Codierungstheorie

Skript zur Vorlesung im SS 2010

Diplom-und Masterstudiengang

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

Inhaltsverzeichnis

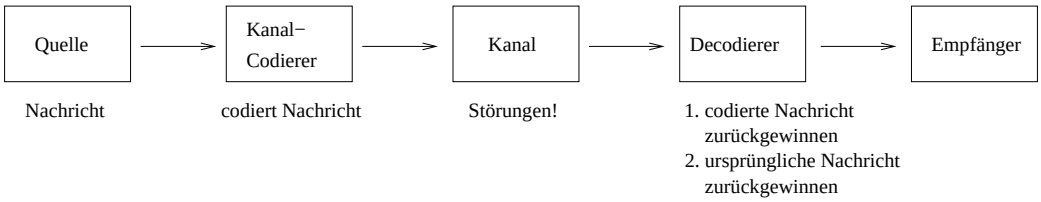
Literatur

1. E. R. Berlekamp, *Algebraic Coding Theory*. Aegean Park Press, 1984.
2. B. Friedrichs, *Kanalcodierung*. Springer, 1995.
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. W. Lütkebohmert, *Codierungstheorie*. Vieweg, 2003.
6. F. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. North Holland, 1977.
7. W. W. Peterson, *Error-Correcting-Codes*. John Wiley & Sons, 1962
8. V. S. Pless and W. C. Huffman (editors), *Handbook of Coding Theory I, II*. Elsevier, 1998.
9. T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge University Press, 2008.
10. S. Roman, *Coding and Information Theory*. Springer, 1992.
11. R.-H. Schulz, *Codierungstheorie*. Vieweg, 2003.
12. J. van Lint, *Introduction to Coding Theory*. Springer, 1999.
13. S. A. Vanstone and P. C. Oorschot, *An Introduction to Error-Correcting Codes with Applications*. Kluwer, 1989.
14. W. Willems, *Codierungstheorie*. De Gruyter, 1999.
15. W. Willems, *Codierungstheorie und Kryptographie*. Birkhäuser, 2008.
16. S. Xambó-Descamps, *Block Error-Correcting Codes*. Springer, 2003.

Einführung

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

Situation (vereinfacht):



Ziele bei der Codierung:

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

2. Der Aufwand zur Codierung und Decodierung soll möglichst gering sein.

Grundprinzip: Hinzufügen von Redundanz

Zwei Hauptaufgaben:

- Konstruktion geeigneter Codes
  Anforderung: Gute Fehlererkennungs, -korrektureigenschaften mit möglichst geringer Redundanz
- Konstruktion von Codierern, Decodierern
  Anforderung: Effizienz

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

Prinzipien bei der Kanalcodierung:

FEC-Verfahren (Forward Error Correction)
Aufgetretene Fehler werden erkannt und korrigiert.
Vorteil: keine Verzögerung bei der Übertragung
Nachteil: ggf. 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
Nachteil: ggf. erhebliche Verzögerung bei wiederholter Übertragung

1 Grundbegriffe und 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.
Zuvor eine Bemerkung zur Schreibweise: Wir werden Wörter der Länge n über einem Alphabet A sowohl in der Form $a_1a_2\ldots a_n$ als auch als Zeilenvektoren $(a_1, a_2, \ldots, a_n)$ schreiben. Darüberhinaus kommt auch die umgekehrte Reihenfolge der Indizes oder ein Laufbereich von 0 bis $n-1$ vor. Die Länge eines Wortes $w = (a_1, a_2, \ldots, a_n)$ bezeichnen wir mit $\ell(w)(= n)$.

1.1 Beispiele (Prüfzifferncodes)**.**
Die Nachrichten bestehen aus Wörtern (Blöcken) der Länge n. An diese wird zur Codierung eine Prüfziffer angefügt.

(a) **Parity-Check-Code**
Die Nachrichten seien z.B. 00, 01, 10, 11 über dem Alphabet $A = \{0, 1\}$. Sie werden über demselben Alphabet auf folgende Weise codiert:

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

Eine Nachricht (x_1, x_2) wird also zu $(x_1, x_2, (x_1 + x_2) \bmod 2)$ codiert.
Der ASCII-Code ist von diesem Typ: 128 Zeichen werden mit 8 Bits codiert, wobei das achte Bit ein Parity-Check-Bit ist, d.h.

$$x_8 = (x_1 + \ldots + x_7) \bmod 2.$$

(b) **alter ISBN-Code**
Die frühere *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).
Uncodierte Wörter sind über dem Alphabet $B = \{0, \ldots, 9\}$ und Codewörter sind über dem Alphabet $A = \{0, 1, \ldots, 9, \mathrm{X}\}$ gebildet. X steht für 10. Die Prüfziffer c_1 wird so gewählt, dass für $c_{10}c_9...c_1$ gilt:

$$\sum_{k=1}^{10} kc_k \equiv 0 \pmod{11}$$

Ist dabei $c_1 = 10$, so wird $c_1 = \mathrm{X}$ gesetzt.

Der Code ist 1-Fehler-erkennend. Durch die gewichtete Quersumme wird auch sichergestellt, dass die Vertauschung zweier Ziffern erkannt wird.

(c) **EAN-13-Prüfzifferncode**
Die *Europäische Artikelnummer* ist ein 13-stelliger Code, der sich auf vielen Produktverpackungen befindet. Nachrichten und Codewörter sind aus dem Alphabet $A = B = \{0, \ldots, 9\}$. Sei $c_{13}...c_1$ ein Codewort. $c_{13}...c_2$ beschreibt Herstellungsland, Hersteller und Produkt. c_1 wird so gewählt, dass gilt:

$$c_{13} + 3c_{12} + c_{11} + \ldots 3c_2 + c_1 \equiv 0 \pmod{10}.$$

Seit 1.1.07 wird statt des in (b) beschriebenen ISBN-10-Codes ISBN-13 verwendet, der mit EAN-13 kompatibel ist. Der Präfix 978 bzw. 979 „Buchland“ wird der 10-stelligen ISBN-10 vorangestellt. Anschließend wird die Prüfziffer gemäß EAN-13 neu berechnet.

(d) **Seriennummern der EURO-Scheine**
Die Seriennummer besteht aus einem führenden Buchstaben (nicht alle Buchstaben A-Z kommen vor, A-I, O, Q nicht; X steht für Deutschland) und einer 11-stelligen Zahl. Die letzte Ziffer ist eine Prüfziffer. Diese wird wie folgt bestimmt:
Codiere den führenden Buchstaben in eine 2-stellige Dezimalzahl entsprechend seiner Position im Alphabet. Das liefert $c_{13} \ldots c_1$. c_1 wird dabei so gewählt, dass gilt:

$$\sum_{i=1}^{13} c_1 \equiv 8 \pmod{9}.$$

(Also könnte man 9 auch durch 0 ersetzten und in $\mathbb{Z}_9$ rechnen.) $9 \leftrightarrow 0$ zeigt, dass nicht alle Einzelfehler erkannt werden. Klar: keine Vertauschungen werden erkannt.
Vorteil: Modulo-9-Rechnung entspricht (iterierte) Quersummenbildung bis zum 1-stelligen Ergebnis.
Warum gerade 8, ist unklar.
(Seriennummern der DM-Scheine waren subtiler aufgebaut.)

1.2 Beispiele.

(a) 1.1(a): Angenommen 111 wird empfangen. Klar: 1 oder 3 Fehler sind aufgetreten. Ein Fehler könnte an jeder beliebigen Stelle entstanden sein.

(b) Codiere stattdessen:

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

3-facher Wiederholungscode.
Zwei verschiedene Codewörter unterscheiden sich an mindestens 3 Stellen. Tritt 1 Fehler auf, so ist das nächstgelegene Codewort das gesendete. Es handelt sich also um einen 1-Fehler-korrigierenden Code.

(c) Es geht mit weniger Redundanz als in (b):

$$\begin{aligned} 00 &\to 00000 \\ 01 &\to 01101 \\ 10 &\to 10110 \\ 11 &\to 11011 \end{aligned}$$

Wieder sind zwei verschiedene Codewörter an mindestens 3 Stellen unterschiedlich, aber die Redundanz ist geringer als in (b).

Mit „Fehler“ ist hier die *Änderung* eines Bits gemeint. Es könnte aber auch sein, dass eine Stelle nicht mehr als 0 oder 1 identifiziert werden kann: *Auslöschung*. (Wir betrachten zunächst nur Änderungen und keine Auslöschungen.)

In Beispiel 1.1, 1.2 wurden zur Codierung Nachrichten fester Länge k in Codewörter fester (größerer) Länge n codiert. Wichtig: Nicht alle Wörter der Länge n sind Codewörter (das wird ausgenutzt zum Erkennen und Korrigieren von Fehlern). Das führt zu folgender Definition:

1.3 Definition.
Sei A eine endliche Menge (Alphabet), $n \in \mathbb{N}$. 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“).

Ist $|\mathcal{C}| = m$, so lassen sich Alphabete B der Größe m codieren. B kann auch aus Blöcken über einem kleineren Alphabet (z.B. $\{0, 1\}$) bestehen (wie in den Beispielen in 1.2).
Folgen von Elementen aus B (d.h. Wörter über B) werden dann in Folgen von Codewörtern codiert.
Die Zuordnung: „Element von $B \mapsto$ Codewort“ ist die Codierungsfunktion. (Sie kann auch kontextabhängig sein.)

In 1.2 hatten wir schon von Abstand gesprochen. Das präzisieren wir:

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

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

$\mathrm{d}(a, b)$ heißt *Hamming-Abstand* von a und b.

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

1.5 Satz.
Sei A ein Alphabet und $a, b, c \in A^n$ beliebig. Dann gilt:

(1) $0 \leq \mathrm{d}(a, b) \leq n$

(2) $\mathrm{d}(a, b) = 0 \Leftrightarrow a = b$

(3) $\mathrm{d}(a, b) = \mathrm{d}(b, a)$

(4) $\mathrm{d}(a, b) \leq \mathrm{d}(a, c) + \mathrm{d}(c, b)$ *(Dreiecksungleichung)*

Also ist d *eine Metrik.*
Ist A eine kommutative Gruppe bezgl. + (z.B. $A = \mathbb{Z}_n$), so:

(5) $\mathrm{d}(a + c, b + c) = \mathrm{d}(a, b)$ *(Translationsinvarianz)*

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

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

1.6 Definition.

(a) *Hamming-Decodierung* für einen Blockcode $\mathcal{C} \subseteq A^n$:
Wird $y \in A^n$ empfangen, so wird y zu einem $x' \in \mathcal{C}$ decodiert mit

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

(x' ist im Allgemeinen nicht eindeutig bestimmt!)
(Wann diese Decodierung sinnvoll ist, werden wir uns in Kürze überlegen.)

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

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

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

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

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

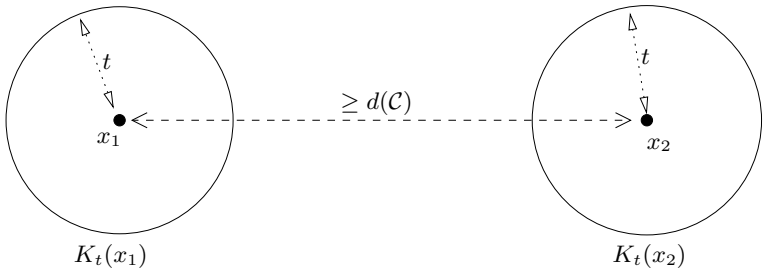
$$\mathrm{d}(\mathcal{C}) \geq t + 1.$$

Zur Bezeichnung in 1.6(c):

- Ist $\mathrm{d}(\mathcal{C}) \geq 2t + 1$ und ist $x \in \mathcal{C}$, so liegt in der „Kugel“

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

vom Radius t um x genau ein Codewort, nämlich x (da $\mathrm{d}(\mathcal{C}) > t$).
Treten also bei der Übertragung von x höchstens t Fehler auf, so liegt das empfangene Wort in $K_t(x)$, aber in keiner Kugel $K_t(x')$, $x' \in \mathcal{C}, x' \neq x$, d.h. $\mathrm{d}(y, x') > \mathrm{d}(y, x)$ für alle $x' \in \mathcal{C}, x' \neq x$.
Der Grund dafür ist, dass die Kugeln von Radius t um Codewörter disjunkt sind. Angenommen $y \in K_t(x_1) \cap K_t(x_2)$. Dann gilt $\mathrm{d}(x_1, x_2) \leq \mathrm{d}(x_1, y) + \mathrm{d}(y, x_2) \leq 2t < \mathrm{d}(\mathcal{C})$. Also liefert bei maximal t aufgetretenen Fehlern Hamming-Decodierung das korrekte gesendete Codewort x. (Hamming-Decodierung kann aber inkorrekt sein, falls mehr als t Fehler aufgetreten sind.)



- Ist $\mathrm{d}(\mathcal{C}) \geq t+1$, und sind bei der Übertragung eines Codewortes maximal t Fehler (und mindestens einer) aufgetreten, so ist das empfangene Wort kein Codewort. Es wird erkannt, dass Fehler aufgetreten sind.

1.7 Beispiele.

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

(b) Code aus Beispiel 1.2(c): $\mathrm{d}(\mathcal{C}) = 3$, $n = 5$. $\mathcal{C}$ ist 1-Fehler-korrigierend, 2-Fehler-erkennend.

(c) Sei $\mathcal{C} = \{\underbrace{000000}_{x_1}, \underbrace{011010}_{x_2}, \underbrace{101101}_{x_3}, \underbrace{110110}_{x_4}\}$. Dann ist $\mathrm{d}(\mathcal{C}) = 3$. $\mathcal{C}$ ist also 1-Fehler-korrigierend.
Angenommen $x_1 = 000000$ wird gesendet und $y = 000011$ wird empfangen (2 Fehler).
$\mathrm{d}(x_i, y) \geq 3$ für alle $i = 2, 3, 4$. Hamming-Decodierung liefert also x_1.
Angenommen $x_1 = 000000$ wird gesendet und $y = 100001$ wird empfangen.
$\mathrm{d}(x_1, y) = \mathrm{d}(x_2, y) = 2$, $\mathrm{d}(x_3, y), \mathrm{d}(x_4, y) \geq 3$. Hamming-Decodierung ist also nicht eindeutig.
Angenommen $x_1 = 000000$ wird gesendet und $y = 011000$ wird empfangen.
$\mathrm{d}(x_2, y) = 1$. Es wird also falsch zu x_2 decodiert.

1.8 Definition.
Sei $\mathcal{C}$ ein Blockcode in A^n mit $|A| = q$.

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

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

Beachte:
$\log_q(|A^n|) = n$. Die Informationsrate gibt an, welcher Anteil eines Codewortes reine Information ist.
Einheit: Infosymbol/Codesymbol.
(Beachte: Ohne Codierung könnte man die $|\mathcal{C}|$ Codewörter jeweils mit $\lceil \log_q(|\mathcal{C}|) \rceil$ Zeichen aus A komprimiert darstellen.)
$\frac{1}{R}$ kannn man auch als Verzögerungsfaktor betrachten, der durch die Codierung der Infosymbole verursacht wird.
Manchmal ist es notwendig, die Rate auf Bits statt auf Symbole zu beziehen. Das führt zur *binären Rate*

$$R_b = R \cdot \log_2(q) \quad \text{Infobit/Codesymbol.}$$

1.9 Beispiele.

(a) m-facher Wiederholungscode, $|A| = q$, $n = k \cdot m$, $|\mathcal{C}| = q^k$.
$R = \frac{k}{k \cdot m} = \frac{1}{m}$ ($q = 2$: Ein Infobit auf m Bits)

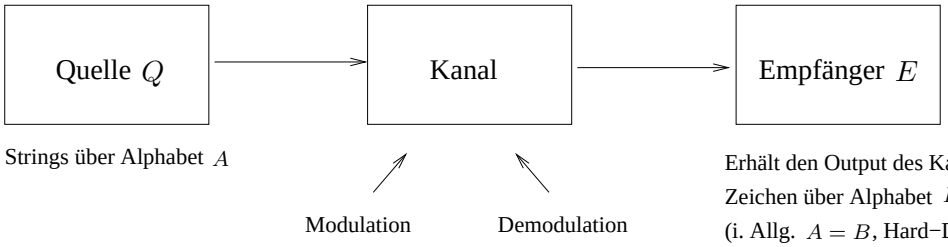
(b) Code aus Beispiel 1.2(c): 4 Codewörter über $\{0, 1\}$ der Länge 5.
$R = \frac{2}{5}$

(c) Parity-Check-Code der Länge n: $R = \frac{n-1}{n}$.

Ziel der Codierungstheorie:

Finde Codes mit hoher Rate und kleiner Decodierfehlerwahrscheinlichkeit (bei Hamming-Decodierung heißt das großer Minimalabstand).
Das geht im Prinzip: Satz von Shannon (s. folgendes Kapitel).

2 Grundbegriffe der Informationstheorie und der Kanalcodierungssatz von Shannon



Quelle:

2.1 Voraussetzung.
Die Quelle Q gebe die Zeichen aus dem Alphabet A mit der Wahrscheinlichkeit $p_Q(a)$, $a \in A$, aus. Also:

$$0 \leq p_Q(a) \leq 1, \quad \sum_{a \in A} p_Q(a) = 1.$$

(Gedächtnislose Quelle; Wahrscheinlichkeit von a ist nicht von dem vorangegangenen Zeichen abhängig. Idealisierung!)

2.2 Definition.
Der *Informationsgehalt* eines von Q ausgegebenen Zeichens $a \in A$ ist

$$I_Q(a) := \log_2 \left(\frac{1}{p_Q(a)} \right) = -\log_2 \left(p_Q(a) \right) \in \mathbb{R}_{\geq 0} \cup \{\infty\}.$$

Einheit: Bit
(Im Folgenden: $\log = \log_2$)

Interpretation:

Je größer $p_Q(a)$, desto geringer ist die Überraschung, wenn a auftritt.
Der Informationsgehalt stellt ein Maß für die Überraschung dar (Beispiel: Erdbeben, kein Erdbeben)

Dass man als Maß gerade $\log\left(\frac{1}{p_Q(a)}\right)$ wählt, hat folgenden Grund: Man will, dass der Informationsgehalt nur von der Wahrscheinlichkeit von a abhängt und dass gilt:

(1) Für zwei unabhängig voneinander ausgegebene Zeichen soll sich der Informationsgehalt addieren.

(2) Die Funktion soll stetig sein und es soll $I(a) < I(b)$ gelten, falls $p_Q(a) > p_Q(b)$ gilt.

(3) Geeignete Normierung

Das führt auf $\log(\frac{1}{p_Q(a)})$.

2.3 Definition.
Die *Entropie* einer Quelle Q ist definiert durch

$$H(Q) := \sum_{a\in A} p_Q(a) I_Q(a) = -\sum_{a\in A} p_Q(a)\log(p_Q(a)) \geq 0$$

(Ist $p_Q(a) = 0$, so definiert man $p_Q(a)I_Q(a) = 0$; das ist sinnvoll, da $\lim_{x\to 0} x \cdot \log\left(\frac{1}{x}\right) = 0$.)

Interpretation:

Die Entropie ist der mittlere Informationsgehalt pro Zeichen einer Quelle Q. Sie wird in Bit/Symbol gemessen.
Sie lässt sich auch so interpretieren: Wieviele Bits werden im Durchschnitt $\left(\sum_{a\in A} p_Q(a) \cdot l(\text{Codewort von } a)\right)$ zur eindeutigen Präfixcodierung des Alphabets A mindestens benötigt.
[Shannon's Quellcodierungssatz → Datenkompression]

2.4 Beispiele.
$A = \{a_1, \ldots, a_q\}$

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

(b) $p_Q(a_i) = \frac{1}{q}$, $1 \leq i \leq q$. Dann $I_Q(a_i) = \log(q)$, $H(Q) = \log(q)$.
($q = 2$: $H(Q) = 1$)

(c) $|A| = 8$

- Ist $p_Q(a_i) = \dfrac{1}{8}$ für alle i, so $H(Q) = 3$ nach (b).
  Kompakteste Präfixcodierung von A benötigt 3 Bits pro Symbol.

- $p_Q(a_1) = p_Q(a_2) = \frac{1}{4}$, $p_Q(a_3) = p_Q(a_4) = \frac{1}{8}$, $p_Q(a_5) = \ldots = p_Q(a_8) = \frac{1}{16}$
  $H(Q) = 2 \cdot \frac{1}{4} \cdot 2 + 2 \cdot \frac{1}{8} \cdot 3 + 4 \cdot \frac{1}{16} \cdot 4 = 2,75.$

Mögliche Codierung (z.B. Huffman):

$$\begin{array}{lcl} a_1 & \to & 11 \\ a_2 & \to & 10 \\ a_3 & \to & 011 \\ a_4 & \to & 010 \\ a_5 & \to & 0011 \\ a_6 & \to & 0010 \\ a_7 & \to & 0001 \\ a_8 & \to & 0000 \end{array}$$

2.5 Bemerkung.
Für alle Wahrscheinlichkeitsverteilungen p_Q gilt: $0 \leq H(Q) \leq \log(q)$ ($|A| = q$). Das Maximum wird bei Gleichverteilung angenommen.

