $u_{r-1}, \ldots, u_{r-m}$ ab: $c_r = f(u_r, u_{r-1}, \ldots, u_{r-m})$.

(Initialisierung: $u_{-1} = \ldots = u_{-m} = 0$)
m heißt *Gedächtnislänge*, $m+1$ *Einflusslänge* des Faltungscodes (falls c_r tatsächlich von u_{r-m} abhängt).
Dabei ist die Codierung *linear*, das heißt die Codebits ergeben sich als Linearkombination der Infobits:
Seien

$$\begin{aligned} u_r &= (u_{r,1}, \ldots, u_{r,k}) \\ \vdots & \qquad \vdots \\ u_{r-m} &= (u_{r-m,1}, \ldots, u_{r-m,k}) \end{aligned}$$

Dann $c_r = (c_{r,1}, \ldots, c_{r,n})$, wobei

$$\begin{aligned} c_{r,i} = \; & g_{1,i,0}u_{r,1} + g_{2,i,0}u_{r,2} + \ldots + g_{k,i,0}u_{r,k} \\ & + g_{1,i,1}u_{r-1,1} + g_{2,i,1}u_{r-1,2} + \ldots + g_{k,i,1}u_{r-1,k} \\ & \quad \vdots \\ & + g_{1,i,m}u_{r-m,1} + g_{2,i,m}u_{r-m,2} + \ldots + g_{k,i,m}u_{r-m,k} \\ & (i = 1, \ldots, n) \end{aligned}$$

Die $g_{v,i,\mu}$ heißen *Generatorkoeffizienten* des Faltungscodes (sie hängen nicht von r ab, bleiben also konstant während der Codierung des Datenstroms).
Üblicherweise: k, n klein (z.B. $k = 1$, $n = 2$ oder 3).

Besonders wichtig:
$\underline{k = 1}$ ($\frac{1}{n}$-Faltungscodes).
Dann sind die $u_r, \ldots, u_{r-m}$ Bits und

$$c_{r,i} = g_{i0}u_r + g_{i1}u_{r-1} + \ldots + g_{im}u_{r-m} \quad (i = 1, \ldots, n)$$

(hier sieht man die Faltung am besten).
(m Gedächtnislänge, falls mindestens ein $g_{im} \neq 0$; ansonsten das größte $m' < m$ mit $g_{im'} \neq 0$ für ein i.)
Wir setzen voraus, dass mindestens ein $g_{i0} \neq 0$. Diese Bedingung stellt sicher, dass man aus einer Codeblockfolge wieder die Infobitfolge rekonstruieren kann:

Angenommen $u_{r-1}, \ldots, u_{r-m}$ sind schon bekannt
(zu Beginn $u_{r-1} = \ldots = u_{r-m} = 0$).
Sei $g_{i,0} \neq 0$, das heißt $g_{i,0} = 1$.
Dann $u_r = c_{r,i} + \sum_{j=1}^{m} g_{ij}u_{r-j}$.

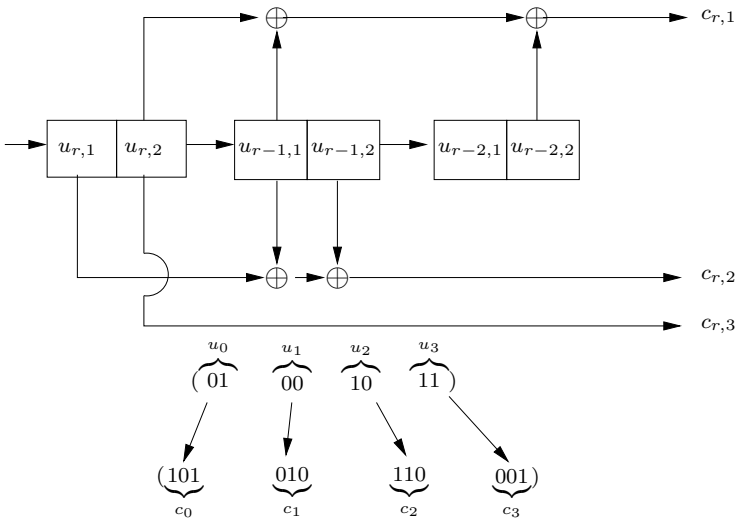
Realisieren lassen sich solche Faltungscodes durch lineare Schieberegister. Die Register enthalten zum Zeitpunkt r die Infoblöcke $u_r, u_{r-1}, \ldots, u_{r-m}$ ($m+1$

Register).
Aus diesen $k \cdot (m+1)$ Bits werden n Linearkombinationen berechnet, die die Komponenten von c_r bilden. Danach werden die Registerinhalte geshiftet, u_{r-m} fällt raus, u_{r+1} wir neu nachgeschoben.
Wir verdeutlichen dies an zwei Beispielen:

10.2 Beispiele.

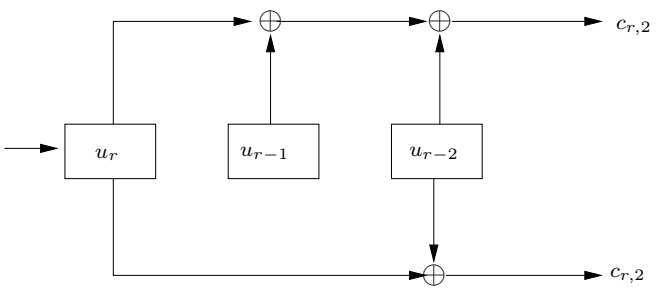
(a) $\frac{2}{3}$-Faltungscode, das heißt $k = 2$, $n = 3$.
$m = 2$

$$\begin{aligned} c_{r,1} &= u_{r,2} + u_{r-1,1} + u_{r-2,2} \\ c_{r,2} &= u_{r,1} + u_{r-1,1} + u_{r-1,2} \\ c_{r,3} &= u_{r,2} \end{aligned}$$