Kanal:

2.6 Voraussetzung.
Der Kanal sei ein *diskreter gedächnisloser Kanal*.
Das bedeutet: Es gibt feste Wahrscheinlichkeiten $p(b|a)$, wobei $p(b|a)$ die Wahrscheinlichkeit ist, dass $b \in B$ empfangen wird, falls $a \in A$ gesendet wird. $p(b|a)$ ist nicht abhängig von zuvor gesendeten Signalen.
Die $p(b|a)$ sind die *Übergangswahrscheinlichkeiten* des Kanals (bedingte Wahrscheinlichkeiten).

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

2.7 Beispiele.

(a) $A = B$, $p(b|a) = \begin{cases} 0 \text{ , } b \neq a \\ 1 \text{ , } b = a \end{cases}$
Ungestörter Kanal

(b) $p(b|a_1) = p(b|a_2)$ für alle $a_1, a_2 \in A$, $b \in B$
D.h. die 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
(für unsere Anwendungen der wichtigste Fall).

Empfänger:

2.8 Definition.
Die Wahrscheinlichkeitsverteilung p_E beim Empfänger E (*Outputverteilung*) ist gegeben durch

$$p_E(b) = \sum_{a \in A} p(b|a) p_Q(a), \quad b \in B.$$

($p_E(b)$ ist die Wahrscheinlichkeit, dass b empfangen wird, wenn ein Symbol aus A von Q generiert und durch den Kanal geschickt wurde.)

Outputentropie:

$$H(E) = -\sum_{b \in B} p_E(b) \log(p_E(b)).$$

2.9 Beispiele.

(a) $A = B$, ungestörter Kanal, so $H(E) = H(Q)$, denn $p_E(a) = p_Q(a)$.

(b) Total gestörter Kanal: $p_E(b) = p(b|a)$ für alle $a \in A$, $b \in B$.

Wichtige Frage:

Wieviel Information transportiert ein Kanal, der von einer Quelle Q gespeist wird, im Mittel pro eingegebenem Zeichen?
Einfluss: Quelle und Kanal (Rauschquelle) tragen beide zu $H(E)$ bei.
Welcher Anteil von $H(E)$ wird durch Kanalstörungen verursacht?

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

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

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

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

$$H(E|Q) = \sum_{a\in A} p_Q(a) H(E|a) = \sum_{a\in A}\sum_{b\in B} p_Q(a)p(b|a)\log\left(\frac{1}{p(b|a)}\right).$$

Interpretation:

$H(E|Q)$ ist der Anteil der Entropie von E, der im Mittel (bei Sendung eines Symbols) vom Kanal (also der Rauschquelle) beigesteuert wird.
Leicht zu sehen: $H(E|Q) \leq H(E)$.

2.12 Beispiele.

(a) Ungestörter Kanal ($A = B$):
$H(E|Q) = \sum_{a\in A} p_Q(a)\log(1) = 0$.

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

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

2.13 Definition.
Die *mittlere Transinformation* eines Kanals mit Ausgang E und Quelle Q ist

$$I(Q,E) := H(E) - H(E|Q).$$

Bemerkung.
Man kann leicht zeigen: $I(Q,E) \leq H(Q)$.

Interpretation:

$I(Q,E)$ gibt an, wieviel nutzbare Information pro gesendetem Symbol im Mittel über den Kanal transportiert werden kann (in Abhängigkeit einer gegebenen Wahrscheinlichkeitsverteilung der Quelle Q).

2.14 Definition.
Die *Kanalkapazität* C eines Kanals ist definiert durch

$$C := \max_{\substack{\text{alle W-Vert.}\\ \text{der Quelle } Q}} I(Q,E)$$

Einheit: Bit/Symbol

Bemerkung.
$0 \leq C \overset{\text{Bem. nach 2.13+2.5}}{\leq} \log(q)$, falls $|A| = q$.

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

2.15 Beispiele.

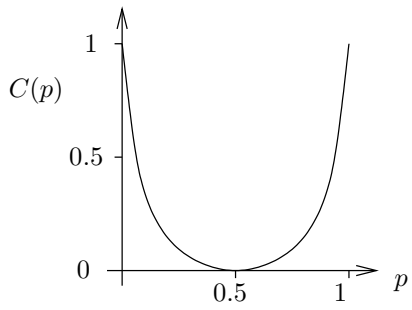
(a) Sei K ein ungestörter Kanal. Dann gilt:
$I(Q,E) \overset{2.12(a)}{=} H(E) \overset{2.9}{=} H(Q)$, also $C = \log(q)$ nach 2.5 (Q mit Gleichverteilung). ($C = 1$ für eine binäre Quelle)

(b) Sei K ein total gestörter Kanal. Dann gilt:
$C = 0$ nach 2.12(b).

2.16 Satz.
Die Kapazität eines binären symmetrischen Kanals ist

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

(Gleichheit wird erreicht für die Gleichverteilung auf Q)
$[C = 1 \Leftrightarrow p = 0$ *(ungestörter Kanal) oder* $p = 1$ *(jedes Bit wird umgedreht)*
$C = 0 \Leftrightarrow p = \frac{1}{2}$ *(total gestörter Kanal)*$]$



Beweis:
$H(E|Q) = -(1-p)\log(1-p) - p\log p$ nach 2.12(c).
Daher ist $I(Q,E) = H(E) + (1-p)\log(1-p) + p\log p$. $I(Q,E)$ wird also maximal, wenn $H(E)$ maximal wird. Dies ist nach 2.5 für $p_E(0) = p_E(1) = \frac{1}{2}$ der Fall.
Aus $p_E(0) = (1-p)\cdot p_Q(0) + p\cdot p_Q(1)$ und $p_E(1) = (1-p)\cdot p_Q(1) + p\cdot p_Q(0)$ folgt dann $p_Q(0) = p_Q(1) = \frac{1}{2}$.
Dann ist $H(Q) = H(E) = 1$ und die Behauptung folgt. □

Also: Die Kanalkapazität gibt an, wieviel nutzbare Information über einen Kanal übertragbar ist. Sei z.B. $|A| = 2$ und die Datenrate des Kanals sei 10^6 Bit/Sekunde (physikalisch). Hat der Kanal eine Kapazität von $0,8$ Bit/Symbol (Symbol $\widehat{=}$ Bit, da $|A| = 2$), so sind im besten Fall nur $0,8 \cdot 10^6$ Bit/Sekunde nutzbare Information übertragbar.

Der Satz von Shannon wird aussagen, dass man diese Übertragbarkeit von nutzbarer Information fast vollständig ausnutzen kann bei beliebig kleiner Decodierfehlerwahrscheinlichkeit.

Situation:

Code $\mathcal{C}$ der Länge n über A, $|A| = q$. $R = \frac{\log_q(\mathcal{C})}{n}$ Infosymbol/Codesymbol. D.h. der Anteil R der Symbole eines Codewortes enthält Information.
Manchmal braucht man auch die binäre Rate R_b (vgl. Bemerkung nach 1.9): Wieviele Bits pro Codesymbol sind Information. Ein Symbol in A entspricht $\log_2(q)$ Bits. Also $R_b = R \cdot \log_2(q)$. ($q = 2$, so $R = R_b$)

Code: R_b Bits pro Codesymbol reine Information
Kanal: überträgt maximal C Bit/Symbol an Information

Kann man einen Code finden, dessen binäre Rate möglichst nahe an der Kapazität des Kanals liegt und der trotzdem eine beliebig kleine Decodierfehlerwahrscheinlichkeit hat?
Antwort: Ja.
Dazu: Wie decodiert man am besten?

2.17 Decodierprinzipien.
Voraussetzung: Die Codewörter werden über einen beliebigen diskreten gedächtnislosen Kanal übertragen.
Angenommen $y = (y_1, \ldots, y_n) \in B^n$ wird empfangen. In welches $\widehat{x} \in \mathcal{C}$ soll decodiert werden?
Bisher haben wir $A = B$ und Hamming-Decodierung betrachtet. Ob das gut ist, hängt natürlich von p_Q ab. (Betrachte den Spezialfall, dass Q immer nur dasselbe Codewort sendet.)
Da der Kanal gedächtnislos ist, ist die Wahrscheinlichkeit, dass y empfangen wird, falls $x \in \mathcal{C}$ gesendet wird: $p(y|x) = \prod_{i=1}^{n} p(y_i|x_i)$.
Die Wahrscheinlichkeitsverteilung p_Q auf A liefert in derselben Art eine Wahrscheinlichkeitsverteilung p_Q auf $\mathcal{C}$. (Idealisierung; ansonsten wählt man direkt eine Wahrscheinlichkeitsverteilung auf $\mathcal{C}$ bzw. A^n und fasst A^n als Alphabet auf, um die vorige Begrifflichkeit anwenden zu können.)

(a) Die beste Decodierung ist die *Maximum-Aposteriori-Decodierung (MAP)*:
Wähle zu empfangenem $y \in B^n$ dasjenige $\widehat{x} \in \mathcal{C}$ mit

$$p_Q(\widehat{x})p(y|\widehat{x}) \geq p_Q(x)p(y|x) \text{ für alle } x \in \mathcal{C}.$$

Damit maximiert man die Aposteriori-Wahrscheinlichkeit

$$\underbrace{p(x|y)}_{\substack{\text{bedingte W.-keit, dass } x \\ \text{gesendet wurde, falls } y \\ \text{empfangen wurde.}}} = \frac{p(y|x)p_Q(x)}{\underbrace{p_Q(y)}_{\text{fest für geg. } y}} \text{ (Bayes-Formel).}$$

Problem: In der Regel kennt man p_Q nicht.
Dann:

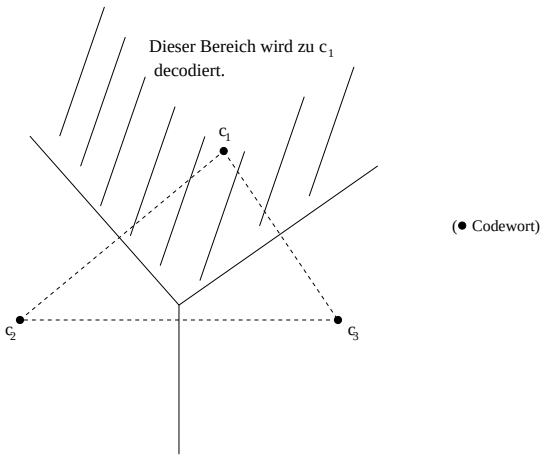
(b) *Maximum-Likelihood-Decodierung (MLD)*:
Zu gegebenem $y \in B^n$ wird zur Decodierung ein Codewort $\widehat{x} \in \mathcal{C}$ gewählt, so dass

$$p(y|\widehat{x}) \geq p(y|x) \text{ für alle } x \in \mathcal{C}.$$

Sind alle Codewörter gleich wahrscheinlich, so entspricht dies MAP.

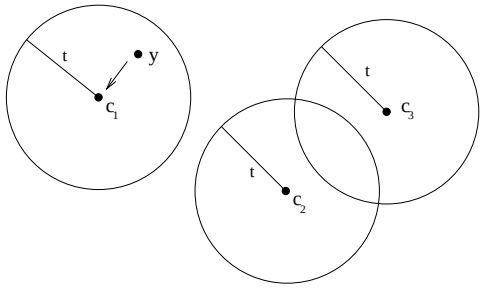
2.18 Decodierprinzipien bei binärem symmetrischem Kanal. ($p < \frac{1}{2}$)

(a) Bei einem binären symmetrischen Kanal ist MLD = Hamming-Decodierung.
Für jedes empfangene Wort wird nach dem nächstgelegenen Codewort decodiert.

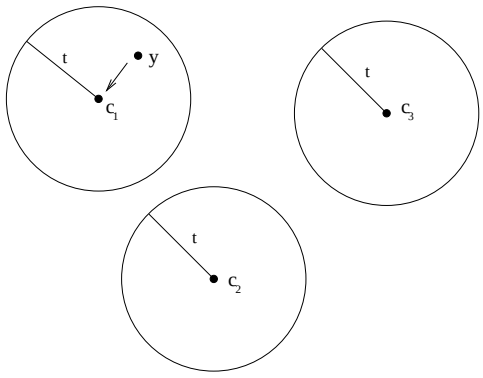


Hamming-Decodierung kann sehr aufwändig sein. Stattdessen gelegentlich einfacheres Verfahren:

(b) *Bounded Distance Decoding (BDD)*:
Es wird ein festes t gewählt und um jedes Codewort wird eine Kugel vom Radius t gelegt. Decodiert werden diejenigen Wörter, die in genau einer Kugel liegen, die anderen nicht.
Schlechtere Decodierung als MLD, aber in gewissen Fällen einfacher.



(c) *Bounded Minimal Distance Decoding (BMD)*:
Um jedes Codewort wird eine Kugel vom Radius $t = \lfloor \frac{d(\mathcal{C})-1}{2} \rfloor$ gelegt. Dann sind die Kugeln disjunkt (siehe 1.6). Es werden nur diejenigen Wörter decodiert, die in einer Kugel liegen. (Form von BDD)



2.19 Kanalcodierungssatz von Shannon. *(Claude Shannon 1916-2001; u.a. MIT)*
Gegeben sei ein diskreter gedächtnisloser Kanal mit Kapazität C, Quellenalphabet A.
Wähle $\varepsilon, \varepsilon' > 0$ beliebig.
Dann existiert ein $n \in \mathbb{N}$ und ein Blockcode der Länge n über A, dessen binäre Rate R die Bedingung

$$C - \varepsilon' \leq R < C$$

erfüllt und wobei die Decodierfehlerwahrscheinlichkeit bei MLD kleiner als ε ist.
Umgekehrt:
Ist $R_b > C$, so kann die Decodierfehlerwahrscheinlichkeit eine gewisse Grenze nicht unterschreiten.

Bevor wir den Beweis skizzieren, vergleichen wir die Aussage des Satzes mit einem Code, den wir schon kennen, dem 3-fachen Wiederholungscode über $\mathbb{Z}_2$.

2.20 Beispiele.
$\mathcal{C}$ = 3-facher Wiederholungscode von Blöcken der Länge 2 über $\mathbb{Z}_2$, $R = R_b = \frac{1}{3}$. $\mathcal{C} = \{(000000), (010101), (101010), (111111)\}$.
Wir nehmen einen binären symmetrischen Kanal an.
$p(0|0) = p(1|1) = 1 - p$, $p(1|0) = p(0|1) = p$, $p < \frac{1}{2}$.
Wie groß ist die Decodierfehlerwahrscheinlichkeit bei Hamming-Decodierung (= MLD, falls alle Codewörter gleich wahrscheinlich sind)?

- 0,1 Fehler treten auf:
  Korrekte Decodierung

- 2 Fehler treten auf:
  Wahrscheinlichkeit für genau 2 Fehler: $p^2(1-p)^4$.
  Bei 6 der 15 Möglichkeiten für die 2 Fehler wird falsch decodiert (bei jedem Codewort).
  Die Wahrscheinlichkeit für 2 Fehler und falsche Decodierung ist also $6p^2(1-p)^4$.

- 3,4,5,6 Fehler treten auf:
  Es wird immer falsch decodiert.
  Wahrscheinlichkeit für i Fehler: $p^i(1-p)^{6-i}$. Diese könne an $\binom{6}{i}$ Stellen auftreten, $i = 3, 4, 5, 6$.

Die Decodierfehlerwahrscheinlichkeit ist also

$$6p^2(1-p)^4 + 20p^3(1-p^3) + 15p^4(1-p)^2 + 6p^5(1-p) + p^6.$$

Angenommen $p = 0,01$. Dann ist die Decodierfehlerwahrscheinlichkeit etwa $5,96 \cdot 10^{-4} < 0,001$.
Beachte: Die Kapazität des Kanals ist $C = 1+p\log p+(1-p\log 1-p) \approx 0,92$. Die Rate $R \approx 0,33$ ist deutlich kleiner.

Beweisidee des Satzes von Shannon bei binärem symmetrischen Kanal.
Sei $p < \frac{1}{2}$.
Sei $x \in \{0,1\}^n$ (n im Moment beliebig), x wird über den Kanal gesendet.
Das empfangene Wort enthält im Mittel $n \cdot p$ Fehler.
(Wahrscheinlichkeit für r Fehler ist $\binom{n}{r}p^r(1-p)^{n-r}$; Binomialverteilung)
Je größer n, umso größer ist die Wahrscheinlichkeit, dass die Anzahl der Fehler nahe bei $n \cdot p$ liegt.
(Verhältnis von Standardabweichung zu Erwartungswert (Variationskoeffizient): $\sqrt{\frac{1-p}{np}}$)
Betrachte die Kugel vom Radius $n \cdot p$ um x (bzgl. der Hamming-Distanz). Sie enthält $\sum_{k=0}^{n\cdot p}\binom{n}{k}$ Wörter.
Eine einfache Abschätzung (siehe z.B. Friedrichs, Anhang A.2) zeigt:
$\sum_{k=0}^{n\cdot p}\binom{n}{k} \approx 2^{nH(p)}$, wobei $H(p) = -p\log p - (1-p)\log(1-p)$ (Entropie für 2 Zeichen mit Wahrscheinlichkeit p, $1-p$).
Könnte man $\{0,1\}^n$ in disjunkte Kugeln vom Radius $n\cdot p$ zerlegen, so erhielte man $\frac{2^n}{2^{nH(p)}} = 2^{n(1-H(p))} = 2^{nC}$ viele Kugeln.
Wählt man die Mittelpunkte dieser Kugeln als Codewörter, so wird (bei großem n) fast immer korrekt decodiert. Die Rate des Codes ist $\frac{nC}{n} = C$.
Die Zerlegung von $\{0,1\}^n$ in disjunkte Kugeln vom Radius np geht allerdings nicht.
Was man aber machen kann:
Man wählt den Radius der Kugeln etwas größer als np (in von ε' und p abhängiger Weise).
Man wählt zufällig $2^{R\cdot n}$ ($C - \varepsilon' \leq R < C$) viele Wörtern in $\{0,1\}^n$ als Codewörter.
Die MLD-Decodierung wird verschlechtert zu BDD mit Parameter t.
Man kann dann zeigen, dass selten ein Wort empfangen wird, das in keiner Kugel liegt (denn dann müssen mehr als t Fehler aufgetreten sein) und dass außerdem im Mittel über alle Zufallswahlen auch die Wahrscheinlichkeit klein ist, dass es zwei Codewörter im Abstand $\leq t$ von einem empfangenen Wort gibt (weil man den Radius der Kugeln nicht zu groß gewählt hat).

Insgesamt kann man dann zeigen, dass der Erwartungswert für einen Decodierfehler $\leq \varepsilon$ ist, wenn man n genügend groß wählt. Daher muss es mindestens einen solchen Code geben.

(Genauer Beweis: Friedrichs, Kapitel 2.7 (für binären symmetrischen Kanal); Roman, Kapitel 3.4)

3 Lineare Codes

3.1 Definition.
Sei K ein endlicher Körper, $n \in \mathbb{N}$. Ein *linearer Code* ist ein Unterraum des K-Vektorraums K^n (Zeilenvektoren); insbesondere ist K das Alphabet.
Ist $\dim(\mathcal{C}) = k$ $(\leq n)$ und $\mathrm{d}(\mathcal{C}) = d$, so heißt $\mathcal{C}$ ein $[n, k]$ oder $[n, k, d]$-Code über K. $|\mathcal{C}| = q^k$.
Ist $|K| = q$, so verwendet man auch die Schreibweise $[n, k, d]_q$-Code.
Rate: $\frac{k}{n}$.

3.2. Endliche Körper

- $\mathbb{Z}_p$ ist bzgl. der Addition und Multiplikation modulo p ein endlicher Körper, falls p eine Primzahl ist.
- 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 (bis auf Isomorphie) genau einen endlichen Körper K mit $|K| = p^m$.
- Konstruktion endlicher Körper der Ordnung p^m, $m > 1$:
  Sei f ein irreduzibles Polynom vom Grad m in $\mathbb{Z}_p[t]$.
  $\mathbb{Z}_p[t]_{m-1}$ ist die Menge der Polynome vom Grad $\leq m-1$ über $\mathbb{Z}_p$.
  Bezgl. der normalen Addition von Polynomen und der Multiplikation $g \odot h = g \cdot h \mod f$ (Rest bei Division durch f) ist $\mathbb{Z}_p[t]_{m-1}$ ein Körper der Ordnung p^m.