(b) $\frac{1}{2}$-Faltungscode, das heißt $k = 1$, $n = 2$.
$m = 2$ (Standardbeispiel)

$$\begin{aligned} c_{r,1} &= u_r + u_{r-1} + u_{r-2} \\ c_{r,2} &= u_r + u_{r-2} \end{aligned}$$



$$\begin{array}{cccccc} u_0 & u_1 & u_2 & u_3 & u_4 & u_5 \\ (1 & 1 & 0 & 1 & 0 & 0) \end{array} \mapsto (\overbrace{11}^{c_0}\ \overbrace{01}^{c_1}\ \overbrace{01}^{c_2}\ \overbrace{00}^{c_3}\ \overbrace{10}^{c_4}\ \overbrace{11}^{c_5})$$

Bei Faltungscodes ist die Codiervorschrift einfach zu verstehen. Die Menge aller erzeugten Codebitfolgen ist allerdings in der Regel schwer zu beschreiben. Unterschied zu Blockcodes: Algebraische Beschreibung der Menge der Codewörter zuerst, dann geeigneter Codierer (z.B. durch Erzeugermatrix in Standardform).

Ab jetzt :
Nur Faltungscodes mit Rate $\frac{1}{n}$, das heißt $k = 1$.
Für diese Faltungscodes geben wir jetzt eine Beschreibung durch Polynome an:

10.3. Polynombeschreibung von $\frac{1}{n}$-Faltungscodes

$u_0, u_1, u_2, \ldots$ Infobits $\quad (u_{-1}, \ldots, u_{-m} = 0)$

$c_{r,i} = \sum_{j=0}^{m} g_{ij} u_{r-j}$, $\; i = 1, \ldots, n$, $r = 0, 1, 2, \ldots$ Codebits $(*)$

$g_i(D) = \sum_{j=0}^{m} g_{ij} D^j$, $i = 1, \ldots, n$ *Generatorpolynome*

(Die Unbekannte wird in der Literatur üblicherweise mit D bezeichnet)

Infobitreihe: $u(D) = \sum_{r=0}^{\infty} u_r D^r$ (formale Potenzreihe)

(Da die Infobitfolge in der Praxis stets endliche Länge hat, ist $u(D)$ tatsächlich ein Polynom; durch die angegebene Schreibweise schließt man auch unendliche Infobitfolgen ein, was für die Beschreibung zu Vereinfachungen führt.)

Vektor der Codeblockreihen:

$$c(D) = (c_1(D), \ldots, c_n(D))$$

$$c_i(D) = \sum_{r=0}^{\infty} c_{r,i} D^r$$

Der Faltungscodierung $(*)$ entspricht dann Polynom-Multiplikation:

$$\begin{aligned} c_i(D) &= u(D) \cdot g_i(D), \quad i = 1, \ldots, n \\ (c_1(D), \ldots, c_n(D)) &= u(D) \cdot (g_1(D), \ldots, g_n(D)) \\ c(D) &= u(D) \cdot G(D) \end{aligned}$$

$G(D) = (g_1(D), \ldots, g_n(D))$ *Generatorvektor.*

Beachte: Ist $u(D)$ Polynom mit Koeffizienten $u_0, \ldots, u_t$
(also $\mathrm{Grad}(u(D)) \leq t$), so ist $\mathrm{Grad}(c_i(D)) = \mathrm{Grad}(u(D)) + \mathrm{Grad}(g_i)$. $\mathrm{Grad}(c_i(D))$ kann größer als t sein; man benötigt dann nur die ersten $t+1$ Koeffizienten.
Menge aller Codefolgen (der Code)

$$\mathcal{C} = \{u(D) \cdot G(D) \ : \ u(D) = \sum_{r=0}^{\infty} u_r D^r, u_r \in \mathbb{Z}_2\}$$

$\mathcal{C}$ ist (unendlich dimensionaler) $\mathbb{Z}_2$-Vektorraum, das heißt $\mathcal{C}$ ist linear.
Es ist $m = \max_{1 \leq i \leq n} \mathrm{Grad}(g_i(D))$.
Um aus $(c_1(D), \ldots, c_n(D))$ auf die Codewortfolge entsprechend 10.1 zu schließen bildet man $c_1(D^n) + D \cdot c_2(D^n) + \ldots + D^{n-1} \cdot c_n(D^n)$ (Multiplexing).