  Beispiel: Körper der Ordnung 4

  $\mathbb{Z}_2 = \{0, 1\}$, $f = t^2 + t + 1$ ist irreduzibel.
  $K = \{0, 1, t, t+1\}$.
  Es ist $t^2 \bmod f = t + 1$, $t^2 + t \bmod f = 1$ und $(t+1)(t+1) \bmod f = t^2 + 1 \bmod f = t$.
  Damit erhält man folgende Multiplikationstabelle:

$\odot$	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

3.3 Beispiele.

(a) n-facher Wiederholungscode über Z_p (jedes Symbol wird n-mal wiederholt):
$\mathcal{C} = \{(0, ..., 0), (1, ..., 1), ..., (p-1, ..., p-1)\}$ ist ein linearer $[n, 1]$-Code über Z_p; $\mathcal{C} = \langle(1, \ldots, 1)\rangle$.

(b) $\mathcal{C} = \{(u_1, ..., u_{n-1}, u_n) : u_i \in \mathbb{Z}_p, \sum_{i=1}^{n} u_i = 0 \text{ in } \mathbb{Z}_p\}$ ist ein $[n, n-1]$-Code über $\mathbb{Z}_p$ („Parity-Check-Code“);
$\mathcal{C} = \langle(1, 0, \ldots, 0, p-1), \ldots, (0, \ldots, 0, 1, p-1)\rangle$.

(c) $\mathcal{C} = \{(0,0,0,0,0), (0,1,1,0,1), (1,0,1,1,0), (1,1,0,1,1)\}$ aus 1.2 (c) ist ein linearer $[5, 2]$-Code über $\mathbb{Z}_2$; $\mathcal{C} = \langle(1,0,1,1,0), (0,1,1,0,1)\rangle$.

3.4 Definition.
Sei K ein endlicher Körper.

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

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

3.5 Satz.
Ist $\mathcal{C} \neq \{0\}$ ein linearer Code über K, so ist $\mathrm{d}(\mathcal{C}) = \mathrm{wt}(\mathcal{C})$.

Beweis.
Da $\mathcal{C}$ linear ist ($x, x' \in \mathcal{C} \Rightarrow x - x' \in \mathcal{C}$) und der Hamming-Abstand translationsinvariant ist (1.5), gilt:

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

□

3.6 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

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

Erzeugermatrix (oder *Generatormatrix*) von $\mathcal{C}$.

3.7 Satz.
Sei $\mathcal{C}$ ein $[n,k]$-Code über K und G eine $k \times n$-Matrix über K.
Dann gilt:
G ist eine Erzeugermatrix von $\mathcal{C} \Leftrightarrow \mathcal{C} = \{uG : u \in \underset{\text{Zeilenvektor}}{\uparrow} K^k\}$.

Beweis:
$\Rightarrow$: $uG = (u_1, \ldots, u_k) \begin{pmatrix} g_1 \\ \vdots \\ g_k \end{pmatrix} = (u_1 g_1 + \ldots + u_k g_k)$.
Also: $\{uG : u \in K^k\}$ besteht aus allen Linearkombinationen von $g_1, ..., g_k$. Behauptung folgt.

$\Leftarrow$: Sei $\mathcal{C} = \{uG : u \in K^k\}$, $G = \begin{pmatrix} g_1 \\ \vdots \\ g_k \end{pmatrix}$.
Wie oben: Alle Linearkombinationen der Zeilen von G bilden $\mathcal{C}$.
$\dim(\mathcal{C}) = k \Rightarrow g_1, ..., g_k$ sind linear unabhängig, bilden also eine Basis von $\mathcal{C}$. □

3.8 Bemerkung.

(a) G beschreibt eine injektive lineare Abbildung $K^k \to K^n$; $\mathcal{C}$ ist das Bild. Durch Multiplikation mit G wird eine Codierungsmöglichkeit von *Infowörtern* in K^k auf *Codewörter* in K^n angegeben.

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

(c) Gibt es eine Erzeugermatrix

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

so ist für ein Infowort $u = (u_1, \ldots, u_k)$ das zugeordnete Codewort $uG = (u_1, \ldots, u_k, * \ldots *)$.
Eine solche Erzeugermatrix heißt von *Standardform* (oder in *systematischer Form*; obige Codierung heißt dann *systematische Codierung*). Nicht jeder Code besitzt eine Erzeugermatrix in Standardform. Doch durch elementare Zeilenumformungen und Spaltenvertauschungen kann man einen „äquivalenten“ Code erhalten mit Erzeugermatrix in Standardform.

Beispiel. $\mathcal{C}$ aus 3.3 (c)
Erzeugermatrix $G = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix}$ in Standardform
$G' = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix}$ ist ebenfalls eine Erzeugermatrix von $\mathcal{C}$, aber nicht in Standardform.

Unterräume kann man auch als Lösungsräume homogener linearer Gleichungssysteme beschreiben. Dies führt zum Begriff der Erzeugermatrix eines Codes:

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

$$\textit{Ist } x \in K^n, \textit{ so: } \quad x \in \mathcal{C} \Leftrightarrow Hx^t = \underset{(n-k)\times 1}{0}. \qquad (\textit{äquivalent: } xH^t = \underset{1\times(n-k)}{0})$$

H heißt Kontrollmatrix von $\mathcal{C}$. Es ist $\operatorname{rg}(H) = n-k$ *(d.h. die Zeilen von H sind linear unabhängig).*
(Dann gilt auch: $HG^t = 0$, falls G Erzeugermatrix des Codes $\mathcal{C}$ ist.)

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

Betrachte das lineare Gleichungssystem:

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

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

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

($h_i \ldots, h_{n-k}$ erfüllen die i-te Gleichung oben).
Also: $Hx^t = 0$ für alle $x \in \mathcal{C}$.
Aus rg(H)$= n-k$ folgt dim(Lösungsraum von $Hx^t = 0) = k$, das heißt $\mathcal{C}$ ist der Lösungsraum von $Hx^t = 0$. □

3.10 Bemerkung.

(a) Der Beweis von 3.9 gibt ein Verfahren an für:
Erzeugermatrix → Kontrollmatrix.

- Löse das LGS $Gx^t = 0$.
- Wähle eine Basis des Lösungsraumes $h_1^t, \ldots, h_{n-k}^t$ (h_i Zeilenvektoren).
- $H := \begin{pmatrix} h_1 \\ \vdots \\ h_{n-k} \end{pmatrix}$ ist eine Kontrollmatrix.

(b) Umgekehrt: Kontrollmatrix → Erzeugermatrix
Bilde eine Basis des Lösungsraumes von $Hx^t = 0$.

3.11 Beispiele.

(a) Parity-Check-Code $\mathcal{C} = \{(u_1, ..., u_n) : \sum_{i=1}^n u_i = 0\}$ über $\mathbb{Z}_2$:

$$G = \begin{pmatrix} 1 & 0 & \ldots & \ldots & 0 & 1 \\ 0 & 1 & 0 & \ldots & 0 & 1 \\ \vdots & \ddots & & & \vdots & \vdots \\ \vdots & & \ddots & & 0 & 1 \\ 0 & \ldots & \ldots & \ldots & 1 & 1 \end{pmatrix} \text{ Erzeugermatrix}$$

$H = (1, 1, ..., 1)$ Kontrollmatrix

(b) $\mathcal{C}$ aus 1.2(c), Erzeugermatrix $G = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix}$
Bestimmung der Kontrollmatrix:
Lösung von $Gx^t = 0$: x_3, x_4, x_5 beliebig und $x_2 = x_3 + x_5$, $x_1 = x_3 + x_4$.

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

An der Kontrollmatrix kann man den Minimalabstand des linearen Codes ablesen:

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

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

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 $s_{i_1}, \ldots, s_{i_w}$. Dann existieren $c_j \in K$ mit $\sum_{j=1}^{n} c_j s_j = 0$ und $c_j \neq 0$ genau für $j \in \{i_1, \ldots, i_w\}$. Also ist auch $\sum_{j=1}^{n} c_j s_j^t = 0$. Setzte $c = (c_1, \ldots, c_n)$. Dann ist

$$cH^t = (c_1, ..., c_n) \begin{pmatrix} s_1^t \\ \vdots \\ s_n^t \end{pmatrix} = \sum_{i=1}^{n} c_i s_i^t = 0. \quad (*)$$

Damit ist $Hc^t = 0$.
Also ist $c \in \mathcal{C}$ und $\mathrm{wt}(c) = w$ und damit $\mathrm{wt}(\mathcal{C}) \leq w$.
Angenommen es gibt ein $0 \neq \overline{c} \in \mathcal{C}$ mit $\overline{w} = \mathrm{wt}(\overline{c}) < w$. Dann folgt aus $\overline{c}H^t = 0$ wie in $(*)$, dass es $\overline{w}$ linear abhängige Spalten in H gibt. Dies ist ein Widerspruch zur Wahl von w. Also $\mathrm{wt}(\mathcal{C}) = w$.
2. Gleichheit klar. □

3.13 Beispiel.
Beispiel aus 3.11(c). Je zwei Spalten von H sind linear unabhängig, aber $\begin{pmatrix}1\\1\\0\end{pmatrix}, \begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\end{pmatrix}$ sind linear abhängig. Also ist $\mathrm{d}(C) = 3$.

3.14 Korollar (Singleton-Schranke)**.**
Ist $\mathcal{C} \neq \{0\}$ ein linearer $[n,k]$-Code über K mit $\mathrm{d}(\mathcal{C}) = d$, so ist $d \leq n-k+1$.

Beweis:
Nach der 2. Gleichheit in 3.12 gilt: $d \leq \mathrm{rg}(H) + 1 = n - k + 1$
(H Kontrollmatrix von $\mathcal{C}$). □

3.15 Bemerkung und Definition.

(a) Beachte: $k = \log_q(|\mathcal{C}|)$.
Man kann auch für beliebige Codes $\mathcal{C}$ über A, $|A| = q$, zeigen, dass $d \leq n - \log_q(|\mathcal{C}|) + 1$, d.h. $\log_q(|\mathcal{C}|) \leq n - d + 1$ ($|\mathcal{C}| \leq q^{n-d+1}$) gilt.
Codes, für die Gleichheit gilt in 3.14 (bzw. 3.15) heißen *MDS-Codes* (maximum distance separable).

(b) Über $\mathbb{Z}_2$ gibt es nur triviale Beispiele von MDS-Codes. Man kann zeigen, dass

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

die einzigen MDS-Codes über $\mathbb{Z}_2$ sind.
Später (Kap. 7) werden wir nicht triviale Beispiele von MDS-Codes über anderen Körpern angeben.

Wir beweisen jetzt eine weitere Schranke, die auch für beliebige Codes gilt. Wir benötigen das folgende Lemma, das wir für $q = 2$ schon implizit im Beweis des Satzes von Shannon verwendet haben.

3.16 Lemma.
Sei $|A| = q$, $x \in A^n$, $t \in \mathbb{N}_0$, $K_t(x) = \{y \in A^n : \mathrm{d}(x,y) \le t\}$.
Dann ist $|K_t(x)| = \sum_{i=0}^{t} \binom{n}{i}(q-1)^i$.

Beweis.
Abstand i zu x: i Positionen von x auswählen:
$\binom{n}{i}$ Möglichkeiten.
An jeder Position Eintrag von x ändern:
je $(q-1)$ Möglichkeiten.
Insgesamt: $\binom{n}{i}(q-1)^i$ Wörter y mit $\mathrm{d}(x,y) = i$.

Behauptung folgt. □

3.17 Satz (Hamming-Schranke, Kugelpackungsschranke)**.**
Sei $\mathcal{C}$ *ein Code der Länge* n *über* A, $|\mathcal{C}| > 1$, $|A| = q$.
Sei $t \in \mathbb{N}_0$ *maximal mit* $\mathrm{d}(\mathcal{C}) \ge 2t+1$, *d.h.* $t = \lfloor \frac{\mathrm{d}(\mathcal{C})-1}{2} \rfloor$. *Dann gilt*

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

Beweis.
Kugeln von Radius t um Codewörter sind disjunkt (siehe Bemerkung nach Definition 1.6).
Mit 3.16: $|\mathcal{C}| \cdot \sum_{i=0}^{t} \binom{n}{i}(q-1)^i \le |A^n| = q^n$. □

3.18 Definition.
Ein Code heißt *perfekt*, falls in 3.17 Gleichheit gilt.
Äquivalent: $\dot{\bigcup}_{x\in\mathcal{C}} K_t(x) = A^n$. ($\dot{\bigcup}$ bezeichnet die disjunkte Vereinigung.)

3.19 Bemerkung.
Ist $\mathcal{C}$ perfekt, $|\mathcal{C}| > 1$, t wie in 3.17, so ist $\mathrm{d}(\mathcal{C}) = 2t + 1$ (d.h. perfekte Codes haben ungeraden Minimalabstand).

Beweis.
Es ist $\mathrm{d}(\mathcal{C}) \geq 2t + 1$ nach Wahl von t.
Wähle $x \in \mathcal{C}$ und ändere x an $t+1$ Stellen ab. Das liefert z mit $\mathrm{d}(x, z) = t+1$.
Also $x \notin K_t(x)$. Perfektheit: $\exists\ x' \in \mathcal{C}$ mit $\mathrm{d}(x', z) \leq t$. $\mathrm{d}(x, x') \leq \mathrm{d}(x, z) + \mathrm{d}(x', z) \leq 2t + 1$.
Also: $\mathrm{d}(\mathcal{C}) = 2t + 1$. □

Wir konstruieren nun mit Hilfe der Kontollmatrix eine Serie linearer perfekter Codes (mit $\mathrm{d} = 3$), die Hamming-Codes.

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

Konstruktion.
$|K^\ell \setminus \{0\}| = q^\ell - 1$. Also enthält K^ℓ genau $n = \frac{q^\ell - 1}{q-1}$ 1-dimensionale Unterräume. Wähle aus jedem einen Vektor $\neq 0$ aus und bilde aus den entsprechenden Spaltenvektoren eine $\ell \times n$-Matrix H.
Es ist $\mathrm{rg}(H) = \ell$, denn H enthält ℓ linear unabhängige Spalten.
Sei $\mathcal{C}$ der Code mit Kontrollmatrix H,

$$\mathcal{C} = \{x \in K^n : Hx^t = 0\}\ \textit{Hamming-Code}.$$

$\dim(\mathcal{C}) = n - \ell = k$, $|\mathcal{C}| = q^k$.
Je zwei Spalten in H sind linear unabhängig, aber es gibt drei linear abhängige Spalten. 3.12: $\mathrm{d}(\mathcal{C}) = 3$.
Perfektheit: (t=1)

$$\frac{q^n}{\sum_{i=0}^{1} \binom{n}{i}(q-1)^i} = \frac{q^n}{1 + n(q-1)} = \frac{q^n}{q^\ell} = q^{n-\ell} = q^k = |\mathcal{C}|.$$

Beachte: Es gibt mehrere Möglichkeiten für einen $[n, k]$-Hamming-Code, je nach Auswahl und Anordnung der Spaltenvektoren. Sie haben aber alle (bei festen q, ℓ) die gleichen Parameter.

3.21 Beispiele.

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

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

liefert einen $[7, 4]$-Hamming-Code über $\mathbb{Z}_2$
Also: $\mathcal{C} = \{(x_1, \ldots, x_7) : x_1 + x_4 + x_6 + x_7 = 0,\ x_2 + x_4 + x_5 + x_7 = 0,$
$x_3 + x_5 + x_6 + x_7 = 0\}$

Erzeugermatrix für $\mathcal{C}$:

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

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

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

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

Toto 13'er Wette:
Fülle 59.049 Toto-Tipps aus mit den Codewörtern des $[13, 10]$-Hamming-Codes über $\mathbb{Z}_3$.
Jeder Vektor $\in \mathbb{Z}_3^{13}$ hat höchstens Abstand 1 von einem Codewort. D.h. man hat mindestens 12 Richtige. (Vgl. Anzahl aller Tipps $= 3^{13} = 1.594.323$)
Ergebnis vom 1./2.5.2010: $(1\ 2\ 0\ 0\ 0\ 1\ 2\ 1\ 0\ 0\ 0\ 2\ 1) = y$

$Hy^t = \begin{pmatrix} 2 \\ 0 \\ 2 \end{pmatrix}$, also $y \notin \mathcal{C}$.

(Wenn man in H statt der Spalte $\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$ die Spalte $\begin{pmatrix} 2 \\ 0 \\ 2 \end{pmatrix}$ gewählt hätte, wäre $y \in \mathcal{C}$.)
Also hat man nur 1×12 Richtige und mehrmals 10 bzw. 11 Richtige.
Einsatz: $29.525,50 + \underbrace{4921 \times 0,35}_{\text{Bearbeitungsgebühr}} = 31.246,85$

Gewinn bei 12 Richtigen: 13.625,90 (bei 13 Richtigen: 81.755,80)

Was ist das nächstgelegene Codewort zu y (Decodierung)?
Es gibt $x \in \mathcal{C}$ mit $\mathrm{d}(x,y) = 1$. Alle Elemente von $\mathcal{C}$ zu testen wäre zu langsam. Schneller:
x unterscheidet sich von y an einer Stelle i, statt y_i steht dort
$(y_i + a) \bmod 3$, $a = 1$ oder 2.

Also $\begin{pmatrix}2\\0\\2\end{pmatrix} = Hy^t = H(x^t + (0,\dots,0,\underset{\underset{i}{\uparrow}}{a},0,\dots,0)^t) = a \cdot \begin{pmatrix}h_{1i}\\h_{2i}\\h_{3i}\end{pmatrix}$,
$a = 1, 2$.

Suche also in H die Spalte, die ein Vielfaches von $\begin{pmatrix}2\\0\\2\end{pmatrix}$ ist:

Die gesuchte Spalte steht an 8. Stelle, d.h. $i = 8$.
$\begin{pmatrix}2\\0\\2\end{pmatrix} = a \cdot \begin{pmatrix}1\\0\\1\end{pmatrix}$ liefert $a = 2$.
Ersetze nun die i-te Stelle von y durch $y_i - a$, also in unserem Fall y_8 durch $y_8 - 2 = y_8 + 1 = 1 + 1 = 2$.
Das liefert die Decodierung

$$y \to x = (1\ 2\ 0\ 0\ 0\ 1\ 2\ \underset{\underset{\text{Hannover-Gladbach 6:1}}{\downarrow}}{\underline{2}}\ 0\ 0\ 0\ 2\ 1). \text{ Das war der Tipp.}$$

Dies geht allgemein:

3.22 Schnelles Verfahren zur Hamming-Decodierung von Hamming–Codes.
Empfängt man y und ist $Hy^t = 0$, so ist $y \in \mathcal{C}$.
Gilt $Hy^t \neq 0$, so suche in H diejenige Spalte s mit $Hy^t = a \cdot s$, $a \in K \setminus \{0\}$. Ist dies die i-te Spalte von H, so ändere y_i in $y_i - a \pmod q$. Dies liefert $x \in \mathcal{C}$, zu dem y decodiert wird.
(Im binären Fall vereinfacht sich das Verfahren, da $a = 1$; ändere das Bit an der i-ten Stelle.)

Wir befassen uns jetzt mit der Hamming-Decodierung beliebiger linearer $[n,k]_q$-Codes.
Wurde y empfangen, so könnte man alle q^k Codewörter testen, welches minimalen Abstand zu y hat. Das kann sehr aufwändig sein.
Bei Codes mit $2k > n$ gibt es ein effizienteres Verfahren: die Syndrom-Decodierung. Dazu benötigt man den Begriff der Nebenklassen eines Unterraums:

3.23 Bemerkung (Nebenklassen von Unterräumen in Vektorräumen)**.**
Sei $\mathcal{C}$ ein Unterraum des Vektorraums V. Für jedes $v \in V$ heißt

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

die *Nebenklasse* von $\mathcal{C}$ zu v. (Beachte: $v = v + 0 \in v + \mathcal{C}$.)

(a) Für $v_1, v_2 \in V$ gilt entweder $v_1 + \mathcal{C} = v_2 + \mathcal{C}$ oder $(v_1 + \mathcal{C}) \cap (v_2 + \mathcal{C}) = \varnothing$.

Beweis: Angenommen, $(v_1 + \mathcal{C}) \cap (v_2 + \mathcal{C}) \neq \varnothing$. 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 ein beliebiges $x \in \mathcal{C}$ gilt $v_1 + x = v_2 + (x_2 - x_1) + x \in v_2 + \mathcal{C}$. D.h. $v_1 + \mathcal{C} \subseteq v_2 + \mathcal{C}$. Analog erhält man die andere Inklusion.
Insgesamt gilt $v_1 + \mathcal{C} = v_2 + \mathcal{C}$.