10.4 Beispiel.
Wir betrachten Beispiel 10.2b).
$c_{r,1} = u_r + u_{r-1} + u_{r-2}$
$c_{r,2} = u_r + u_{r-2}$
$g_1(D) = \underbrace{1 + D + D^2}_{u_0 \ldots u_5}$, $g_2(D) = 1 + D^2$
$u = (\overbrace{110100}) \leftrightarrow u(D) = 1 + D + D^3$

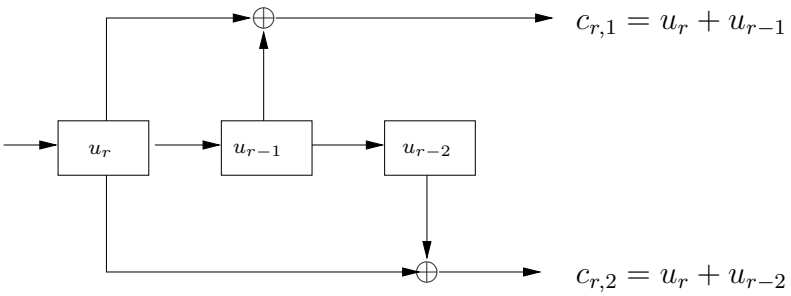
$$\begin{aligned} (c_1(D), c_2(D)) &= ((1 + D + D^3) \cdot (1 + D + D^2), (1 + D + D^3) \cdot (1 + D^2)) \\ &= (1 + D + D^2 + D + D^2 + D^3 + D^3 + D^4 + D^5, \\ &\quad 1 + D^2 + D + D^3 + D^3 + D^5) \\ &= (1 + D^4 + D^5, 1 + D + D^2 + D^5) \\ &\qquad\qquad \updownarrow \\ &\quad (\dot{1}\dot{1}, 0\dot{1}, 0\dot{1}, 00, \dot{1}0, \dot{1}\dot{1}) \\ &\leftrightarrow \quad c_1(D^2) + D \cdot c_2(D^2) = 1 + D^8 + D^{10} + D + D^3 + D^5 + D^{11} \end{aligned}$$

Hätte man $u = (11010)$ gewählt, so hätte man dieselben $u(x)$, $c_i(x)$ erhalten. Man hätte dann beim Codeword hinter 10 abbrechen müssen

10.5 Definition.
Bei einem *terminierten Faltungscode* mit Einflusslänge m werden nach jeweils

L Infobits m Nullen in den Datenstrom eingefügt (*tail bits*).
Nach Verarbeitung dieser $L+m$ Bits sind wieder alle Register mit Null besetzt, wie zu Beginn. Man hat dann einen Blockcode, der L Infobits auf $(L+m)n$ Codebits abbildet, er hat also Rate $\frac{L}{L+m} \cdot \frac{1}{n} (< \frac{1}{n})$. Für große L ist der Verlust in der Rate vernachlässigbar.

10.6 Beispiel.
$\frac{1}{2}$-Code, $n = 2 \quad G(D) = (1 + D, 1 + D^2)$



Angenommen $u = (1, 1, \ldots)$, das heißt $u(D) = 1 + D + D^2 + D^3 + \ldots$
$\rightarrow c = (11, 01, 00, 00, \ldots)$
Das ergibt sich aus

$$\begin{aligned} C(D) &= ((1 + D + D^2 + D^3 + \ldots)(1 + D), (1 + D + D^2 + D^3 + \ldots)(1 + D^2)) \\ &= (1, 1 + D) \\ c_1(D^2) + D \cdot c_2(D^2) &= 1 + D + D^3 \end{aligned}$$

Angenommen $u' = (0, 0, \ldots)$
Dann $c' = (00, 00, 00, \ldots)$.
u, u' unterscheiden sich an allen Stellen, c und c' an nur 3 Stellen. Werden diese 3 Stellen bei der Übertragung gestört, so werden bei der Decodierung unendlich viele Fehler gemacht.

10.7 Definition.
Ein Faltungscode heißt *katastrophal*, wenn es eine Infofolge unendlichen Gewichts (d.h. mit unendlich vielen Einsen) gibt, so dass die erzeugte Codefolge endliches Gewicht hat.

10.8 Satz.
Ein $\frac{1}{n}$-Faltungscode ist genau dann katastrophal, wenn $\mathrm{ggT}(g_1, \ldots, g_n) \neq 1$ *(gebildet im Polynomring über $\mathbb{Z}_2[D]$).*

Beweis:
$\Leftarrow$:
Angenommen $\mathrm{ggT}(g_1, \ldots, g_n) = p$, p Polynom vom Grad ≥ 1,
$G(D) = (p(D)\tilde{g}_1(D), \ldots, p(D)\tilde{g}_n(D))$. Konstanter Term von $p \neq 0$, da mindestens ein $g_i(D)$ konstanten Term $\neq 0$ hat.
Dann kann man zeigen, dass man $\frac{1}{p(D)}$ als Potenzreihe $u(D) = \sum_{r=0}^{\infty} u_r D^r$ schreiben kann (vergleiche Skript 'Kombinatorische Methoden in der Informatik' -Erzeugende Funktionen-).
$u(D)$ ist kein Polynom, denn sonst $1 = u(D)p(D)$, Polynom vom Grad ≥ 1, Widerspruch.
Also ist $(u_0, u_1, \ldots)$ Folge unendlichen Gewichts. Codiert man diese, so ergibt sich

$$c(D) = u(D) \cdot G(D) = (\tilde{g}_1(D), \ldots, \tilde{g}_n(D),$$

also eine Codefolge von endlichem Gewicht.

$\Rightarrow$:
Angenommen $\mathrm{ggT}(g_1, \ldots, g_n) = 1$. Dann existieren Polynome $f_j \in \mathbb{Z}_2[D]$ mit $\sum_{i=1}^{n} f_i g_i = 1$ (erweiterter euklidscher Algorithmus).
Sei $c(D) = (c_1(D), \ldots, c_n(D)) = u(D)(g_1(D), \ldots, g_n(D))$.
Dann ist $\sum_{i=1}^{n} c_i(D) f_i(D) = \sum_{i=1}^{n} u(D) g_i(D) f_i(D) = u(D)$.