(b) $v_1 + \mathcal{C} = v_2 + \mathcal{C} \Leftrightarrow v_1 - v_2 \in \mathcal{C} \Leftrightarrow v_1 \in v_2 + \mathcal{C}$
(Insbesondere: $v_1 + \mathcal{C} = \mathcal{C} (= 0 + \mathcal{C}) \Leftrightarrow v_1 \in \mathcal{C}$)

Beweis: 1. Äquivalenz:
„$\Rightarrow$“: $v_1 = v_1 + 0 = v_2 + x$ für ein $x \in \mathcal{C} \Rightarrow v_1 - v_2 = x \in \mathcal{C}$.
„$\Leftarrow$“: $v_1 - v_2 = x \in \mathcal{C} \Rightarrow v_1 + 0 = v_2 + x \in (v_1 + \mathcal{C}) \cap (v_2 + \mathcal{C}) \overset{\text{(a)}}{\Rightarrow} v_1 + \mathcal{C} = v_2 + \mathcal{C}$.
2. Äquivalenz klar.

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

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

(d) Sei V ein Vektorraum über einem endlichen Körper K. Dann gilt $|\mathcal{C}| = |v + \mathcal{C}|$ für alle $v \in V$:
Die Abbildung $\left\{ \begin{array}{ccc} \mathcal{C} & \to & v + \mathcal{C} \\ x & \mapsto & v + x \end{array} \right.$ ist bijektiv.

(e) Aus (c) und (d) folgt: Ist $\mathcal{C}$ ein $[n, k]$-Code über K, $|K| = q$, so hat $\mathcal{C}$ genau q^{n-k} verschiedene Nebenklassen in K^n.

3.24 Syndrom-Decodierung linearer Codes.
Sei $\mathcal{C}$ ein linearer $[n, k]$-Code über K, $|K| = q$, mit Kontrollmatrix H; also ist H eine $(n - k) \times n-$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, y_2 liegen in derselben 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 haben das gleiche Syndrom.)
$[y_1 + \mathcal{C} = y_2 + \mathcal{C} \Leftrightarrow y_1 - y_2 \in \mathcal{C} \overset{(a)}{\Leftrightarrow} 0 = H(y_1 - y_2)^t = Hy_1^t - Hy_2^t]$

(c) Jedes $z \in K^{n-k}$ tritt als Syndrom auf (klar nach (b) und 3.23(e)).

Daher: Wird $x \in \mathcal{C}$ gesendet und $y = x + f$ 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, einen sogenannten *Nebenklassenführer* (im Allgemeinen nicht eindeutig).
Decodierung: $y \to y - e$.
Dazu: Ordne die Nebenklassenführer in lexikographischer Ordnung ihrer Syndrome (definiere dazu eine Ordnung auf K). Z.B. im binären Fall: Das Syndrom gibt die Stelle $(0, ..., 2^{n-k} - 1)$ des gesuchten Nebenklassenführers an $(0 < 1)$.
Das kann immer noch sehr aufwändig sein, z.B. bei einem $[70, 50]$-Code über $\mathbb{Z}_2$. $\mathcal{C}$ hat dann 2^{20} Nebenklassen, jeder Nebenklassenführer hat 70 Bit.
Speicherplatzbedarf: $70 \cdot 2^{20}$ Bit $= 8,75$ Megabyte (Allgemein: $q^{n-k} \cdot n \cdot \log q$ Bits)
Speichert man hingegen alle 2^{50} Codewörter, so hat man einen Speicherplatzbedarf von $70 \cdot 2^{50}$ Bit ≈ 9.000 Terabyte.

3.25 Definition.
Sei K ein endlicher Körper. Die Abbildung $< , >: K^n \times K^n \to K$ ist definiert durch

$$< u, v >:= \sum_{i=1}^{n} u_i v_i = uv^t \in K, \text{ wobei } u = (u_1, \ldots, u_n),\ v = (v_1, \ldots, u_n).$$

(Vgl. Euklidsches Skalarprodukt im $\mathbb{R}^n$.)

3.26 Bemerkung.

(a) $< u + v, w > = < u, w > + < v, w >$
(b) $< \lambda u, v > = \lambda < u, v >$
(c) $< u, v > = < v, u >$

} für alle $u, v, w \in K^n$, $\lambda \in K$

(a)-(c) besagen, dass $< , >$ eine symmetrische Bilinearform ist.

(d) $< 0, v > = < v, 0 > = 0$ für alle $v \in K^n$

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

Beweis. (e) Sei $v = e_i$. Dann gilt $0 =< u, e_i >= u_i$. Dies gilt für $i = 1, \ldots, n$, also $u = 0$. □

Beachte: Aus $< v, v >= 0$ folgt im Allgemeinen nicht $v = 0$.
Z.B. gilt in $K = \mathbb{Z}_2$: $< v, v >= 0 \Leftrightarrow v$ hat eine gerade Anzahl an Einsen.

3.27 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* (oder *orthogonale*) Code zu $\mathcal{C}$.

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

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

(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 $\mathcal{C}$, $y \in K^n$, $G = \begin{pmatrix} x_1 \\ \vdots \\ x_k \end{pmatrix}$, $\{x_i\}_{i=1}^k$ Basis von $\mathcal{C}$.

$$\begin{aligned} y \in \mathcal{C}^\perp &\Leftrightarrow < x, y >= 0 \text{ für alle } x \in \mathcal{C} \\ &\Leftrightarrow < x_i, y >= 0 \quad i = 1, \ldots, k \\ &\Leftrightarrow Gy^t = 0 \end{aligned}$$

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

(b) $\mathcal{C} \subseteq (\mathcal{C}^\perp)^\perp$ ist klar. Die Gleichheit folgt aus Dimensionsgründen nach (a).

(c) G Erzeugermatrix von $\mathcal{C} \Rightarrow G$ Kontrollmatrix von $\mathcal{C}^\perp$ (siehe (a)).

Sei umgekehrt G Kontrollmatrix von $\mathcal{C}^\perp$, $G = \begin{pmatrix} x_1 \\ \vdots \\ x_k \end{pmatrix}$.

Dann gilt $Gy^t = 0$ für alle $y \in \mathcal{C}^\perp$, d.h. $< x_i, y >= 0$ für alle $i = 1, \ldots, k$ und alle $y \in \mathcal{C}^\perp$.

$\Rightarrow x_1, \ldots, x_k \in \left(\mathcal{C}^\perp\right)^\perp \overset{(b)}{=} \mathcal{C}$.

Zeilen von G linear unabhängig, $\dim(\mathcal{C}) = k \Rightarrow G$ Erzeugermatrix von $\mathcal{C}$.

Die zweite Äquivalenz folgt aus (b).

(d) Es gilt:

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

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

□

3.29 Beispiel.

Duale Codes zu den Hamming-Codes:

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

Die Kontrollmatrix H enthält aus jedem 1-dimensionalen Unterraum von K^ℓ einen Vektor $\neq 0$.

Sei $\mathcal{C} = \mathcal{H}^\perp$.

Nach 3.28(c) ist H Erzeugermatrix von $\mathcal{C}$. $\mathcal{C}$ ist also ein $\left[\frac{q^\ell-1}{q-1}, \ell\right]$-Code. Er heißt *Simplex-Code*. Es gilt:

Jedes Codewort $\neq 0$ in $\mathcal{C}$ hat Gewicht $q^{\ell-1}$, d.h. der Abstand zwischen zwei verschiedenen Codewörtern ist immer konstant $q^{\ell-1}$; insbesondere ist $\mathrm{d}(\mathcal{C}) = q^{\ell-1}$:

Seien $z_1, \ldots, z_l$ 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

$x = \sum_{i=1}^{\ell} a_i z_i = \sum_{i=1}^{\ell} a_i \cdot (z_{i1}, \ldots, z_{in})$.

Setze $a = (a_1, \ldots, a_\ell) \in K^\ell$. Beachte: $a \neq 0$.

Es gilt: $x_j = 0 \Leftrightarrow \sum_{i=1}^{\ell} a_i z_{ij} = 0 \Leftrightarrow \underbrace{\begin{pmatrix} z_{1j} \\ \vdots \\ z_{j\ell} \end{pmatrix}}_{j\text{-te Spalte von } H} \in \langle a \rangle^{\perp}$.

$\langle a \rangle^{\perp}$ hat Dimension $\ell - 1$ nach 3.28(a).
$\langle a \rangle^{\perp}$ enthält also $\frac{q^{\ell-1}-1}{q-1}$ der Spalten h_j von H.
Damit sind in x genau $\frac{q^{\ell-1}-1}{q-1}$ viele Komponenten $x_j = 0$.
$\mathrm{wt}(x) = n - \frac{q^{\ell-1}-1}{q-1} = \frac{q^{\ell-1}(q-1)}{q-1} = q^{\ell-1}$.
Der Simplex-Code ist nicht perfekt, von kleiner Dimension, hat aber großen Minimalabstand.

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

$$\text{Erzeugermatrix} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

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

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

4.1 Definition und Bemerkung.

(a) Eine *Boole'sche Funktion* ist eine Abbildung $f : \mathbb{Z}_2^m \to \mathbb{Z}_2$.
Ordne die Elemente von $\mathbb{Z}_2$ lexikographisch:
$v_0 = (0, \ldots, 0)$, $v_1 = (0, \ldots, 0, 1), \ldots, v_{2^m-1} = (1, \ldots, 1)$.
(D.h. v_i entspricht dem Koeffizientenvektor der Binärdarstellung von i, also $i = \sum_{j=1}^{m} x_j 2^{j-1}$, falls $v_i = (x_m, \ldots, x_1)$.)

$$f \to \underline{f} = (f(v_0), \ldots, f(v_{2^m-1}))$$

liefert eine bijektive Abbildung von der Menge der Boole'schen Funktionen $\mathbb{Z}_2^m \to \mathbb{Z}_2$ auf $\mathbb{Z}_2^{2^m}$. Also gibt es 2^{2^m} Boole'sche Funktionen $\mathbb{Z}_2^m \to \mathbb{Z}_2$. Sie bilden einen $\mathbb{Z}_2$-Vektorraum und einen kommutativen Ring mit Eins.

(b) Sei $J = \{j_1, \ldots, j_r\} \subseteq \{1, \ldots, m\}$, $j_1 < \ldots < j_r$.
Definiere die *Boole'sche Monomfunktion* f_J durch $f_J(x_m, \ldots, x_1) = x_{j_1} \cdot \ldots \cdot x_{j_r}$ (Multiplikation in $\mathbb{Z}_2$) für $(x_m, \ldots, x_1) \in \mathbb{Z}_2^m$.
(Also $f_J(x_m, \ldots, x_1) = 1 \Leftrightarrow x_{j_1} = \ldots = x_{j_r} = 1$.)
Es gibt 2^m Boole'sche Monomfunktionen f_J. $J = \varnothing$, so $f_J = 1$.

(c) Die Boole'schen Monomfunktionen $f_J : Z_2^m \to \mathbb{Z}_2$ sind linear unabhängig über $\mathbb{Z}_2$:
Sei $\sum_{J \subseteq \{1,\ldots,m\}} c_J f_J = 0$.
Einsetzen von $(0, \ldots, 0)$ liefert: $c_\varnothing = 0$.
Einsetzen von $(0, \ldots, 0, 1), \ldots, (1, 0, \ldots, 0)$ liefert: $c_i = 0$, $i = 1, \ldots, m$.
So fortfahrend erhält man $c_J = 0$ für $|J| = 2, 3, \ldots, m$.

(d) Eine *Boole'sche Polynomfunktion* ist eine $\mathbb{Z}_2$-Linearkombination von Boole'schen Monomfunktionen. Nach (b), (c) gibt es davon 2^{2^m}.
Nach (a): Jede Boole'sche Funktion ist eine Boole'sche Polynomfunktion.
Grad(f) = maximaler Grad der auftretenden Monome.
Grad(0) = $-\infty$.
Das Produkt von zwei Boole'schen Polynomfunktionen erhält man durch distributives Ausmultiplizieren. Beachte dabei, dass $x_i x_j = x_j x_i$ und $x_i^2 = x_i$ (als Funktionen) gilt.

Beispiel:
$f(x_3, x_2, x_1) = 1 + x_2 + x_1x_3 + x_2x_3$ (Summe und Produkt über $\mathbb{Z}_2$).
Dann ist $\underline{f} = (1, 1, 0, 0, 1, 0, 1, 0)$.

[Jedes Polynom in n Variablen $x_1, \ldots, x_n$ liefert auch eine Funktion $\mathbb{Z}_2^m \to \mathbb{Z}_2$; da aber x_i^e und x_i denselben Wert für $x_i = 1, 0$ liefern, braucht man nur Polynome, wo in den Monomen kein x_i mit Potenz ≥ 2 auftritt.]

4.2 Bemerkung.
Gegeben sei eine Boole'sche Funktion f durch ihren „Wertevektor" $\underline{f}$. Wie findet man die Darstellung von f als Polynomfunktion?
Die Summe entspricht XOR und das Produkt entspricht $\wedge$.
Es ist leicht, jede Boole'sche Funktion mit $\wedge$, $\vee$, $-$ (und, oder, Negation) darzustellen:

$$f = \bigvee_{\{a=(a_m,\ldots,a_1)\in\mathbb{Z}_2^m : f(a)=1\}} \left(\bigwedge_{\{j:a_j=1\}} x_j \wedge \bigwedge_{\{k:a_k=0\}} \overline{x_k} \right).$$

Benutze jetzt

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

Beispiel:
$\underline{f} = (00011000)$, d.h. $f(011) = 1 = f(100)$, Rest 0.

$$\begin{aligned} f &= x_1x_2\overline{x}_3 \vee \overline{x}_1\overline{x}_2x_3 \\ &= x_1x_2(1 + x_3) \vee (1 + x_1)(1 + x_2)x_3 \\ &= x_1x_2(1 + x_3) + (1 + x_1)(1 + x_2)x_3 + \underbrace{x_1x_2(1 + x_3)(1 + x_1)(1 + x_2)x_3}_{=0} \\ &= x_1x_2 + x_1x_2x_3 + x_3 + x_1x_3 + x_2x_3 + x_1x_2x_3 \\ &= x_1x_2 + x_1x_3 + x_2x_3 + x_3. \end{aligned}$$

4.3 Definition.
Sei $0 \leq r \leq m$. Definiere den binären *Reed-Muller-Code* $RM(r, m)$ durch
$RM(r, m) = \{\underline{f} \in \mathbb{Z}_2^{2^m} : f : \mathbb{Z}_2^m \to \mathbb{Z}_2$ Boole'sche Polynomfunktion vom Grad $\leq r\}$.

Muller,1954, Reed, 1954 (Majority-Logic-Decodierung für RM-Codes)

(Reed-Muller-Codes sind sogenannte „Auswertungscodes“, denn die Codewörter entstehen durch die Auswertung gewisser Funktionen an allen Punkten des $\mathbb{Z}_2^m$.)
Per Konstruktion gilt also $RM(0,m) \subseteq RM(1,m) \subseteq \ldots \subseteq RM(m,m)$.

4.4 Beispiel.
$m = 3$, $r = 2$, $f(x_3, x_2, x_1) = x_1 + x_2x_3 + 1$

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
$\underline{f}$	1	0	1	0	1	0	0	1

’

d.h. $(10101001) \in RM(2,3)$.

4.5 Satz.
Sei $0 \leq r \leq m$. Dann ist $RM(r,m)$ ein linearer Code der Länge 2^m, Dimension $k = \sum_{i=0}^{r} \binom{m}{i}$ und Minimalabstand $d = 2^{m-r}$.

Beweis.
Länge 2^m ist klar.
Es gibt $\binom{m}{i}$ Boole’sche Funktionen vom Grad i. Damit folgt die Behauptung über die Dimension aus 4.1(c).
Per Induktion nach m zeigen wir nun $d = \mathrm{d}\,(RM(r,m)) = 2^{m-r}$:
$m = 0$ (also auch $r = 0$): $RM(0,0) = \{(0),(1)\}$ und $\mathrm{d} = 1$. $\surd$
Sei $m > 0$. Es ist $\mathrm{wt}\left(\underline{x_1 \ldots x_r}\right) = 2^{m-r}$, also $d \leq 2^{m-r}$.
Sei $f \neq 0$ eine Polynomfunktion vom Grad $\leq r$.

$$0 \neq f(x_m, \ldots, x_1) = g(x_{m-1}, \ldots, x_1) + x_m h(x_{m-1}, \ldots, x_1).$$

Wir müssen zeigen, dass $\mathrm{wt}(\underline{f}) \geq 2^{m-r}$, d.h. dass mindestens 2^{m-r} Vektoren $v \in \mathbb{Z}_2^m$ existieren mit $f(v) \neq 0$.
Beachte: $\mathrm{Grad}(h) \leq r - 1$.
Ist $h = 0$, so ist $r \leq m - 1$. Per Induktion ist $\mathrm{wt}(\underline{g}) \geq 2^{m-r-1}$, wobei g als Funktion $\mathbb{Z}_2^{m-1} \to \mathbb{Z}_2$ betrachtet wird. Für jedes $a = (a_{m-1}, \ldots, a_1) \in \mathbb{Z}_2^{m-1}$ mit $g(a) \neq 0$ ist dann $f(0, a_{m-1}, \ldots, a_1) = f(1, a_{m-1}, \ldots, a_1) \neq 0$. Damit ist $\mathrm{wt}(\underline{f}) \geq 2 \cdot 2^{m-r-1} = 2^{m-r}$. $\surd$
Sei $h \neq 0$. Ist $g \neq 0$, so existieren per Induktion mindestens 2^{m-r-1} Vektoren $a \in \mathbb{Z}_2^{m-1}$ mit $g(a) \neq 0$. Damit ist auch $f(0,a) \neq 0$.
Ist zusätzlich $g + h \neq 0$, so existieren mindestens 2^{m-r-1} Vektoren $b \in \mathbb{Z}_2^{m-1}$ mit $(g+h)(b) \neq 0$, also ist $f(1,b) \neq 0$.
Dies liefert 2^{m-r} Vektoren $v \in \mathbb{Z}_2^m$ mit $f(v) \neq 0$. $\surd$

Ist $g = 0$, so ist $\text{Grad}(h) \leq r-1$, und es existieren per Induktion $2^{(m-1)-(r-1)} = 2^{m-r}$ Vektoren $c \in \mathbb{Z}_2^{m-1}$ mit $h(c) \neq 0$, also auch $f(1, c) \neq 0$. $\surd$
Ist $g+h = 0$, so ist $\text{Grad}(g)$=$\text{Grad}(h)$, also $\text{Grad}(g) \leq r-1$. Dann existieren per Induktion $2^{(m-1)-(r-1)} = 2^{m-r}$ Vektoren $w \in \mathbb{Z}_2^{m-1}$ mit $g(w) \neq 0$, also $f(0, w) \neq 0$. $\surd$ □

4.6 Beispiele.

(a) $\underbrace{(1, \ldots, 1)}_{2^m} \in RM(r, m)$ für alle $0 \leq r \leq m$, denn $(1, \ldots, 1)$ ist der Wert der Funktion 1.

(b) $RM(0, m)$ ist der $[2^m, 1]$-Wiederholungscode $\{\underbrace{(0, \ldots, 0)}_{2^m}, \underbrace{(1, \ldots, 1)}_{2^m}\}$.

(c) $RM(m, m) = (\mathbb{Z}_2)^{2^m}$

(d) $\mathcal{C} = RM(1, m)$ ist ein $[2^m, m+1]$-Code mit $\text{d}(\mathcal{C}) = 2^{m-1}$.
Speziell $RM(1, 5)$ ist ein $[32, 6]$-Code mit $\text{d}(\mathcal{C}) = 16$.
Er wurde bei den Mariner Expeditionen 6, 7 und 9 zum Mars (1969 - 1972) für die Bildübertragung zur Erde eingesetzt. Die $2^6 = 64$ Codewörter entsprachen der Helligkeit (Grautöne) eines Bildpunktes. Wegen $7 \leq \frac{\text{d}(\mathcal{C})-1}{2} = \frac{15}{2}$ ist $RM(1, 5)$ ein 7 Fehler-korrigierender Code der Länge 32. Ein Bild wurde in ein 600×600 Gitterpunkte zerlegt und jeder dieser 360.000 Punkte in ein Codewort codiert. [1]

(e) $RM(m-1, m) = \{x \in (\mathbb{Z}_2)^{2^m} : \text{wt}(x) \text{ ist ungerade}\}$.
(Das kann man mit der Methode aus dem Beweis von Satz 4.5 beweisen.) Insbesondere haben alle Wörter in jedem $RM(r, m)$, $r \leq m-1$, gerades Gewicht.

4.7 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}.)

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

Beweis.
Sei $\underline{f} \in RM(r,m)$ und $\underline{g} \in RM(m-r-1,m)$.
Dann ist $\text{grad}(f \cdot g) \leq r+m-r-1 = m-1 \Rightarrow \underline{f \cdot g} \in RM(m-1,m)$.
Nach 4.6(e) hat $\underline{f \cdot g}$ gerades Gewicht.
Sei $\underline{f} = (f(v_0), \ldots, f(v_{2^m-1})) = (c_0, \ldots, c_{2^m-1})$ und $\underline{g} = (g(v_0), \ldots, g(v_{2^m-1})) = (d_0, \ldots, d_{2^m-1})$. Dann hat $\underline{f \cdot g} = (c_0 d_0, ..., c_{2^m-1} d_{2^m-1})$ gerades Gewicht.
$\Rightarrow \langle \underline{f}, \underline{g} \rangle = \sum_{i=0}^{2^m-1} c_i d_i = 0$ (in $\mathbb{Z}_2$).
Also: $RM(m-r-1,m) \subseteq RM(r,m)^\perp$.

$$\begin{aligned}
\dim(RM(m-r-1,m)) &= \sum_{i=0}^{m-r-1} \binom{m}{i}. \\
\dim(RM(r-m)^\perp) &\overset{3.28}{=} 2^m - \sum_{i=0}^{r} \binom{m}{i} \\
&= \sum_{i=0}^{m} \binom{m}{i} - \sum_{i=0}^{r} \binom{m}{i} \\
&= \sum_{i=0}^{m} \binom{m}{i} - \sum_{j=m-r}^{m} \binom{m}{j} \\
&= \sum_{i=0}^{m-r-1} \binom{m}{i}.
\end{aligned}$$

Aufgrund der Gleichheit der Dimensionen gilt Gleichheit. □

4.8 Bemerkung.
Es gibt eine 1-1-Beziehung zwischen Boole'schen Funktionen $\mathbb{Z}_2^m \to \mathbb{Z}_2$ und Teilmengen des $\mathbb{Z}_2^m$. Betrachte dazu die beiden injektiven Abbildungen

$$\begin{aligned}
f &\mapsto \text{Tr}(f) = \{x \in \mathbb{Z}_2^m : f(x) = 1\} \quad \text{(Träger von } f\text{)} \quad \text{und} \\
M &\mapsto \chi_M : \quad \chi_M(x) = \begin{cases} 1 & x \in M \\ 0 & x \notin M \end{cases} \quad \text{(charakteristische Funktion von } f\text{)}.
\end{aligned}$$

Es gilt $\chi_{\text{Tr}(f)} = f$ und $\text{Tr}(\chi_M) = M$ (inverse Operationen).
(Also: Jede Boole'sche Funktion ist von der Form χ_M.)

Frage: Für welche $M \subseteq \mathbb{Z}_2^m$ ist $\underline{\chi_M} \in RM(r,m)$?

Eine hinreichende Bedingung liefert der folgende Satz:

4.9 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 von $\mathbb{Z}_2^m$ ist M Lösungsraum eines homogenen LGS in m Unbekannten mit $m-t$ Gleichungen (Koeffizientenmatrix = Kontrollmatrix von M):

$$\begin{aligned} a_{1,1}x_1 + ... + a_{1,m}x_m &= 0 \\ \vdots \qquad\qquad &\quad \vdots \\ a_{m-t,1}x_1 + ... + a_{m-t,m}x_m &= 0 \end{aligned}$$

Setze $p_i(x_1, ..., x_m) = a_{i,1}x_1 + ... + a_{i,m}x_m + 1$ (Boole'sches Polynom vom Grad 1).
Sei $p(x_1, ..., x_m) = \Pi_{i=1}^{m-t} p_i(x_1, ..., x_m)$. Dann ist p ein Boole'sches Polynom vom Grad $m - t \leq r$, d.h. $\underline{p} \in RM(r, m)$.
Nun gilt für $(y_1, ..., y_m) \in \mathbb{Z}_2^m$:

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

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

4.10 Bemerkung.

(a) Ein t-dimensionaler affiner Unterraum des $\mathbb{Z}_2^m$ ist eine Nebenklasse eines t-dimensionalen (linearen) Unterraums des $\mathbb{Z}_2^m$.
Die Elemente jedes t-dimensionalen affinen Unterraumes des $\mathbb{Z}_2^m$ lassen sich als Lösungen eines inhomogenen LGS in m Unbekannten mit $m-t$ Gleichungen beschreiben.
Indem man im obigen Beweis die Definition der p_i modifiziert ($+1$ nur, wenn in der entsprechenden Gleichung auf der rechten Seite eine 0 steht), lässt sich genauso zeigen:
Ist M ein t-dimensionaler affiner Unterraum des $\mathbb{Z}_2^m$, $t \geq m - r$, so ist $\underline{\chi_M} \in RM(r, m)$.

(b) Man kann zeigen:
$RM(r, m) = \langle \underline{\chi_M} \in \mathbb{Z}_2^{2^m} : M \text{ } t\text{-dim. affiner UR von } \mathbb{Z}_2^m, \ t \geq m - r \rangle$.
Beweis: MacWilliams, Sloane, Chapter 13, §6.

4.11 Beispiel.
$m = 3$, $r = 1$.
$M = \{(000), (101), (011), (110)\}$ (2-dimensionaler Unterraum von $\mathbb{Z}_2^3$)
$\underline{\chi_M} = (10010110)$

Kontrollmatrix für M:

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

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

Es gilt also:

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

Der Lösungsraum dieses LGS wird erzeugt von $(1,1,1)$, also ist $(a_{11}, a_{12}, a_{13}) = (1,1,1)$.
4.9: Für $p(x_3, x_2, x_1) = x_1 + x_2 + x_3 + 1$ ist $\underline{p} = \underline{\chi_M} \in RM(1,3)$.
(Man hätte p natürlich auch mit der Methode von 4.2 aus $\underline{\chi_M}$ berechnen können.)

Für Reed-Muller-Codes gibt es ein effizientes Decodierverfahren (das aber nicht auf Reed-Muller-Codes beschränkt ist): Majority-Logic-Decodierung. Dazu benötigen wir zunächst folgende Definition:

4.12 Definition.
Seien $y^{(1)}, \ldots, y^{(\ell)} \in K^n$, K ein endlicher Körper, $1 \leq i_1 < i_2 < \ldots < i_s \leq n$. $y^{(1)}, \ldots, y^{(\ell)}$ heißen *orthogonal bezüglich der Positionen* $i_1, \ldots, i_s$, falls gilt:

$$\mathrm{Tr}(y^{(i)}) \cap \mathrm{Tr}(y^{(j)}) = \{i_1, \ldots, i_s\} \quad \text{für alle } i, j = 1, \ldots, \ell,\ i \neq j.$$

$(\mathrm{Tr}(x_1, \ldots, x_n) = \{i : x_i \neq 0\})$

D.h.

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

(2) Ist $a \in \{1, \ldots, n\} \setminus \{i_1, \ldots, i_s\}$, so ist $y_a^{(j')} \neq 0$ für höchstens ein $j' \in \{1, \ldots, \ell\}$. (D.h. außerhalb der Positionen $i_1, \ldots, i_s$ sind die $y^{(j)}$ „orthogonal“ zueinander.)

(Wichtiger Fall: orthogonal bezüglich einer Position)

Wir betrachten im Folgenden nur den Fall $K = \mathbb{Z}_2$.
Grundlage für die Majority-Logic-Decodierung ist der folgende Satz:

4.13 Satz.
Sei $\mathcal{C}$ ein $[n,k]$-Code über $\mathbb{Z}_2$, $y^{(1)},\dots,y^{(\ell)} \in \mathbb{Z}_2^n$ orthogonal bezüglich der Positionen $i_1,\dots,i_s$.
Angenommen $x \in \mathcal{C}$ wird gesendet und $z \in \mathbb{Z}_2^n$ wird empfangen, und es sind maximal $t \leq \frac{1}{2}\ell$ Fehler aufgetreten. Dann gilt:

Die Anzahl der Fehler in z an den Stellen $i_1,\dots,i_s$ ist ungerade
$\Leftrightarrow$ $|\{j$: Anzahl der Fehler an den Positionen von $Tr(y^{(j)})$ ist ungerade$\}| >$
$|\{j$: Anzahl der Fehler an den Positionen von $Tr(y^{(j)})$ ist gerade$\}|$.

Beweis.
Von den t fehlerhaften Positionen in z seien $u \geq 0$ innerhalb von $i_1,\dots,i_s$, o.B.d.A. $i_1,\dots,i_u$, und $t-u$ außerhalb von $i_1,\dots,i_s$.
Für jede der $t-u$ fehlerhaften Stellen außerhalb von $i_1,\dots,i_s$ gibt es maximal ein $y^{(j')}$, das an der Stelle Eintrag $\neq 0$ hat (Bedingung (2) aus 4.12).
Also: Mindestens $\ell-(t-u)$ der $y^{(j)}$ haben an sämtlichen fehlerhaften Stellen außerhalb von $i_1,\dots,i_s$ eine Null. Seien dies bei geeigneter Nummerierung $y^{(1)},\dots,y^{(v)}$, $v \geq \ell-(t-u)$. Es ist $v \geq \ell - t \geq \frac{\ell}{2}$, da $t \leq \frac{\ell}{2}$.

„$\Rightarrow$“:
Es ist u ungerade, also $u \geq 1$. Daher $v \geq \ell-(t-u) \geq \ell-(t-1) \geq \frac{\ell}{2}+1$.
Da die fehlerhaften Positionen in $\mathrm{Tr}(y^{(j)})$ für $j=1,\dots,v$ sämtlich in $i_1,\dots,i_s$ liegen, ist die Anzahl der fehlerhaften Positionen dieser $\mathrm{Tr}(y^{(j)})$ gerade u, also ungerade.
Daher:
$|\{j$: Anzahl der Fehler an den Positionen von $\mathrm{Tr}(y^{(j)})$ ist ungerade$\}| \geq v \geq \frac{\ell}{2}+1$. Also hat die Mehrzahl der $y^{(j)}$, $j=1,\dots,\ell$ eine ungerade Anzahl von Fehlern an den Positionen ihrer Träger.

„$\Leftarrow$“:
Angenommen, u ist gerade. Dann haben alle $y^{(j)}$, $i=1,\dots,v$, eine gerade Anzahl von Fehlern an den Positionen von $\mathrm{Tr}(y^{(j)})$. Aus $v \geq \frac{\ell}{2}$ folgt somit
$|\{j$: Anzahl der Fehler an den Positionen von $\mathrm{Tr}(y^{(j)})$ ist ungerade$\}| \leq$
$|\{j$: Anzahl der Fehler an den Positionen von $\mathrm{Tr}(y^{(j)})$ ist gerade$\}|$. □

Als Korollar erhalten wir jetzt eine Aussage, falls die Vektoren $y^{(1)},\dots,y^{(\ell)}$ aus 4.13 zusätzlich in $\mathcal{C}^\perp$ liegen.

4.14 Korollar.
Sei $\mathcal{C}$ ein $[n,k]$-Code über $\mathbb{Z}_2$, $y^{(1)},\dots,y^{(\ell)} \in \mathcal{C}^\perp$ orthogonal bezüglich der Positionen $i_1,\dots,i_s$.
Angenommen $x \in \mathcal{C}$ wird gesendet und $z \in \mathbb{Z}_2^n$ wird empfangen, und es sind maximal $t \leq \frac{1}{2}\ell$ Fehler aufgetreten. Dann gilt:

Anzahl der Fehler in z an den Stellen $i_1, \ldots, i_s$ ist ungerade
$\Leftrightarrow |\{j : \langle y^{(j)}, z\rangle = 1\}| > |\{j : \langle y^{(j)}, z\rangle = 0\}|$.

Ist $s = 1$, so lässt sich auf diese Weise feststellen, ob Position i_1 in z fehlerhaft ist oder nicht. Diese lässt sich dann korrigieren ($K = \mathbb{Z}_2$): *(Einfache) Majority-Logic-Decodierung (Majoritäts-Decodierung).*

(Will man alle Positionen korrigieren, so braucht man für jede Position geeignete ℓ Vektoren aus $\mathcal{C}^\perp$.)

Beweis.
Wir behaupten, dass gilt:
$|\{j : \langle y^{(j)}, z\rangle = 1\}| =$
$|\{j :$ Anzahl der Fehler an den Positionen von $\mathrm{Tr}(y^{(j)})$ ist ungerade$\}|$.
Für alle $j = 1, \ldots, \ell$ ist $\langle y^{(j)}, x\rangle = 0$, da $y^{(1)}, \ldots, y^{(\ell)} \in \mathcal{C}^\perp$.
Sei $j \in \{1, \ldots, \ell\}$, u die Anzahl der Fehler von z an den Positionen von $\mathrm{Tr}(y^{(j)})$. Dann gilt:

$$\begin{aligned}
\langle y^{(j)}, z\rangle &= \langle y^{(j)}, z\rangle - \langle y^{(j)}, x\rangle \\
&= \sum_{i=1}^{n} y_i^{(j)} z_i - \sum_{i=1}^{n} y_i^{(j)} x_i \\
&= \sum_{i \in \mathrm{Tr}(y^{(j)})} y_i^{(j)} (z_i - x_i) \\
&= \sum_{i \in \mathrm{Tr}(y^{(j)})} z_i - x_i = \begin{cases} 1 & u \text{ ungerade} \\ 0 & u \text{ gerade} \end{cases}
\end{aligned}$$

(Beachte, dass die Summen in $\mathbb{Z}_2$ gebildet werden.)
Damit ist die Eingangsbehauptung bewiesen und 4.14 folgt aus 4.13. □

4.15 Beispiel.
Sei $\mathcal{C}$ der [7,3]-Simplex-Code über $\mathbb{Z}_2$ (der duale Code zum binären [7,4]-Hamming-Code, vgl. 3.29).
[Beachte: $\mathrm{d}(\mathcal{C}) = 4$, 1-Fehler-korrigierend]
Erzeugermatrix von $\mathcal{C}^\perp$ = Erzeugermatrix des Hamming-Codes
(=Kontrollmatrix von $\mathcal{C}$)

$$\tilde{H} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \text{ (vgl. 3.21(a))}$$

Wähle

$$\begin{array}{lcll} y^{(1)} & = & \text{erste Zeile von } \tilde{H} & = (1101000) \\ y^{(2)} & = & \text{dritte Zeile von } \tilde{H} & = (1010010) \\ y^{(3)} & = & \text{Summe der zweiten und vierten Zeile} & = (1000101) \end{array}$$

$y^{(1)}, y^{(2)}, y^{(3)} \in \mathcal{C}^\perp$ sind orthogonal bezüglich Position 1.
Angenommen $z = (0100110)$ wird empfangen, maximal 1 Fehler sei aufgetreten.

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

Mit 4.14 folgt: Stelle z_1 ist fehlerhaft.
Decodierung zu $x = (1100110)$.
Tatsächlich gilt $x \in \mathcal{C}$, da $\tilde{H}x = 0$. Wäre dies nicht der Fall (wenn also mehr als ein Fehler aufgetreten ist), so würde man nicht decodieren.

Angenommen $\tilde{z} = (0110010)$ wird empfangen, maximal 1 Fehler sei aufgetreten.

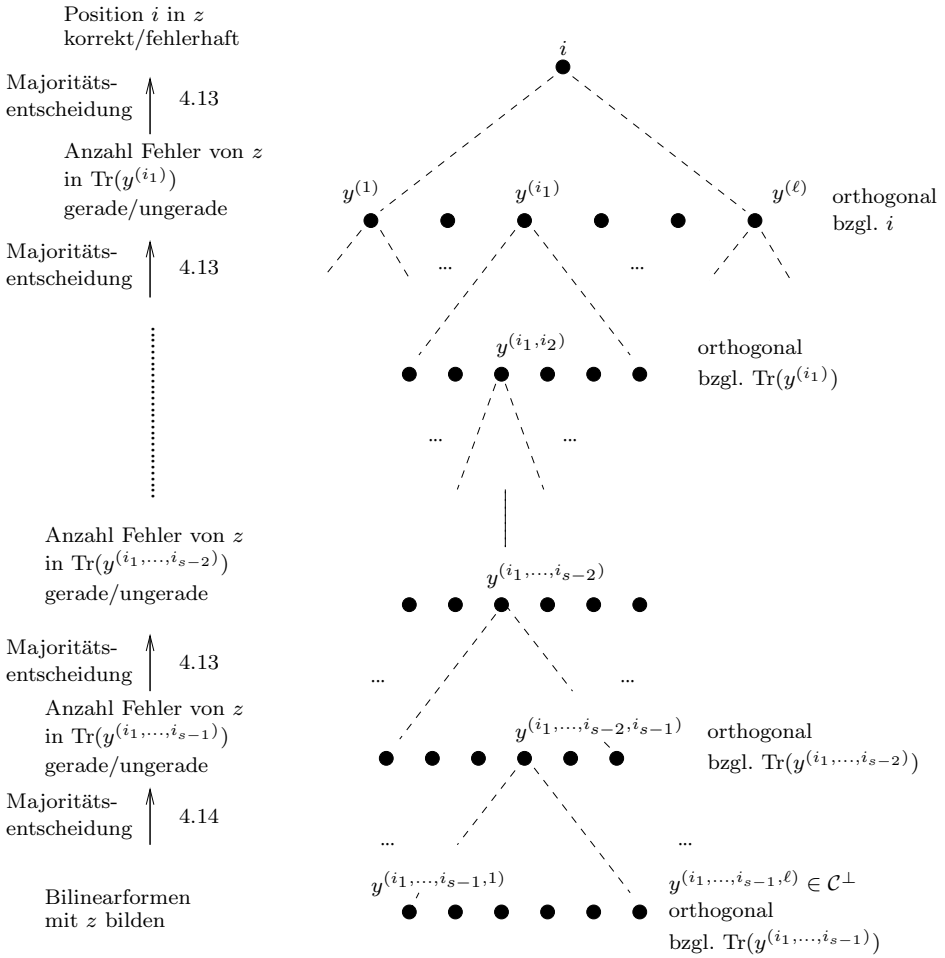
$$\langle y^{(1)}, z\rangle = 1, \quad \langle y^{(2)}, z\rangle = 0, \quad \langle y^{(3)}, z\rangle = 0.$$

4.14: z_1 korrekt.
(Über die übrigen Stellen lässt sich mit $y^{(1)}, y^{(2)}, y^{(3)}$ nichts aussagen!)

4.16. Prinzip der mehrstufigen Majority-Logic-Decodierung

Sei $\mathcal{C}$ ein $[n, k]$-Code über $\mathbb{Z}_2$, $d = \mathrm{d}(\mathcal{C})$, $t \leq \lfloor \frac{\mathrm{d}(\mathcal{C})-1}{2} \rfloor$
(im besten Fall $t = \lfloor \frac{\mathrm{d}(\mathcal{C})-1}{2} \rfloor$).
Sei $x \in \mathcal{C}$ gesendet und $z \in \mathbb{Z}_2^n$ empfangen worden. Es seien maximal t Fehler aufgetreten. Sei $\ell = 2t$.

s-stufige Majority-Logic-Decodierung (der Position i)



Insgesamt: maximal ℓ^s Berechnungen von Bilinearformen und $\sum_{j=0}^{s-1} \ell^j = \frac{\ell^s-1}{\ell-1}$ Majoritätsbestimmungen. Dabei muss s so gewählt werden, dass man in der untersten Stufe jeweils genügend viele, nämlich ℓ, orthogonale Vektoren in $\mathcal{C}^\perp$ zur Verfügung hat.
[ℓ ist jeweils die Mindestanzahl der benötigten Vektoren; das Verfahren funktioniert auch, wenn in einzelnen Schritten mehr als ℓ orthogonale Vektoren verwendet werden. Das kann von Vorteil sein, wenn man gleichzeitig mehrere Positionen von z korrigieren möchte, wie man bei den Reed-Muller-Codes sehen wird.]

Anwendung auf Reed-Muller-Codes:

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

Beweis.
$|M| = 2^p$. Für jedes $v \in \mathbb{Z}_2^m \setminus M$ ist $\langle M \cup v \rangle$ ein $(p+1)$-dimensionaler Unterraum, der M enthält. Er enthält $2^{p+1} - 2^p = 2^p$ Vektoren, die nicht in M liegen. Jeder dieser Vektoren spannt mit M also den gleichen $(p+1)$-dimensionalen Unterraum auf.
Wegen $|Z_2^m \setminus M| = 2^m - 2^p$ ist die Anzahl der gesuchten Unterräume $\frac{2^m - 2^p}{2^p} = 2^{m-p} - 1$.
Die zweite Aussage folgt aus Dimensionsgründen. □

4.18. Mehrstufige Majority-Logic-Decodierung für Reed-Muller-Codes

Gegeben: $\mathcal{C} = RM(r,m)$, $r < m$, $\mathrm{d}(\mathcal{C}) = 2^{m-r}$, also $(2^{m-r-1} - 1)$-Fehler-korrigierend.