Hat also die Codefolge zu $c(D)$ endliches Gewicht, das heißt alle $c_i(D)$ sind Polynome, so ist auch $u(D)$ ein Polynom. Damit ist der Faltungscode nicht katastrophal. □

Wir geben jetzt noch eine Beschreibung von Faltungscodes an, die insbesondere für die Decodierung wichtig ist. Es handelt sich dabei um die sogenannten Trellisdiagramme (trellis (engl.)=Spalier), die von Forney 1973 eingeführt wurden.

10.9 Definition.
Das *Trellisdiagramm* eines Faltungscodes $\mathcal{C}$ (Rate $\frac{1}{n}$) mit Gedächnislänge m (d.h. $c_r = f(u_r, u_{r-1}, \ldots, u_{r-m})$):

(a) Jedes binäre m-Tupel $(u_{r-1}, \ldots, u_{r-m})$ wird als *Zustand* bezeichnet. Die 2^m möglichen Zustände werden mit $\xi_1, \ldots, \xi_{2^m}$ bezeichnet,
$\xi_1 = (0, \ldots, 0)$ *Nullzustand*.
Zum Zeitpunkt r befindet sich im Schieberegister im 1. Register das momentane Infobit u_r und die übrigen Register enthalten einen Zustand $z_r \in \{\xi_1, \ldots, \xi_{2^m}\}$. Für $r = 0$ ist $z_0 = \xi_1$.

(b) Ein (vollständiges) *Trellissegment* besteht aus zwei vertikal angeordneten Punktreihen, jeweils mit 2^m Punkten, die den möglichen Zuständen

entsprechen. Sie werden von oben nach unten angeordnet in der lexikographischen Ordnung ($0 < 1$), aber von rechts nach links gelesen. Also steht $0u_{r-1} \dots u_{r-m}$ immer oberhalb von $1u_{r-1} \dots u_{r-m}$.
Von jedem Zustand $(u_{r-1}, \dots, u_{r-m})$ der ersten vertikalen Punktreihe gehen immer 2 Kanten zu den Zuständen $(0, u_{r-1}, \dots, u_{r-m-1})$ und $(1, u_{r-1}, \dots, u_{r-m-1})$. Diese Kanten sind beschriftet mit dem Codewort, das sich ergibt, wenn die letzten m Infobits gerade $u_{r-1}, \dots, u_{r-m-1}$ waren und dann $u_r = 0$ bzw. $u_r = 1$ das nächste Infobit ist. Die Kante mit $u_r = 0$ geht also immer zu einem höherem Punkt als die mit $u_r = 1$ (Trellissegment beschreibt Übergang von Zeitpunkt r zu Zeitpunkt $r + 1$).

(c) Das Trellisdiagramm ist die Hintereinanderreihung der Trellissegmente zu den Zeitpunkten $r = 0, 1, \dots$.
Zu Beginn hat man noch keine vollständigen Trellissegmente, da die ersten Kanten nur vom Nullzustand ausgehen.
Bei terminierten Faltungscodes gibt es einen entsprechenden Endeffekt, da die Kanten wieder auf den Nullzustand zurückgehen.

(d) Im Trellisdiagramm enden (außer im Anfangsbereich und bei terminierten Faltungscodes im Endbereich) je 2 Kanten: In $(u_r, u_{r-1}, \dots, u_{r-m-1})$ kommen Kanten von $(u_{r-1}, \dots, u_{r-m-1}, 0)$ und $(u_{r-1}, \dots, u_{r-m-1}, 1)$.

10.10 Beispiel.
Wir wählen das Standardbeispiel aus 10.2b).
$\frac{1}{2}$-Faltungscode, $m = 2$
$c_{r,1} = u_r + u_{r-1} + u_{r-2}$
$c_{r,2} = u_r + u_{r-2}$
terminiert mit $L = 5$.
Zum Input $(u_0 \dots, u_6) = (1101000)$ gehört das Codewort (11010100101100) (siehe Abbildung 9).

Jedem Pfad im Trellisdiagramm entspricht Inputfolge (oberer Pfeil 0, unterer Pfeil 1) und zugehörige Codewortfolge (ablesbar an Kanten). Die Trellissegmente außerhalb des Anfangs- und Endbereichs sind identisch. Bei nichtterminierten Faltungsodes erhält man ein einseitig unendliches Trellisdiagramm.

10.11. Viterbi-Decodierung
(Viterbi, 1967) Gegeben sei ein $\frac{1}{n}$-Faltungscode.

(a) Der Viterbi-Algorithmus ist ein Maximum-Likelihood-Decodier-Verfahren, das heißt zu einem empfangenen Wort $v = (v_0, \dots, v_t)$, $v_i \in \mathbb{Z}_2^n$, ist eine Codewortfolge $c = (c_0, \dots, c_t)$, $c_i \in \mathbb{Z}_2^n$, so zu bestimmen, dass $p(v|c)$