Voraussetzungen: $x \in RM(r,m)$ wird gesendet, $z \in \mathbb{Z}_2^{2^m}$ wird empfangen, maximal $t = 2^{m-r-1} - 1$ Fehler seien aufgetreten.
(Dann ist $\ell \geq 2^{m-r} - 2$, vgl. 4.16 und Bem. nach 4.16.)

1. Schritt:
Sei M ein r-dimensionaler Unterraum von $\mathbb{Z}_2^m$. Dann gibt es $2^{m-r} - 1$ $(r+1)$-dimensionale Unterräume von Z_2^m, die M enthalten (4.17). Wähle $2^{m-r} - 2$ von diesen, $M_1, \ldots, M_{2^{m-r}-2}$.
Da $RM(r,m)^\perp = RM(m-r-1, m)$ (4.7) und $m - (m-r-1) = r+1$, ist $\underline{\chi_{M_j}} \in RM(r,m)^\perp$ nach 4.9.
Bestimme $\left\langle \underline{\chi_{M_j}}, z \right\rangle$, $j = 1, \ldots, 2^{m-r} - 2$.
Die $\underline{\chi_{M_j}}$ sind orthogonal bezüglich $\mathrm{Tr}(\underline{\chi_M})$, da $M_i \cap M_j = M$.
Stelle fest, ob die Mehrzahl der $\left\langle \underline{\chi_{M_j}}, z \right\rangle = 1$ ist oder nicht.
4.14: Es ist bekannt, ob die Anzahl der Fehler von z an den Positionen von $\mathrm{Tr}(\underline{\chi_{M_j}})$ ungerade oder gerade ist.
Führe diese Untersuchung für alle r-dimensionalen Unterräume durch.

2. Schritt:
Wähle einen $(r-1)$-dimensionalen Unterraum N von $\mathbb{Z}_2^m$.
Nach 4.17 gibt es $2^{m-r+1} - 1 > 2^{m-r} - 2$ r-dimensionale Unterräume, die N enthalten. Wähle $2^{m-r} - 2$ von diesen, $N_1, N_2, \ldots N_{2^{m-r}-2}$.

Nach dem 1. Schritt ist bekannt, ob an den Positionen von $\mathrm{Tr}(\underline{\chi_{N_j}})$ eine ungerade oder eine gerade Anzahl von Fehlern von z vorliegt.
4.13: Stelle durch Majoritätsentscheidung fest, ob an den Positionen von $\mathrm{Tr}(\underline{\chi_{N_j}})$ eine ungerade oder eine gerade Anzahl von Fehlern von z vorliegt.
Führe dies für alle $(r-1)$-dimensionalen Unterräume durch.

3. Schritt:
Führe den 2. Schritt für alle $(r-2), \ldots, 1, 0$-dimensionalen Unterräume durch.
0-dimensionaler Fall: $\{0\} = \{v_0\}$. (Anzahl der Fehler ungerade $= z_0$ fehlerhaft.) Wir wissen daher, ob Position z_0 fehlerhaft ist oder nicht.
Da wir auch wissen, ob die Anzahl der Fehler von z an den Positionen $\{0, i\}$, $i = 1, \ldots, 2^m - 1$ gerade oder ungerade ist (denn $\{v_0, v_i\}$ ist 1-dimensionaler Unterraum), wissen wir auch, ob z_i fehlerhaft ist oder nicht.

4.19 Beispiel (Mehrstufige Majority-Logic-Decodierung von $RM(1,4)$)**.**
$n = 2^m = 16$, $k = 1 + 4 = 5$, $d = 2^3 = 8$, $t = \lfloor \frac{d-1}{2} \rfloor = 3$ $(\ell = 6)$
Wir wählen folgende Bezeichnungen für die Elemente des $\mathbb{F}_2^4$:

$v_0 = (0000)$	$v_4 = (0100)$	$v_8 = (1000)$	$v_{12} = (1100)$
$v_1 = (0001)$	$v_5 = (0101)$	$v_9 = (1001)$	$v_{13} = (1101)$
$v_2 = (0010)$	$v_6 = (0110)$	$v_{10} = (1010)$	$v_{14} = (1110)$
$v_3 = (0011)$	$v_7 = (0111)$	$v_{11} = (1011)$	$v_{15} = (1111)$

Angenommen, ein Wort $x = (x_0, \ldots, x_{15}) \in R(1,4)$ wird gesendet und $z = (1011010001001010)$ wird empfangen. Bei der Übertragung seien maximal 3 Fehler aufgetreten.
Wir nehmen an, dass maximal drei Fehler aufgetreten sind und wollen z mit Multistep Majority-Logic decodieren.
$r = 1$:
Betrachte alle 1-dimensionaen Unterräume, also $\{v_0, v_i\}$, $i = 1, \ldots, 15$. (Beachte: v_0 ist der Nullvektor.)
Bestimme, ob die Anzahl der Fehler an den zugehörigen Positionen $0, i$ gerade oder ungerade ist. Beginne mit $\{v_0, v_1\} =: M$.
Teste für alle 2-dimensionalen Unterräume $M_j \supseteq M$, ob $\langle z, \underline{\chi}_{M_j} \rangle$ gleich 0 oder 1 ist.
Es gibt sieben solcher Unterräume, nämlich

$M_1 = \{v_0, v_1, v_2, v_3\}$	$M_5 = \{v_0, v_1, v_{10}, v_{11}\}$
$M_2 = \{v_0, v_1, v_4, v_5\}$	$M_6 = \{v_0, v_1, v_{12}, v_{13}\}$
$M_3 = \{v_0, v_1, v_6, v_7\}$	$M_7 = \{v_0, v_1, v_{14}, v_{15}\}$
$M_4 = \{v_0, v_1, v_8, v_9\}$	

Für $\langle z, \underline{\chi}_{M_1} \rangle$, $i = 1, \ldots, 7$ gilt:

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, hat 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 für v_2, v_3 sind in folgender Tabelle zusammengefasst:

M	M_j	$\langle z, \underline{\chi}_{M_j} \rangle$	
$\{v_0, v_2\}$	$\{v_0, v_2, v_1, v_3\}$	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

Analog ergeben sich die übrigen Fälle. Man erhält

M	Position	Anzahl der Fehler
$\{v_0, v_1\}$	$0, 1$	gerade
$\{v_0, v_2\}$	$0, 2$	gerade
$\{v_0, v_3\}$	$0, 3$	ungerade
$\{v_0, v_4\}$	$0, 4$	gerade
$\{v_0, v_5\}$	$0, 5$	gerade
$\{v_0, v_6\}$	$0, 6$	gerade
$\{v_0, v_7\}$	$0, 7$	ungerade
$\{v_0, v_8\}$	$0, 8$	gerade
$\{v_0, v_9\}$	$0, 9$	gerade
$\{v_0, v_{10}\}$	$0, 10$	gerade
$\{v_0, v_{11}\}$	$0, 11$	ungerade
$\{v_0, v_{12}\}$	$0, 12$	gerade
$\{v_0, v_{13}\}$	$0, 13$	gerade
$\{v_0, v_{14}\}$	$0, 14$	gerade
$\{v_0, v_{15}\}$	$0, 15$	gerade

Die Mehrzahl der Anzahl der Fehler ist also gerade. Daher ist Position 0 von z korrekt. Die Positionen 3,7,11 von z müssen also fehlerhaft sein.
z wird daher zum Codewort $x = (1010010101011010)$ decodiert.

Bemerkung.
Die erste Beschreibung der Majority-Logic-Decodierung stammt von Reed (1954).
Es gibt Modifikationen und effizientere Verfahren der Majority-Logic-Decodierung.
z.B. -Mehrfachnutzung der $y^{(i_1,\ldots,i_j)}$
-Weniger Stufen
Chen (1971/72):
$RM(r, m)$ kann für $r \leq \frac{m}{2}$ mit 2-stufiger Majority-Logic decodiert werden:
Zu jeder Position i gibt es $2^{m-r}-2$ r-dimensionale affine Unterräume $M_1, \ldots, M_{2^{m-r}-2}$ mit $\underline{\chi_{M_j}}$ orthogonal bezüglich i.
Zu jedem M_i gibt es $2^{m-r}-2$ $(r+1)$-dimensionale affine Unterräume N_{i_j}, $j = 1, \ldots, 2^{m-r}-2$, mit $\underline{\chi_{N_{i_j}}}$ orthogonal bezüglich $\mathrm{Tr}(\underline{\chi_{M_i}})$; $\underline{\chi_{N_{i_j}}} \in RM(r, m)^{\perp}$.

Literatur: Mac Williams, Sloane (Chap. 10)
Roman (6.2)
van Lint (4.5)
Lin, Costello (4.3, 4.4, 8.)

5 Expander-Codes und LDPC-Codes

5.1. Vorüberlegung

(a) Gegeben sei ein binärer symmetrischer Kanal mit Fehlerwahrscheinlichkeit p. Bei einem Code der Länge n sind dann pro Codewort $n \cdot p$ Fehler zu erwarten.
Damit diese korrigierbar sind, sollte der Minimalabstand d des Codes die Bedingung $d > 2np + 1$ erfüllen.
Der sogenannte *relative Minimalabstand* $\frac{d}{n}$ muss daher größer als $2p$ sein.
Wenn man nun nicht nur einen Code betrachtet, sondern eine Familie von (linearen) Codes $\mathcal{C}_i$, $i \in \mathbb{N}$, der Länge n_i mit $\dim(C_{i+1}) > \dim(\mathcal{C}_i)$ (z.B. um eine Wahl für die Anzahl der benötigten Codewörter zu haben), so sollten die relativen Minimalabstände $\frac{d_i}{n_i}$ durch eine positive Konstante nach unten beschränkt sein, so dass alle Codes für genügend kleine p die erwartete Anzahl von Fehlern pro Codewort korrigieren können.
Gleichzeitig sollte die Rate $\frac{\dim(\mathcal{C}_i)}{n_i}$ der Codes nicht gegen Null gehen.
Also:
Konstruiere eine Familie von $[n_i, k_i, d_i]$-Codes, $i \in \mathbb{N}$, so dass Konstanten $R_0 > 0$ und $M_0 > 0$ existieren mit $\frac{k_i}{n_i} \geq R_0$ und $\frac{d_i}{n_i} \geq M_0$.
Solche Familien gibt es natürlich nach Shannons Satz (und er besagt sogar noch mehr), aber eine explizite Konstruktion, die nicht auf probabilistischen Argumenten beruht, wurde erst in den 70er Jahren gefunden. Diese sogenannten Expander-Codes sind spezielle LDPC-Codes (Low-Density-Parity-Check-Codes) und besitzen auch schnelle Decodierverfahren. Darauf wollen wir im Folgenden eingehen.

(b) Zunächst betracheten wir uns bekannte Familien von Codes, nämlich Hamming-Codes und Reed-Muller-Codes.

- Bei (binären) Hamming-Codes ist der Minimalabstand 3, Länge $2^\ell - 1$, $\ell \in \mathbb{N}$.
  Der relative Minimalabstand geht also gegen Null.
  Die Rate $\frac{2^\ell - 1 - \ell}{2^\ell - 1}$ geht dagegen gegen 1.
- Bei Reed-Muller-Codes $RM(r, m)$ ist der relative Minimalabstand $\frac{2^{m-r}}{2^m} = \frac{1}{2^r}$. Soll dieser nicht gegen Null gehen, darf r nicht beliebig wachsen. Wir könnten also ein festes r wählen und m wachsen lassen. Dann ist die Rate $\frac{\sum_{i=0}^{r} \binom{m}{i}}{2^m}$. Der Zähler ist ein Polynom in m vom Grad r. Daher geht die Rate mit wachsendem m (und festem r) gegen Null.

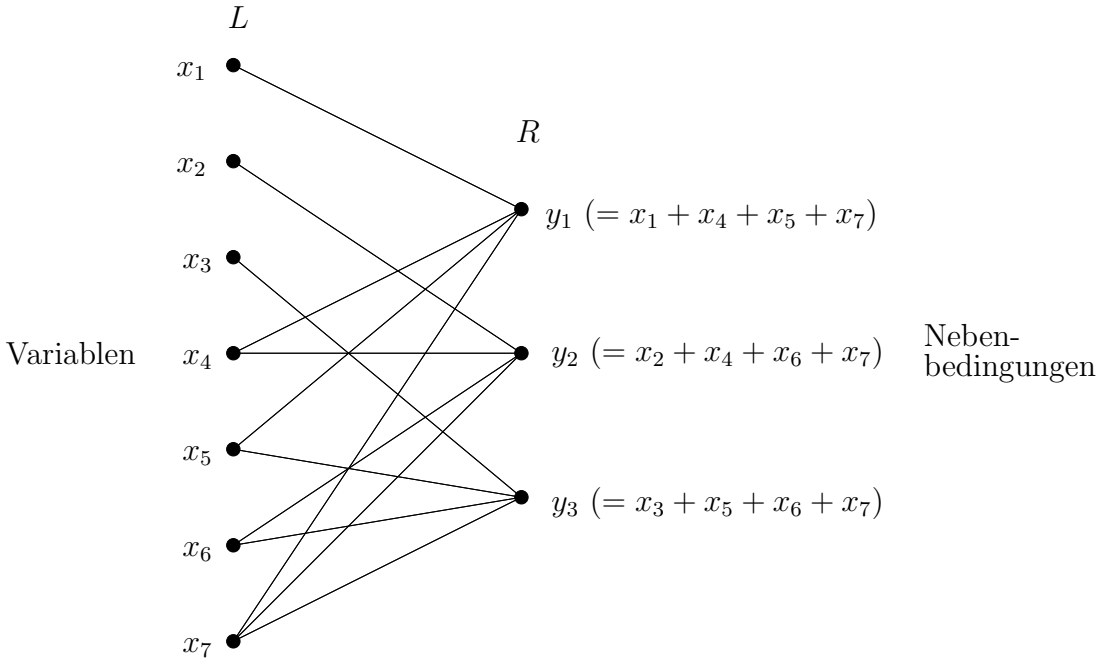
5.2 Bemerkung.
Ist $\mathcal{C}$ ein $[n,k]$-Code, so besitzt $\mathcal{C}$ eine $(n-k) \times n$-Kontrollmatrix H vom Rang $n-k$. Die Zeilen sind also linear unabhängig. (H ist nicht eindeutig bestimmt.)
Fügt man weitere Zeilen hinzu, die Linearkombinationen der Zeilen von H sind, so erhält man eine Matrix $\tilde{H}$, für die ebenfalls gilt:

$$x \in \mathcal{C} \quad \Leftrightarrow \quad \tilde{H}x^t = 0.$$

$\tilde{H}$ ist also eine Kontrollmatrix in einem allgemeineren Sinne.
In diesem Kapitel werden wir auch solche allgemeinen Kontrollmatrizen betrachten, deren Rang kleiner als die Anzahl der Zeilen sein kann. Wir werden diese auch als Kontrollmatrizen von $\mathcal{C}$ bezeichnen.

5.3 Definition.
Sei $H = (h_{i,j})_{\substack{i=1,\ldots,m \\ j=1,\ldots,n}}$ eine Kontrollmatrix des binären $[n,k]$-Codes $\mathcal{C}$, also $\mathrm{rg}(H) = n-k$, $m \geq n-k$.
Definiere den *Tannergraphen* $\Gamma(H)$ zu H (manchmal $\Gamma(\mathcal{C})$, obwohl er durch $\mathcal{C}$ nicht eindeutig bestimmt ist) als bipartiten Graphen mit einer (linken) Eckenmenge $L = \{x_1, \ldots, x_n\}$ (Variablen) und einer (rechten) Eckenmenge $R = \{y_1, \ldots, y_m\}$ (Nebenbedingungen), und den Kanten $(x_i, y_j) \in V \times W$, falls $h_{j,i} = 1$.

5.4 Beispiel.
Sei $\mathcal{C}$ der $[7,4]$-Hamming-Code, $H = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$.
Dann ist $\Gamma(H)$ der folgende Graph:



Also $(x_1, \ldots, x_7) \in \mathcal{C} \Leftrightarrow$ alle Nebenbedingungen sind 0.

5.5 Bemerkung.
Jeder bipartite Graph bestimmt umgekehrt eine Matrix (nämlich seine Inzidenzmatrix, wenn die Zeilen mit R und die Spalten mit L indiziert sind), die als Kontrollmatrix einen Code bestimmt.
(Hier benötigt man die allgemeine Definition der Kontrollmatrix.)

Wir betracheten spezielle bipartite Graphen:

5.6 Definition.
Ein bipartiter Graph mit Eckenmengen L, R heißt

- *links-ℓ-regulär*, falls jede Ecke in L Grad ℓ hat
- *rechts-r-regulär*, falls jede Ecke in R Grad r hat
- *(ℓ, r)-regulär*, falls er links-ℓ-regulär und rechts-r-regulär ist.

Der Graph in 5.4 ist rechts 4-regulär, aber nicht beidseitig regulär.

5.7 Bemerkung.
Ist Γ ein bipartiter Graph mit $|L| = n$, $|R| = m$, so gilt:

- Ist Γ links-ℓ-regulär, so ist die Anzahl der Kanten $n \cdot \ell$.

- Ist Γ rechts-r-regulär, so ist die Anzahl der Kanten $m \cdot r$.
- Ist Γ (ℓ, r)-regulär, so ist $n \cdot \ell = m \cdot r$.

Beachte: Hat man eine Familie (ℓ, r)-regulärer bipartiter Graphen (ℓ fest) mit wachsendem n (und damit auch wachsendem m), so sind die Kontrollmatrizen immer dünner besetzt:
Anzahl aller Einträge: $m \cdot n$;
Anzahl der Einsen = Anzahl der Kanten $= m \cdot r = n \cdot \ell$;
Verhältnis: $\frac{r}{n} = \frac{\ell}{m} \to 0$.

5.8 Definition.

(a) Ein links-ℓ-regulärer Graph Γ mit den Eckenmengen L, R, $|L| = n$, $|R| = m$ heißt ein $(n, \ell, \alpha, \delta)$-*Expander*, wobei $\alpha, \delta > 0$, falls gilt:

$$\text{Ist } \varnothing \neq U \subseteq L \text{ mit } |U| \leq \alpha n, \text{ so ist } |\partial U| > \delta |U|.$$

Dabei bezeichnet ∂U die *Nachbarschaft* von U. d.h.

$$\partial U = \{y \in R : \exists x \in U,\ (x, y) \text{ ist Kante in } \Gamma\}.$$

Man nennt δ den *Ausdehnungsfaktor* des Expanders (bezüglich Teilmengen der Größe $\leq \alpha n$ in L).

(b) Ist Γ (ℓ, r)-regulär und erfüllt (a), so spricht man von einem $(n, \ell, r, \alpha, \delta)$-*Expander*.

(c) Ein binärer linearer Code, der eine Kontrollmatrix besitzt, die durch einen Expander definiert wird, wird *Expander-Code* $\mathcal{C}(\Gamma)$ genannt.

5.9 Bemerkung.
Bezeichnungen wie in 5.8(a), $\alpha n \geq 1$.

(a) $\delta < \ell$

(b) Ist $0 < \varepsilon < \frac{\ell}{\alpha n}$, so kann man stets $\delta = \left(\frac{\ell}{\alpha n} - \varepsilon\right)$ wählen.
(Man möchte zu gegebenem α natürlich δ möglichst groß wählen.)

Beweis.

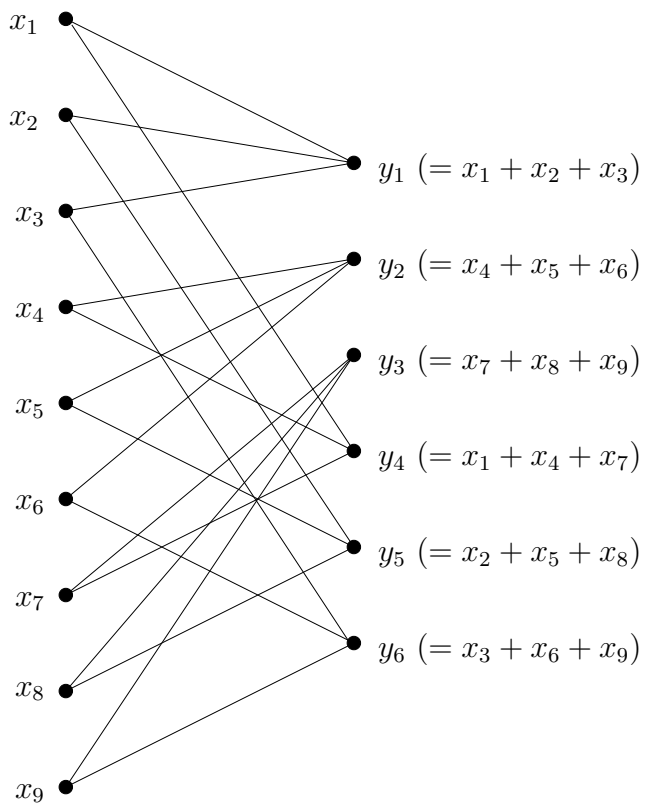
(a) Ist $|U| = 1$, so $|\partial U| = \ell$, also $\delta < \frac{|\partial U|}{|U|} = \ell$.