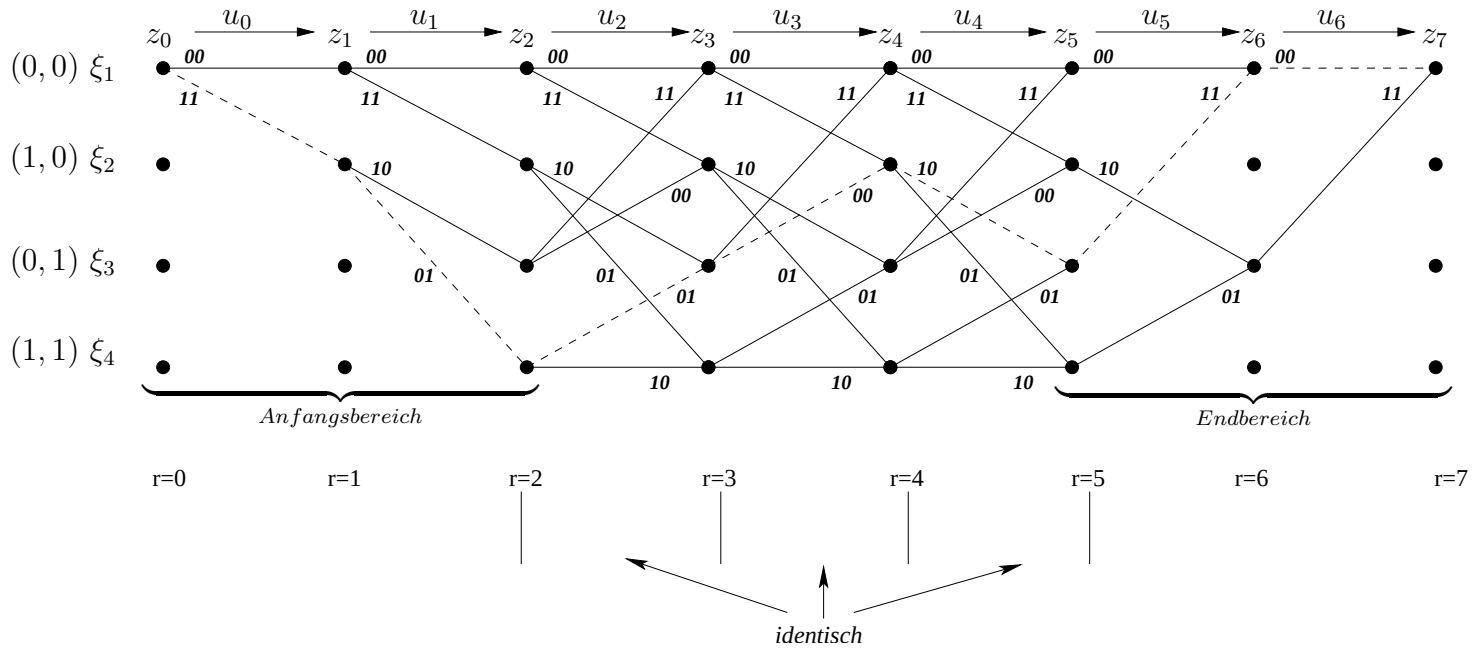


Abbildung 9: Trellisdiagramm für das Beispiel 10.2b). z_i = Zustand zur Zeit i.

maximal wird.

Ist $v_i = (v_{i1}, \dots v_{in})$, $c_i = (c_{i1}, \dots, c_{in})$, so ist

$$p(v|c) = \prod_{i=0}^{t} p(v_i|c_i) = \prod_{i=0}^{t}\prod_{j=1}^{n} \underbrace{p(v_{ij}|c_{ij})}_{\substack{\text{Übergangswahr-}\\ \text{scheinlichkeiten des Kanals}}} \qquad \begin{matrix}\text{(diskreter}\\ \text{gedächnisloser Kanal)}\end{matrix}$$

Der Viterbi-Algorithmus minimiert also die Wahrscheinlichkeit v in eine falsche Codewortfolge zu decodieren (Maximum-Likelihood-Decodierung, vergleiche Kapitel 3).

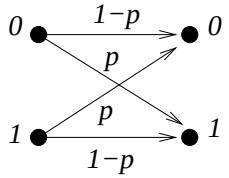
(Demgegenüber gibt es auch Decodierverfahren -prominentestes Beispiel ist der BCJR-Algorithmus (Bahll, Cocke, Jelinek, Raviv)- die die Bit-Fehlerwahrscheinlichkeit in der Infobitfolge minimieren. Hierauf gehen wir nicht ein.)

Häufig arbeitet man mit $\log p(v|c)$ anstelle von $p(v|c)$ (sogenannte Maximum-Log-Likelihood-Decodierung). Da log eine monoton wachsende Funktion ist, ist dies äquivalent mit der Max-Likelihood-Decodierung und bedeutet die Maximierung von

$$\underbrace{\log p(v|c)}_{\substack{=:M(v|c)\\ \underline{\text{Metrik}} \text{ zur Codewortfolge } c\\ \text{(Pfad in Trellisdiagramm)}\\ \underline{\text{Viterbi-Metrik}}}} = \sum_{i=0}^{t} \underbrace{\log p(v_i|c_i)}_{\substack{=:M(v_i|c_i)\\ \underline{\text{Kantenmetrik}}, \text{ denn } c_i\\ \text{entspr. Kante in}\\ \text{Trellisdiagramm}}} = \sum_{i=0}^{t}\sum_{j=1}^{n} \underbrace{\log p(v_{ij}|c_{ij})}_{\substack{=:M(v_{ij}|c_{ij})\\ \underline{\text{Bitmetrik}}}}$$

Kennt man die Übergangswahrscheinlichkeiten des Kanals, so kann man für jeden Weg über $t+1$ Segmente des Trellisdiagramms, also für jede Codewortfolge $c = (c_0, \dots, c_t)$, die Viterbi-Metrik bestimmen. Derjenige mit maximaler Metrik wird dann zur Decodierung verwendet.