(b) Für alle $U \neq \varnothing$ ist $|\partial U| \geq \ell$.
Ist also $|U| \leq \alpha n$, so $\frac{|\partial U|}{|U|} \geq \frac{\ell}{\alpha n} > \frac{\ell}{\alpha n} - \varepsilon$.

□

5.10 Beispiel. (nach Willems, Codierungstheorie und Kryptographie)
Wir geben ein Beispiel für einen $(9, 2, 3, \frac{2}{9}, \delta)$-Expander an, wobei man für δ jede positive reelle Zahl $< \frac{3}{2}$ wählen kann.
(Beachte: Es gilt dann $m = \frac{n \cdot \ell}{r} = \frac{9 \cdot 2}{3} = 6$.)

Γ



Klar: Γ ist $(2,3)$-regulär.
$\alpha = \frac{2}{9}$. D.h. wir haben Eckenmengen U mit $|U| = 1, 2$ zu betrachten.
$|U| = 1 \Rightarrow |\partial U| = 2$
$|U| = 2 \Rightarrow |\partial U| = 3$ oder 4. Also $\min_{|U|\leq 2} \frac{|\partial U|}{|U|} = \frac{3}{2}$.
Jedes $0 < \delta < \frac{3}{2}$ ist also wählbar. (Beachte: $\frac{\ell}{\alpha n} = 1$.)

5.11 Lemma.
Sei Γ ein $(n, \ell, \alpha, \delta)$-Expander mit $\delta \geq \frac{\ell}{2}$. Sei $\varnothing \neq U \subseteq L$, $|U| \leq \alpha n$. Dann existiert ein $y \in \partial U$, in dem genau eine Kante aus U endet (d.h. $|\partial(y) \cap U| = 1$ für $\partial(y) := \{x \in L : (x, y)$ ist Kante$\}$).

Beweis.
In U beginnen $\ell|U|$ Kanten. Expander-Eigenschaft:
Diese Kanten enden in $|\partial U| > \delta|U| \geq \frac{\ell}{2}$ Ecken in R. In jeder dieser Ecken können nicht 2 oder mehr Kanten aus U enden, denn sonst würden die $\ell|U|$ Kanten aus U in höchstens $\frac{\ell}{2}|U|$ Ecken enden. Also existiert mindestens eine Ecke in ∂U, in der weniger als 2, also genau eine Kante aus U endet. □

5.12 Satz (Sipser, Spielman; 1996)**.**

(a) Sei Γ *ein* $(n, \ell, \alpha, \delta)$*-Expander mit* $\delta \geq \frac{\ell}{2}$.
Dann hat der zugehörige Expander-Code $\mathcal{C}(\Gamma)$ *Minimalabstand* $d > \alpha n$.
(Also relativen Minimalabstand $> \alpha$*)*

(b) Ist Γ *ein* $(n, \ell, r, \alpha, \delta)$*-Expander, so hat* $\mathcal{C}(\Gamma)$ *Rate* $R \geq 1 - \frac{\ell}{r}$.

Beweis.

(a) Angenommen $d = \mathrm{d}(\mathcal{C}(\Gamma)) \leq \alpha n$. Dann existiert ein $0 \neq v \in \mathcal{C}(\Gamma)$ mit $\mathrm{wt}(v) \leq \alpha n$. Sei U die Menge der x_i, die in v mit dem Wert 1 vorkommen (entspricht $\mathrm{Tr}(v)$). Dann ist $|U| = \mathrm{wt}(v) \leq \alpha n$.
Nach 5.11 existiert eine Nebenbedingung in ∂U, die mit genau einer Variablen aus U verbunden ist. Für diese Nebenbedingung hat v den Wert 1, also $v \notin \mathcal{C}(\Gamma)$. Widerspruch.

(b) $\mathcal{C}(\Gamma)$ wird durch $m = \frac{n \cdot \ell}{r}$ Gleichungen beschrieben, d.h. der Rang $n-k$ der Kontrollmatrix ist höchstens $\frac{n \cdot \ell}{r}$, also $k \geq n - \frac{n \cdot \ell}{r}$ ($k = \dim(\mathcal{C}(\Gamma))$).
Dann $R = \frac{k}{n} \geq 1 - \frac{\ell}{r}$

□

5.13 Beispiel.
Wir betrachten wieder das Beispiel 5.10.
Aus 5.12 folgt, dass die Rate von $\mathcal{C}(\Gamma) \geq 1 - \frac{2}{3} = \frac{1}{3}$ ist. Also ist $\dim(\mathcal{C}(\Gamma)) \geq 3$. Außerdem besagt 5.12, dass der Minimalabstand $> \alpha n = 2$ ist, also $\mathrm{d}(\mathcal{C}(\Gamma)) \geq 3$.
Man kann nachrechnen, dass die Kontrollmatrix zu Γ Rang 5 hat, also $\dim(\mathcal{C}(\Gamma)) = 9 - 5 = 4$. Ferner kann man zeigen, dass $\mathrm{d}(\mathcal{C}(\Gamma)) = 4$ gilt.

5.14. Iterative Decodierung (Flipping Algorithmus, Gallager, 1962)
Sei $\mathcal{C} = \mathcal{C}(\Gamma)$ ein Expander-Code zum $(n, \ell, \alpha, \delta)$-Expander Γ.
Ist $z = (z_1, \ldots, z_n) \in \mathbb{Z}_2^n$, so sagen wir, dass eine Nebenbedingung für z erfüllt ist, falls sie für z den Wert 0 hat. Ansonsten nennen wir sie inkorrekt.

Algorithmus:
Sei $z \in \mathbb{Z}_2^n$ das empfangene Wort.

Schritt 1: Berechne alle Nebenbedingungen für z.
Schritt 2: Suche eine Variable x_i, die in mehr inkorrekten Nebenbedingungen vorkommt als in korrekten. Gibt es keine solche Variable, so stop.
Schritt 3: Ändere den Wert dieser Variablen in z.
Schritt 4: Gehe zu Schritt 1.

5.15 Beispiel.
Betrachte den Expander-Code $\mathcal{C}$ aus Beispiel 5.10.
Sei $z = (110111011)$ empfangen worden.

Nebenbedingungen für z:

$$\begin{aligned} y_1 &= 0 \\ y_2 &= 1 \\ y_3 &= 0 \\ y_4 &= 0 \\ y_5 &= 1 \\ y_6 &= 0 \end{aligned}$$

x_5 kommt nur in inkorrekten Nebenbedingungen vor, nicht in korrekten.
Ändere x_5 von 1 zu 0. Das liefert das Wort (110101011). Jetzt sind alle Nebenbedingungen = 0. Das Wort liegt also in $\mathcal{C}$. Da $\mathcal{C}$ 1-Fehler-korrigierend ist (5.13), wurde korrekt decodiert, falls genau ein Fehler aufgetreten ist.

Frage: Wann decodiert der Flipping-Algorithmus korrekt?

5.16 Satz (Sipser, Spielman; 1996)**.**
Sei $\mathcal{C}(\Gamma)$ ein Expander-Code zum $(n, \ell, \alpha, \delta)$-Expander Γ.
Ist $\delta \geq \frac{3}{4}\ell$, so decodiert der Flipping-Algorithmus 5.14 korrekt, falls maximal $\lfloor \frac{\alpha n}{2} \rfloor$ Fehler aufgetreten sind.

Beweis.
Sei $z = (z_1, \ldots, z_n) \in \mathbb{Z}_2^n$ ein Wort, welches sich an höchstens $\lfloor \frac{\alpha n}{2} \rfloor$ Stellen von einem $x \in \mathcal{C}$ unterscheidet (beachte: x ist nach 5.12 eindeutig bestimmt, da $\mathrm{d}(\mathcal{C}) > \alpha n$).
Wir nennen die Variablen, in denen sich z von x unterscheidet *schlecht*, die anderen *gut*.
Zu Beginn der i-ten Iteration, $i = 1, 2, \ldots$, sei

t_i = Anzahl der schlechten Variablen
f_i = Anzahl der inkorrekten Nebenbedingungen
k_i = Anzahl der korrekten Nebenbedingungen, in denen schlechte Variablen vorkommen.

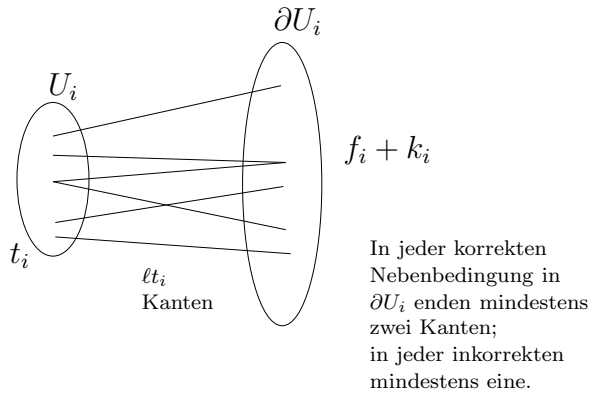
Beachte: $t_1 \leq \lfloor \frac{\alpha n}{2} \rfloor$.
Ist $t_1 = 0$, so ist z korrekt und der Algorithmus terminiert.
Angenommen, zu Beginn der i-ten Iteration ist $t_i \leq \alpha n = \alpha |L|$ und $t_i \neq 0$.
Die Menge $U_i \neq \varnothing$ der schlechten Variablen hat mehr als δt_i Nachbarn (Expandereigenschaft, $t_i \leq \alpha n$).

Jede dieser Nebenbedingungen ist entweder inkorrekt oder korrekt mit schlechten Variablen. Also:

$$\frac{3}{4}\ell t_i \leq \delta t_i < |\partial U_i| = f_i + k_i. \quad (1)$$

In einer korrekten Nebenbedingung, die schlechte Variablen enthält, müssen mindestens 2 schlechte Variablen vorkommen, in einer inkorrekten Nebenbedingung mindestens eine. Also:

$$\ell t_i \geq f_i + 2k_i. \quad (2)$$



(1) und (2) liefern:

$$\ell t_i \geq f_i + k_i + k_i > \frac{3}{4}\ell t_i + k_i \;\Rightarrow\; k_i < \frac{1}{4}\ell t_i \overset{(1)}{\Rightarrow} f_i > \frac{1}{2}\ell t_i \quad (3)$$

Dies besagt, dass eine schlechte Variable existiert, für die mehr als die Hälfte der Kanten in inkorrekten Nebenbedingungen landet.
In Schritt 2 des Algorithmus wird also eine Variable gefunden (nicht notwendig eine schlechte), die in mehr inkorrekten als korrekten Nebenbedingungen auftritt. Flippen dieser Variablen verringert den Wert von f_i, d.h. $f_{i+1} < f_i$. Wir zeigen jetzt, dass auch weiterhin $t_{i+1} \leq \alpha n$ gilt.
Beachte: $t_{i+1} = t_i \pm 1$ (Nur eine Stelle wird verändert).
Ist $t_{i+1} = t_1 - 1$, so ist $t_{i+1} \leq \alpha n$ erfüllt. Sei also $t_{i+1} = t_1 + 1$. Angenommen,

$$t_i + 1 > \alpha n \geq t_i, \text{ d.h. } \lfloor \alpha n \rfloor = t_i. \quad (*)$$

Beachte: $f_i \leq f_1 \overset{(2)}{\leq} \ell t_1 \overset{Vor.}{\leq} \ell \lfloor \frac{\alpha n}{2} \rfloor \leq \ell \frac{\lfloor \alpha n \rfloor}{2} \overset{(*)}{=} \frac{\ell t_i}{2} \overset{(3)}{<} f_i$. Widerspruch.
Also gilt auch $t_{i+1} \leq \alpha n$.
Da $f_{i+1} < f_i$ für alle i, terminiert der Algorithmus (und zwar in einem Codewort). □

5.17 Bemerkung.

(a) Der Algorithmus in 5.14 erfordert maximal m Iterationsschritte, da $f_1 \leq m$ und $f_{i+1} < f_i$ für alle i. Zu jedem Iterationsschritt werden m Bilinearformen (Nebenbedingungen) berechnet und maximal $n \cdot m$ Vergleiche durchgeführt.

(b) Das iterative Decodierverfahren aus 5.14 ist ein BDD (vergleiche 2.18 (b)) nach 5.16. Er decodiert garantiert korrekt, wenn maximal $\lfloor \frac{\alpha n}{2} \rfloor$ Fehler aufgetreten sind. Diese Zahl kann kleiner als $\lfloor \frac{d-1}{2} \rfloor$ sein (vergleiche Beispiel 5.13, dort $\alpha n = 2$, d $= 4$; dennoch gilt hier $\lfloor \frac{\alpha n}{2} \rfloor = \lfloor \frac{\mathrm{d}-1}{2} \rfloor = 1$.)

(c) Die Bedingung $\delta \geq \frac{3}{4}\ell$ aus 5.14 ist bei unserem Beispiel 5.15 (Code aus 5.10) nicht erfüllt. Dort ist $\delta < \frac{3}{2} = \frac{3}{4} \cdot 2$. Dennoch wird in 5.15 korrekt decodiert.

(d) Die Bedingung in Schritt 2 des Algorithmus 5.14, wann z_i geändert wird, ist äquivalent zu der Bedingung

$$\underbrace{\mathrm{wt}\left(H(z+e_i)^t\right)}_{\substack{\text{Anzahl der inkorrekten} \\ \text{Nebenbed. in } z+e_i}} < \underbrace{\mathrm{wt}\left(Hz^t\right)}_{\substack{\text{Anzahl der inkorrekten} \\ \text{Nebenbed. in } z}}$$

(e) Der Algorithmus 5.14 ist ein Spezialfall von Verfahren, die man *Belief-Propagation-Algorithmus* nennt (sie spielen auch in KI und anderen Bereichen eine wichtige Rolle): In jedem Iterationsschritt wird derjenige Eintrag geändert, von dem man am stärksten glaubt, dass er falsch ist.

5.18 Bemerkung.

(a) Nach 5.12 und 5.14 benötigt man Familien von Expander-Codes mit großem δ ($\delta \geq \frac{3}{4}\ell$), damit das Decodierverfahren funktioniert, und nicht zu kleinem α, damit möglichst viele Fehler korrigiert werden können.
Es gibt Möglichkeiten solche Codes mit nach unten beschränktem α und wachsendem n zu konstruieren.
Sie stammen z.B. von Margulis (1982), Lubotzky, Phillips, Sarnak (1988), u.a.
Die Beschreibung ist relativ einfach, aber der Nachweis der Expander-Eigenschaft kompliziert.

(b) Der Decodier-Algorithmus 5.14 ist umso schneller, je kleiner ℓ ist, denn dann hat man nur zu kontrollieren, ob eine der in den inkorrekten Nebenbedingungen auftretenden Variablen dort mehr als $\frac{\ell}{2}$ mal auftaucht.

Also sind Familien von Expander-Codes mit wachsendem n und konstantem ℓ gut. Das ist bei obiger Familie der Fall.
Dann: Das Verhältnis von Einsen in der Konrollmatrix zu allen Einträgen $\frac{n \cdot \ell}{n^2} = \frac{\ell}{n}$ geht für $n \to \infty$ gegen 0.
Dünn besetzte Kontrollmatrix: LDPC-Codes.
Dies ist ein echter Hinweis auf die Nützlichkeit von LDPC-Codes für die Decodiergeschwindigkeit.
Gallager (1962) hat dies als erster entdeckt; auf ihn gehen Message-Passing-Algorithmen für LDPC-Codes zurück, die wir aber hier nicht behandeln werden.

Literatur:

Lin, Costello, Error Control Coding (Kap. 17) (LDPC)
Willems, Codierungstheorie und Kryptographie (Kap. 4)
T. Richardson, R. Urbanke, Modern Coding Theory (Kap. 8) (viel über LDPC-Codes)

Übersichtsartikel über Expander-Graphen:

S. Hoory, N. Linial, A. Wigderson, Bulletin American Math. Society 43, 2006, 439-561

6 Zyklische Codes

6.1 Definition.
Ein linearer Code $\mathcal{C}$ der Länge n über einem endlichen Körper K heißt *zyklisch*, falls gilt:

$$(c_0, \ldots, c_{n-1}) \in \mathcal{C} \Rightarrow \sigma(c_0, \ldots, c_{n-1}) := (c_{n-1}, c_0, c_1, \ldots, c_{n-2}) \in \mathcal{C}.$$

(Dann auch $\sigma^i(c_0, \ldots, c_{n-1}) = (c_{n-i}, \ldots, c_{n-1}, c_0, \ldots, c_{n-1-i}) \in \mathcal{C}$.)

6.2 Beispiele.

(a) Sei $\mathcal{C}$ der $[7, 4]$-Hamming-Codes über $\mathbb{Z}_2$ mit der Kontrollmatrix

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

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

$\mathcal{C}$ ist nicht zyklisch:
$(0110100) \in \mathcal{C}$ (2. Zeile von G). Angenommen, $x = (0011010) \in \mathcal{C}$.
$y = (1010010) \in \mathcal{C}$ (3. Zeile von G). Aber: $\mathrm{d}(x, y) = 2$. Das steht im Widerspruch zu $\mathrm{d}(\mathcal{C}) = 3$.

(b) Man nennt zwei binäre Codes $\mathcal{C}_1$, $\mathcal{C}_2$ *äquivalent*, wenn $\mathcal{C}_2$ aus $\mathcal{C}_1$ durch eine Permutation der Positionen hervorgeht (entspricht einer Permutation der Spalten der Erzeugermatrix).
[Alle Parameter eines Codes bleiben bei Äquivalenz erhalten.]
Tatsächlich gibt es zu jedem binären Hamming-Code einen äquivalenten Code, der zyklisch ist. Im obigen Beispiel:

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

Zyklische Vertauschung von Zeile 1 von $\tilde{G}$ = Zeile 2
Zyklische Vertauschung von Zeile 2 von $\tilde{G}$ = Zeile 1 + Zeile 4
Zyklische Vertauschung von Zeile 3 von $\tilde{G}$ = Zeile 1
Zyklische Vertauschung von Zeile 4 von $\tilde{G}$ = Zeile 1 + Zeile 3

(Das gilt für jeden Hamming-Code; bei $|K| > 2$ muss der Äquivalenzbegriff erweitert werden: Die Spalten der Erzeugermatrix dürfen auch mit Skalaren $\neq 0$ multipliziert werden.)

Erste gute Eigenschaft zyklischer Codes:

6.3 Bemerkung.
Sei $\mathcal{C}$ ein zyklischer Code.

(a) $\mathcal{C}^\perp$ ist zyklisch.

(b) Sind $y^{(1)}, \ldots, y^{(\ell)} \in \mathcal{C}^\perp$ orthogonal bezüglich Position i, so sind $\sigma(y^{(1)}), \ldots, \sigma(y^{(\ell)}) \in \mathcal{C}^\perp$ orthogonal bezüglich $i+1 \pmod n$.
(Positionen $0, \ldots, n-1$)

Beweis.

(a) $y = (y_0, \ldots, y_{n-1}) \in \mathcal{C}^\perp$, $(c_0, \ldots, c_{n-1}) \in \mathcal{C}$.
Dann $\sigma^{n-1}(c) = (c_1, \ldots, c_{n-1}, c_0) \in \mathcal{C}$.
$0 = \langle \sigma^{n-1}(c), y\rangle = c_1 y_0 + \ldots + c_{n-1} y_{n-2} + c_0 y_{n-1} = \langle c, \sigma(y)\rangle$
$\Rightarrow \sigma(y) \in \mathcal{C}^\perp$.

(b) folgt aus (a).

□

Zyklische Codes lassen sich besonders gut beschreiben, wenn man Polynomdarstellung verwendet. Betrachte dazu den Vektorraumisomorphismus

$$\begin{aligned} K^n &\to K[x]_n = \{f \in K[x] : \operatorname{grad}(f) < n\} \\ a = (a_0, \ldots, a_{n-1}) &\mapsto a(x) = \sum_{i=0}^{n-1} a_i x^i. \end{aligned}$$

6.4 Bemerkung.

(a) Zur Erinnerung: In $K[x]$ hat man Division mit Rest.
Seien $f, g \in K[x]$, $g \neq 0$. Dann gibt es eindeutig bestimmte $r, h \in K[x]$ mit $\operatorname{grad}(r) < \operatorname{grad}(g)$ so dass $f = h \cdot g + r$ gilt.
Schreibe dann $r = f \bmod g$.

Beispiel: $K = \mathbb{Z}_2$. Bestimme $x^4 + x^3 + 1 \bmod x^2 + 1$.