Bei einem binären symmetrischen Kanal



$$p < \frac{1}{2}$$

ist

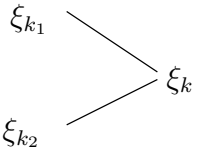
$$\begin{aligned}\log p(v|c) &= d(v,c)\log p + ((N - d(v,c))\log(1-p)) \\ &= d(v,c)\log\frac{p}{1-p} + N\log(1-p)\end{aligned}$$

wobei $N = (t+1) \cdot n$.
Wegen $p < \frac{1}{2}$ ist $\log \frac{p}{1-p} < 0$, so dass Maximum-Likelihood-Decodierung hier bedeutet, den Hamming-Abstand $d(v, c)$ zu minimieren (wie bei den Blockcodes, siehe Kapitel 3).

(b) Der Viterbi-Algorithmus vereinfacht nun die Bestimmung von c in der Weise, dass nicht mehr für alle Pfade im Trellis-Diagramm $M(v|c)$ zu bestimmen ist. Deren Anzahl ist exponentiell in t (Verdoppelung nach jedem Segment). Im Viterbi-Algorithmus ist der Aufwand linear in t. Wir betrachten den Fall terminierter Faltungscodes, das heißt $t = L+m$.

Viterbi-Algorithmus für terminierte Faltungscodes

1. Starte bei $s = m$. Für jeden Zustand (in der vertikalen Reihe m im Trellis-Diagramm) berechne die Metrik aller eintreffenden Pfade und bestimme den mit der höchsten Metrik (Survivor-Pfad). Survivor-Pfad und Metrik für jeden Zustand speichern (bei Mehrdeutigkeiten bei gleicher Metrik, einen Survivor-Pfad auswählen oder beide speichern).
2. $s := s + 1$
   Für jeden Zustand ξ_k (in der vertikalen Reihe s des Trellis-Diagramms) wird folgende Berechnung durchgeführt:

   Bei ξ_k enden 2 Kanten von Zuständen ξ_{k1}, ξ_{k2} der vorangegangenen Reihe. Berechne: Metrik Survivor-Pfad bei ξ_{k_i} + Kantenmetrik $\overline{\xi_{k_i}\xi_k}$. Speichere die größere der beiden Summen ab und speichere den entsprechenden Pfad als Survivor-Pfad bei ξ_k (Bei Mehrdeutigkeiten wähle einen der beiden aus oder speichere beide).
   Eliminiere alle früheren Kanten, die nicht mehr zu Survivor-Pfaden gehören.
3. Falls $s < L + m$ wiederhole 2), sonst stop.

(c) Nun gilt:

Satz.
Der letzte Survivor-Pfad (entsprechend $c = (c_0, \ldots, c_{L+m-1})$) des Viterbi-Algorithmus ist der Maximum-Likelihood-Pfad (ML-Pfad), das heißt $M(v|c) \geq M(v|c')$ für alle Pfade $c' = (c'_0, \ldots, c'_{L+m-1})$.

Beweis:
Angenommen der ML-Pfad wurde zum Zeitpunkt $s < L + m$ eliminiert (siehe Abbildung 10 und dortige Bezeichnungen). Metrik von $P_1 <$ Metrik von P_2

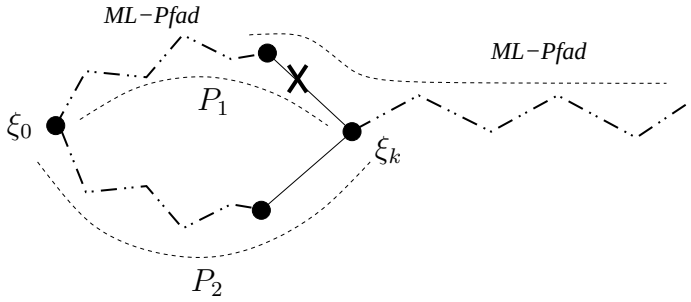


Abbildung 10: Beweisskizze für Satz aus (c)

Dann: Metrik von P_2 + Metrik des Restes des ML-Pfads ab $\xi_k >$ Metrik ML-Pfad. Widerspruch. □

(d) Da die Metriken in der Regel nicht ganzzahlig sind, wird häufig statt der Bit-Metrik $\log p(v_i|c_i)$ der Ausdruck $a[\log p(v_i|c_i) + b]$, $a \in \mathbb{R}_{>0}$, $b \in \mathbb{R}$ verwendet. Dann wird $\sum \log p(v_{ij}|c_{ij})$ genau dann maximiert, wenn $\sum a[\log(p(v_{ij}|c_{ij})) + b]$ maximiert wird.
Man wählt b so, dass der kleinste Wert der Metrik 0 wird und dann a so, dass alle Werte (annähernd) ganzzahlig werden. Schätzung des Algorithmus ist dann etwas suboptimal, aber im Allgemeinen ist die Verschlechterung minimal.

(e) Beim binären symmetrischen Kanal kann man zeigen:

$$M(v|c) \text{ maximal} \iff \underbrace{\sum_{i=0}^{t}\sum_{j=1}^{n} v_{ij} \cdot c_{ij}}_{=:v\cdot c} \in \mathbb{N}_0 \text{ maximal}$$

(Summieren in $\mathbb{Z}$, nicht in $\mathbb{Z}_2$)
Also wählt man dort als Metrik gerade $v \cdot c$.

10.12 Beispiel.
$\frac{1}{2}$ terminierter Faltungscode ($L = 5$) mit Trellisdiagramm aus Beispiel 10.10.
Empfangene Folge: $v = (1100110110110)$; Metrik $M(v|c) = v \cdot c$.
Viterbi-Decodierung siehe Abbildung 11.

10.13 Beispiel.
Wir nehmen jetzt an, dass wir einen Kanal mit Input-Alphabet $\{0, 1\}$ und Output-Alphabet $\{0_1, 0_2, 1_1, 1_2\}$ haben (Soft-Decision-Modulator).

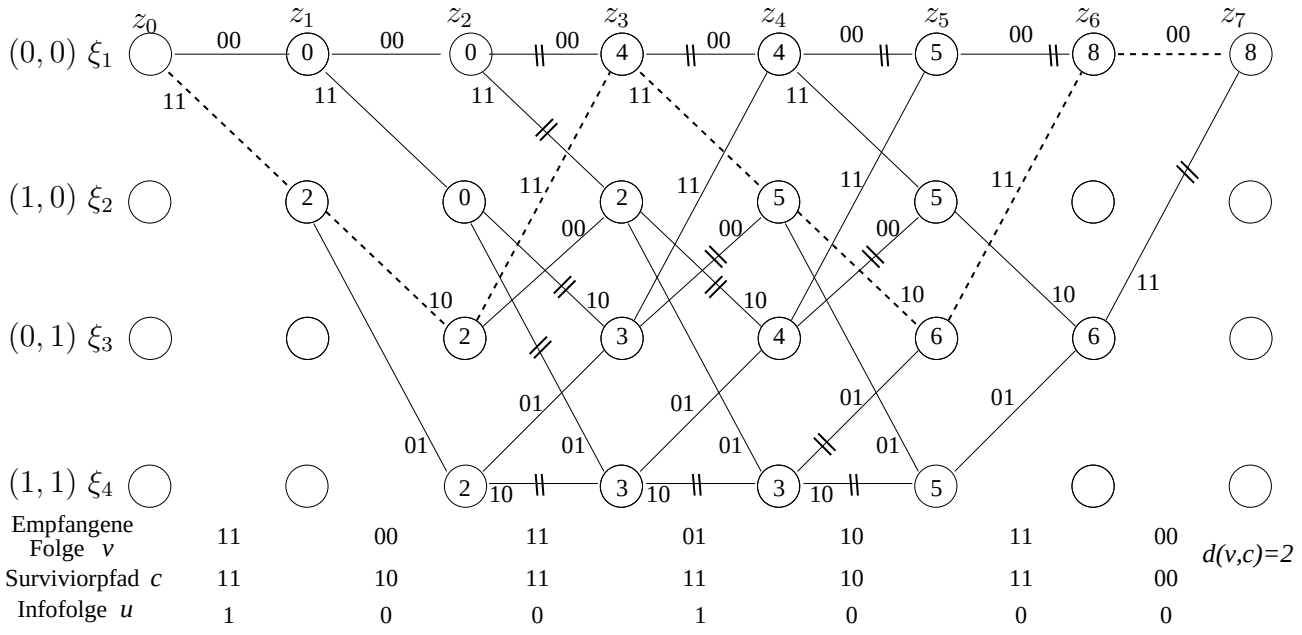
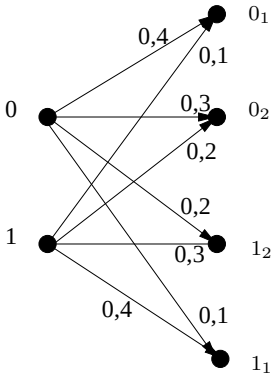


Abbildung 11: Viterbi-Decodierung für das Beispiel 10.12.

0_1 = sehr wahrscheinlich 0
0_2 = eher 0
1_2 = eher 1
1_1 = sehr wahrscheinlich 1

Übergangswahrscheinlichkeiten:



Logarithmen zur Basis 10

v_i \ c_i	0_1	0_2	1_2	1_1
0	$-0,4$	$-0,52$	$-0,7$	$-1,0$
1	$-1,0$	$-0,7$	$-0,52$	$-0,4$

Mache Metrik ganzzahlig entsprechend 10.11d): $b = 1$, $a = 16,7$

v_i \ c_i	0_1	0_2	1_2	1_1
0	10	8	5	0
1	0	5	8	10

Bit-Metrik (Kanten-, Pfad-Metriken = Summe der Bit-Metriken)

Selber $\frac{1}{2}$-Faltungscode wie in 10.12.
Empfangene Folge ($1_1 1_2\ 0_1 0_1\ 1_2 1_2\ 0_1 1_2\ 1_2 0_1\ 1_1 1_2\ 0_1 0_1$).
Viterbi-Decodierung siehe Abbildung 12.
Einträge in den Knoten: Metrik des jeweiligen Survivior-Pfades.
Survivior-Pfad bei $r = 7$: $c = 11010100101100$.
Zugehörige Infofolge: 1101000.
(Andere Folge als die aus Beispiel 10.12; die dortige ist genau diejenige, die bei Hard-Decision-Demodulation entsteht.)

10.14. Viterbi-Decodierung für nicht-terminierte Faltungscodes
Wird eine Infobitfolge $u_0, u_1, \ldots, u_t$ mit einem nicht-terminierten Faltungscode codiert und dann $v_0, \ldots, v_t \in \mathbb{Z}_2^n$ empfangen, so verläuft die Viterbi-Decodierung wie im terminierten Fall. Man wird allerdings nicht wieder am Ende in den Nullzustand gelangen, sondern zum Zeitpunkt t für alle 2^m Zustände eine Metrik mit zugehörendem Survivor-Pfad erhalten. Man wählt dann zur Decodierung den Zustand mit maximaler Metrik und den zugehörigen Pfad.
Man stellt dabei fest, dass in der Regel die Survivor-Pfade der 2^m Zustände zum Zeitpunkt t bei Zurückverfolgung ab einem Zeitpunkt $t - r$ alle den gleichen Anfangspfad besitzen. Ein solches r nennt man die *Rückgrifftiefe* des Faltungscodes. (Ein instruktives Beispiel findet man z.B. in Friedrichs, Kanalcodierung, S.280/281.)
Die Existenz eines solchen r impliziert, dass man zur Decodierung von $v_0, \ldots v_t$

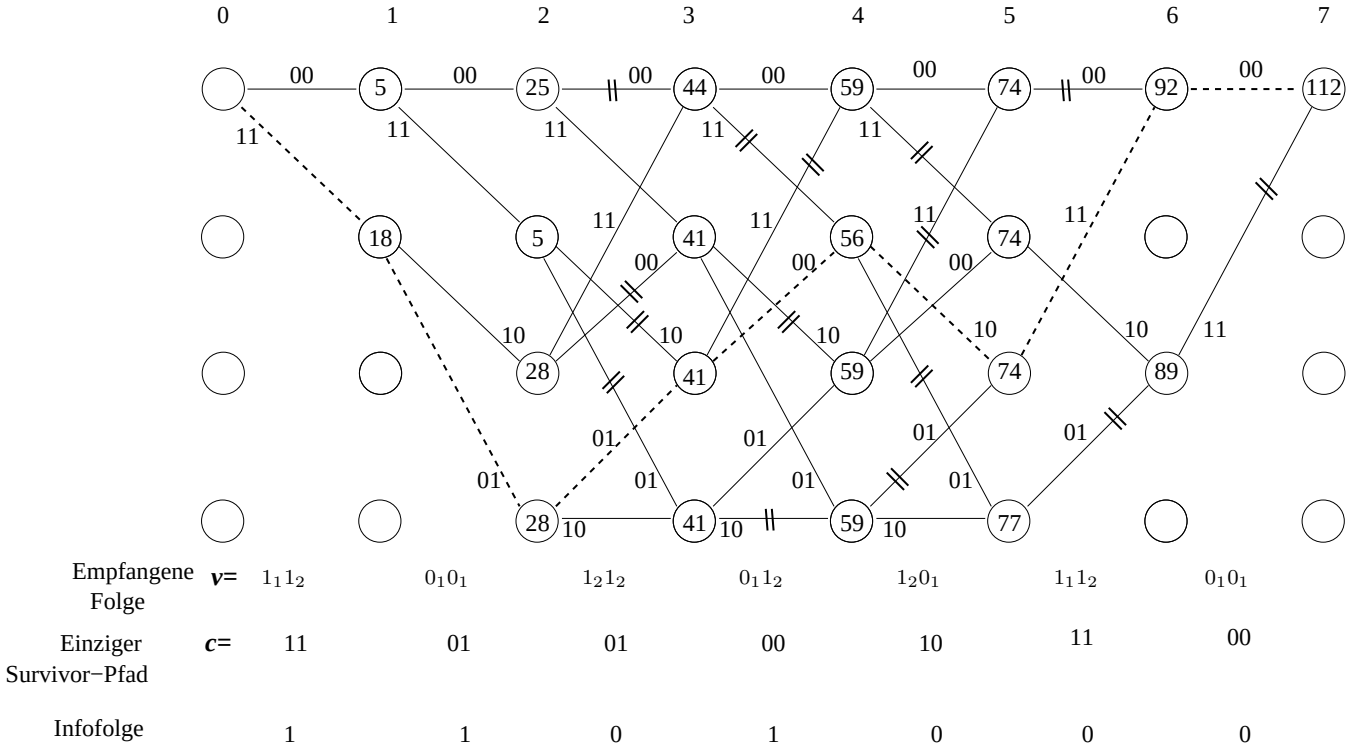


Abbildung 12: Viterbi-Decodierung für das Beispiel 10.13.

nicht bis zum Zeitpunkt t warten muss, sondern bei Zeitpunkt r schon Bit u_0, zum Zeitpunkt $r+1$ Bit u_1 etc. decodieren kann. Damit hat man ein fast-synchrones Decodierverfahren.
Als Faustregel wird $r \approx 5 \cdot m$ gewählt.

10.15 Bemerkung.
Es ist leicht einzusehen, dass der Viterbi-Algorithmus bei nicht-korrekter Decodierung immer ganze Stücke falscher Codewörter liefert; denn wenn ein Codewort einmal falsch ist, werden aufgrund der Faltungscodedecodierug auch die nächsten Codewörter dadurch beeinflusst (Für die zugehörige Infobitfolge muss der Unterschied aber nicht groß sein. Das sieht man auch an den Beispiel 10.12 und 10.13.)
Daher produziert die Viterbi-Decodierung vor allem Bündelfehler und dies macht Faltungscodes als innere Codes in Zusammenhang mit Interleavingverfahren (vgl. Kapitel 9) besonders geeignet. In dieser Weise werden Faltungscodes etwa im Mobilfunk eingesetzt, z.B. im GSM- bzw. UMTS-Standard.
Weiterführende Informationen über Faltungscodes findet man z.B. in

Friedrichs, Kanalcodierung und
Lin, Costello Jr., Error Control Coding.