$x^4 + x^3 + 1 : x^2 + 1 = x^2 + x + 1$
$\underline{x^4 + x^2}$
$x^3 + x^2 + 1$
$\underline{x^3 + x}$
$x^2 + x + 1$
$\underline{x^2 + 1}$
x

Daher ist $x^4 + x^3 + 1 \bmod x^2 + 1 = x$.

Es gilt: $f_1 \stackrel{+}{\cdot} f_2 \bmod g = (f_1 \bmod g \stackrel{+}{\cdot} f_2 \bmod g) \bmod g$.

(b) $a = (a_0, \ldots, a_{n-1}) \in K^n \leftrightarrow a(x) = \sum_{i=0}^{n-1} a_i x^i$
Dann:
$\sigma(a) = (a_{n-1}, a_0, \ldots, a_{n-2}) \leftrightarrow x \cdot a(x) \bmod x^n - 1$.

Beweis.
$x \cdot a(x) = a_0 x + \ldots + a_{n-1} x^n = a_{n-1} + a_0 x + \ldots + a_{n-2} x^{n-1} + a_{n-1}(x^n - 1)$
Also: $x \cdot a(x) \bmod x^n - 1 = a_{n-1} + a_0 x + \ldots + a_{n-2} x^{n-1} \leftrightarrow \sigma(a)$.
(modulo $x^n - 1$ Rechnen bedeutet: Ersetze x^n durch 1.)

6.5 Satz.
Sei $\mathcal{C}$ ein Code der Länge n über K. Sei $\tilde{\mathcal{C}} = \{c(x) : c \in \mathcal{C}\} \subseteq K[x]_n$ (entsprechend obiger Bemerkung). Dann gilt:

$$\mathcal{C} \text{ ist zyklisch } \Leftrightarrow f(x)c(x) \bmod x^n - 1 \in \tilde{\mathcal{C}} \text{ für alle } f(x) \in K[x],\ c(x) \in \tilde{\mathcal{C}}.$$

Beweis.
„$\Leftarrow$“:
Insbesondere gilt $x \cdot c(x) \bmod x^n - 1 \in \tilde{\mathcal{C}}$ für alle $c(x) \in \tilde{\mathcal{C}}$. Daher gilt $\sigma(c) \in \mathcal{C}$ für alle $c \in \mathcal{C}$ nach 6.4.(b). $\mathcal{C}$ ist also zyklisch.
„$\Rightarrow$“:
6.4(b): $x \cdot c(x) \bmod x^n - 1 \in \tilde{\mathcal{C}}$ für alle $c(x) \in \tilde{\mathcal{C}}$
$\Rightarrow x^i \cdot c(x) \bmod x^n - 1 \in \tilde{\mathcal{C}}$ für alle $c(x) \in \tilde{\mathcal{C}}$
$\Rightarrow$ Behauptung, da $\mathcal{C}$ linear. □

Sei $\mathcal{C} \neq \{0\}$ ein zyklischer Code der Länge n.
Wähle in $\tilde{\mathcal{C}}$ ein normiertes Polynom $g(x) \neq 0$ von minimalem Grad.
Angenommen es gibt ein weiteres normiertes $g_1(x) \in \tilde{\mathcal{C}}$ mit $\operatorname{grad}(g_1) = \operatorname{grad}(g)$. Dann ist $g(x) - g_1(x) \in \tilde{\mathcal{C}}$, $\operatorname{grad}(g - g_1) < \operatorname{grad}(g)$, also $g - g_1 = 0$, $g = g_1$.
g ist also eindeutig bestimmt.
Wir behaupten, dass g ein Teiler von $x^n - 1$ ist.
$x^n - 1 = h(x)g(x) + r(x)$, $\operatorname{grad}(r(x)) < \operatorname{grad}(g(x))$

$h(x)g(x) \bmod x^n - 1 = -r(x) \bmod (x^n - 1) = -r(x)$
Nach 6.5 folgt $r(x) \in \tilde{\mathcal{C}}$. Ist $r(x) \neq 0$, so ergibt sich ein Widerspruch zur Wahl von g.
Also: $x^n - 1 = g(x)h(x)$. $h(x)$ ist normiert und eindeutig bestimmt.
Damit haben wir die ersten beiden Teile des folgenden Satzes bewiesen:

6.6 Satz.
Sei $\mathcal{C}$ ein zyklischer Code der Länge n über K. Sei $0 \neq g(x) \in \tilde{\mathcal{C}}$ normiert und von minimalem Grad in $\tilde{\mathcal{C}}$. Sei $\operatorname{grad}(g(x)) = \ell$ *(also $\ell < n$).*

(a) $g(x)$ *ist eindeutig bestimmt*

(b) $x^n - 1 = g(x)h(x)$, $h(x)$ *ist normiert und eindeutig bestimmt,* $\operatorname{grad}(h(x)) = n - \ell$

(c) $\tilde{\mathcal{C}} = \{f(x)g(x) : f(x) \in K[x], \operatorname{grad}(f(x)) < n - \ell\}$

(d) $\dim(\mathcal{C}) = \dim(\tilde{\mathcal{C}}) = n - \ell$, *d.h.* $\operatorname{grad}(g(x)) = n - k$, $k = \dim(\mathcal{C})$.

(e) $c(x) \in \widetilde{\mathcal{C}} \Leftrightarrow c(x) \cdot h(x) = 0 \bmod x^n - 1$

Beweis.

(a),(b) siehe oben

(c) Nach 6.5 gilt „$\supseteq$“, da $\operatorname{grad}(f(x) \cdot g(x)) < n$.
Sei $0 \neq c(x) \in \tilde{\mathcal{C}}$. Division mit Rest: $c(x) = k(x) \cdot g(x) + r(x)$, $\operatorname{grad}(r(x)) < \operatorname{grad}(g(x)) = \ell$.
$r(x) = c(x) - \underbrace{k(x) \cdot g(x)}_{\substack{\operatorname{grad}(c(x)) \leq n-1, \\ \text{also} \in \tilde{\mathcal{C}} \text{ nach 6.5}}} \in \tilde{\mathcal{C}} \underset{\substack{\text{Wahl von} \\ g}}{\Longrightarrow} r(x) = 0.$
$\operatorname{grad}(k(x)) + \operatorname{grad}(g(x)) = \operatorname{grad}(c(x)) \leq n-1 \Rightarrow \operatorname{grad}(k(x)) \leq n-\ell-1$.
Also gilt auch „$\subseteq$“.

(d) Wegen (c) ist $g(x), x \cdot g(x), \ldots, x^{n-\ell-1} \cdot g(x)$ eine Basis von $\tilde{\mathcal{C}}$. Die Behauptung folgt.

(e) Folgt aus $x^n - 1 = g(x) \cdot h(x)$ und $c(x) \in \mathcal{C} \Leftrightarrow g(x) | c(x)$.

□

6.7 Definition.
Sei $\mathcal{C} \neq \{0\}$ ein zyklischer Code.
Dann heißen die eindeutig bestimmten normierten Polynome $g(x)$, $h(x)$ aus 6.6 *Erzeugerpolynom (Generatorpolynom)* bzw. *Kontrollpolynom* von $\mathcal{C}$.

6.8 Satz.
Sei $\mathcal{C} \neq \{0\}$ ein zyklischer $[n,k]$-Code, $g(x) = \sum_{i=0}^{n-k} g_i x^i$ das Erzeugerpolynom und $h(x) = \sum_{i=0}^{k} h_i x^i$ das Kontrollpolynom von $\mathcal{C}$.

(a) Die $k \times n$-Matrix

$$G = \begin{pmatrix} g_0 & g_1 & \dots & g_{n-k} & & & \\ 0 & g_0 & \dots & \dots & g_{n-k} & & 0 \\ \vdots & & \ddots & & & \ddots & \\ 0 & \dots & 0 & g_0 & \dots & \dots & g_{n-k} \end{pmatrix}$$

ist eine Erzeugermatrix von $\mathcal{C}$.
($g_{n-k} = 1$)

(b) Die $(n-k) \times n$-Matrix

$$H = \begin{pmatrix} h_k & h_{k-1} & \dots & h_0 & & & \\ 0 & h_k & \dots & \dots & h_0 & & 0 \\ \vdots & & \ddots & & & \ddots & \\ 0 & \dots & 0 & h_k & \dots & \dots & h_0 \end{pmatrix}$$

ist eine Kontrollmatrix von $\mathcal{C}$.
($h_k = 1$)

Beweis.

(a) folgt aus 6.6 (d).

(b) $x^n - 1 = g(x) \cdot h(x) = \sum_{i=0}^{n} \left(\sum_{j=0}^{n-k} g_i h_{i-j} \right) x^i$ (alle nicht definierten Koeffizienten gleich 0 setzen)
Koeffizientenvergleich liefert:

$(*)$ $\sum_{j=0}^{n-k} g_i h_{i-j} = 0$ für $i = 1, \dots, n-1$

$(**)$ $g_0 h_0 = -1$

$(*)$ bedeutet gerade $GH^t = 0$.
Wegen $(**)$ ist $\mathrm{rg}(H) = n - k$. Also ist H Kontrollmatrix von $\mathcal{C}$.

□

6.9 Bemerkung.
Sei $\mathcal{C} \neq \{0\}$ ein zyklischer $[n,k]$-Code über K. Nach 3.28 (c) ist H aus 6.8 (b) eine Erzeugermatrix von $\mathcal{C}^\perp$, $\mathcal{C}^\perp$ ist zyklisch nach 6.3 (a). Dann ist auch $h_0^{-1}H$ eine Erzeugermatrix von $\mathcal{C}^\perp$ und das Erzeugerpolynom von $\mathcal{C}^\perp$ ist gerade

$$x^k + \frac{h_1}{h_0}x^{k-1} + \ldots + \frac{h_{k-1}}{h_0}x + \frac{h_k}{h_0} = h_0^{-1} \underbrace{x^k h\left(\frac{1}{x}\right)}_{=: \text{ duales Poynom zu } h} \quad .$$

6.10 Korollar.
Es gibt genauso viele zyklische Codes der Länge n über K wie es (normierte) Teiler von $x^n - 1$ gibt.

Beweis.
Ist $\mathcal{C}$ zyklisch, so gehört zu $\mathcal{C}$ nach 6.6 ein eindeutig bestimmtes Erzeugerpolynom $g(x)$, $g(x)|x^n - 1$.
Umgekehrt kann man zu jedem Teiler $g(x)$ von $x^n - 1$ den Code
$\widetilde{\mathcal{C}} = \{f(x) \cdot g(x) : f(x) \in K[x],\ \operatorname{grad}(f(x)) < n - \operatorname{grad}(g(x))\}$ bilden. Dies ist ein zyklischer Code mit Erzeugerpolynom $g(x)$.
(Ist $\mathcal{C} = \{0\}$, so ist $g(x) = x^n - 1$.) □

6.11 Beispiel.
$n = 4$, $K = \mathbb{Z}_2$. Es gibt die folgenden zyklischen Codes der Länge 4 über $\mathbb{Z}_2$:

Teiler von $x^4 - 1 = (x-1)^4$ in $K[x]$	zugehöriger zyklischer Code
1	$\mathcal{C}_1 = \mathbb{Z}_2^4$
$(x-1)$	Erzeugermatrix $\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$, $\dim \mathcal{C}_2 = 3$ (Parity-Check-Code)
$(x-1)^2 = x^2 - 1$	Erzeugermatrix $\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$, $\dim \mathcal{C}_3 = 2$
$(x-1)^3 = x^3 + x^2 + x + 1$	Erzeugermatrix (1111), $\dim \mathcal{C}_4 = 1$
$(x-1)^4 = x^4 - 1$	$\mathcal{C}_5 = \{0\}$

6.12 Definition.

(a) Ein Vektor $v = (0, \ldots, 0, v_i, \ldots, v_{i+b-1}, 0, \ldots, 0) \in K^n$ mit $v_i \neq 0$, $v_{i+b-1} \neq 0$ heißt *Bündel* der Länge b.
(Beachte: Nicht alle v_j, $2 \leq j \leq i+b-2$, müssen $\neq 0$ sein.)

(b) Sei $\mathcal{C}$ ein Code über K, $x \in \mathcal{C}$ gesendet und $z \in K^n$ empfangen. Sei $z = x + f$, wobei f ein Bündel der Länge b ist (Fehlerbündel), so spricht man von einem *Bündelfehler* der Länge b.

(Beachte: Bei einem Bündelfehler der Länge b können auch weniger als b Fehler aufgetreten sein.)

6.13 Satz.
Sei $\mathcal{C}$ ein zyklischer $[n,k]$-Code. Dann enthält $\mathcal{C}$ keine Bündel der Länge $\leq n-k$.
Insbesondere entdeckt ein zyklischer $[n,k]$-Code Fehlerbündel der Länge $\leq n-k$.

Beweis.
Angenommen $(0, \ldots, 0, v_i, \ldots, v_{i+b-1}, 0, \ldots, 0) \in \mathcal{C}$, $v_i, v_{i+b-1} \neq 0$, $b \leq n-k$.
Dann ist auch $v = (v_i, \ldots, v_{i+b-1}, \underbrace{0, \ldots, 0}_{\geq k}) \in \mathcal{C}$.
Die b-te Zeile der Kontrollmatrix aus 6.8 (b) ist

$$h = (\underbrace{0, \ldots, 0}_{b-1}, \underbrace{h_k, \ldots, h_0}_{k+1}, 0, \ldots, 0) \quad (b - 1 + (k+1) \leq n)$$

$hv^t = h_k \cdot v_{i+b-1} \neq 0$, also $Hv^t \neq 0$. Widerspruch.
Ist $z = x + f$, f Fehlerbündel der Länge $\leq n-k$, so gilt $Hz^t = Hf^t \neq 0$, da $f \notin \mathcal{C}$. □

6.14 Bemerkung.
Für einen linearen $[n,k,d]$-Code gilt nach 3.14 (Singleton-Schranke)

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

Also kann ein linearer Code maximal $n-k$ Fehler erkennen (Schranke wird für MDS-Codes angenommen).
Demgegenüber erkennt jeder zyklische Code Fehlerbündel der Länge $n-k$, gleichgültig, wie groß d ist.

Die Erzeugermatrix G aus 6.8 (b) ist nicht in Standardform. Bildet man Codewörter durch uG, $u \in K^k$, so lässt sich u aus uG nicht direkt ablesen. Es gibt eine andere Form der Codierung bei zyklischen Codes, wo das möglich ist.

6.15. CRC-Codierung (CRC=Cyclic Redundancy Check)
Sei $\mathcal{C}$ ein zyklischer $[n,k]$-Code über K mit Erzeugerpolynom vom Grad $n-k$. Infoblöcke der Länge k (über K) werden als Polynome vom Grad $\leq k-1$ geschrieben:

$$m = m_0 \dots m_{k-1} \leftrightarrow m(x) = \sum_{i=0}^{k-1} m_i x^i$$

$$x^{n-k} m(x) \leftrightarrow (\underbrace{0, \dots, 0}_{n-k}, m_0, \dots, m_{k-1})$$

Codiere $m(x)$ in $c(x) = x^{n-k}m(x) - \big(x^{n-k}m(x) \mod g(x)\big)$ (das ist ein Vielfaches von $g(x)$).
Also: $c(x) \in \mathcal{C}$ nach 6.6 (c).
Ist $x^{n-k}m(x) \mod g(x) = b_0 + b_1 x + \ldots + b_{n-k-1}x^{n-k-1}$, so

$$c(x) \leftrightarrow (\underbrace{-b_0, -b_1, \dots, -b_{n-k-1}}_{\text{Kontrollsymbole}}, \underbrace{m_0, \dots, m_k}_{\text{Infosymbole}}).$$

Beispiel: $n = 4$, $k = 3$, $g(x) = x + 1$
$m = (m_0, m_1, m_2) = (1,1,1) \leftrightarrow m(x) = x^2 + x + 1$
$xm(x) = x^3 + x^2 + x \quad x^3 + x^2 + x : x + 1 = x^2 + 1$
$\underline{x^3 + x^2}$
x
$\underline{x + 1}$
1

$xm(x) \mod g(x) = 1$
$c(x) = 1 + x + x^2 + x^3 \leftrightarrow (1,1,1,1)$
Beachte: Erzeugermatrix nach 6.8 (b): $G = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$, $mG = (1001)$.

Erhält der Empfänger das Wort v bzw. $v(x)$ mit $g(x) \nmid v(x)$, so sind nach 6.6 (c) Fehler aufgetreten.
Ist kein Fehler aufgetreten, so kann er die Nachricht direkt an den letzten k Symbolen ablesen.
Die Berechnung von $x^{n-k}m(x) \mod g(x)$ geht sehr schnell, z.B. mit linearen Schieberegistern.

CRC-Codierung wird in Computer-Netzen verwendet, wenn es nur auf Fehlererkennung ankommt (ARQ-Verfahren). Dort werden häufig Polynome der Form $g(x) = (x+1)p(x)$, $p(x) \neq x$ irreduzibel, verwendet. Z.B.

$$\underbrace{g(x) = x^{16} + x^{12} + x^5 + 1}_{CRC-CCITT-Polynom} \text{ oder } \underbrace{g(x) = x^{16} + x^{15} + x^2 + 1}_{CRC-16-Polynom}$$

(CCITT=Comité Consultativ International de Télégraphique et Téléphonique; Vorläufer von ITU)
Mit solchen Erzeugerpolynomen werden neben Fehlerbündeln der Länge $\leq$ grad$(g(x))$ (6.13) auch eine ungerade Anzahl von Fehlern erkannt und häufig auch 2 Fehler.

7 Reed-Solomon-Codes

7.1 Definition (Allgemeine Reed-Solomon-Codes)**.**
Sei K ein Körper, $|K| = q$, $1 \leq d \leq n \leq q$.
Wähle $\mathcal{M} = \{a_1, \ldots, a_n\} \subseteq K$, $|\mathcal{M}| = n$ (d.h. $a_i \neq a_j$ für $i \neq j$).
Setze $\mathcal{C} = \mathcal{C}_{\mathcal{M}} = \{f(a_1), \ldots, f(a_n) : f \in K[x]_{n-d+1}\} \subseteq K^n$, wobei $K[x]_{n-d+1} = \{g \in K[x] : \operatorname{grad}(g) \leq n - d\}$.
$\mathcal{C}$ heißt *Allgemeiner Reed-Solomon-Code.* ($\mathcal{C}$ ist ein Auswertungscode.)

7.2 Satz.
Verwende die Bezeichnungen aus 7.1 und setze $k = n - d + 1$.

(a) $\mathcal{C}$ ist ein $[n, k]$-Code mit $\mathrm{d}(\mathcal{C}) = d$, also ein MDS-Code.

(b) $G = \begin{pmatrix} 1 & \ldots & 1 \\ a_1 & \ldots & a_n \\ \vdots & & \vdots \\ a_1^{k-1} & \ldots & a_n^{k-1} \end{pmatrix}$ *ist Erzeugermatrix von $\mathcal{C}$.*

Beweis.

(a) Sei $\alpha : \begin{cases} K[x] & \to \mathcal{C} \\ f & \mapsto (f(a_1), \ldots, f(a_n)) \end{cases}$.
α ist eine surjektive K-lineare Abbildung.
Angenommen $f \in \operatorname{Kern}(\alpha)$, d.h. $f(a_1) = \ldots = f(a_n) = 0$.
Da $\operatorname{grad}(f) \leq k - 1 < n$, kann f aber nicht n verschiedene Nullstellen haben, falls $f \neq 0$. Damit ist $f = 0$.
Also ist f bijektiv. Daher gilt $\dim \mathcal{C} = \dim K[x]_k = k$.
Jedes $f \neq 0$, $f \in K[X]_k$, hat höchstens $k - 1$ Nullstellen in K, also sind in jedem Codewort $\neq 0$ mindestens $n - (k - 1) = d$ Einträge $\neq 0$.
Nach der Singleton-Schranke (3.14): $\mathrm{d}(\mathcal{C}) \leq n - k + 1 = d$. Es gilt also Gleichheit.

(b) $1, x, \ldots, x^{k-1}$ ist eine Basis von $K[x]_k$. Daher ist $\underbrace{\alpha(1)}_{\substack{1.\text{ Zeile} \\ \text{von } G}}, \ldots, \underbrace{\alpha(x^{k-1})}_{\substack{k.\text{ Zeile} \\ \text{von } G}}$ eine Basis von $\mathcal{C}$.

□

Wir betrachten nun einen Spezialfall von 7.1. Dazu:

7.3 Bemerkung.
Sei K ein endlicher Körper, $|K| = q$. Dann ist die multiplikative Gruppe $K^* = K \setminus \{0\}$ zyklisch, d,h, es existiert ein $\alpha \in K^*$ mit

$$K^* = \{\alpha^0 = 1, \alpha^1, \alpha^2, \ldots, \alpha^{q-2}\}$$

($\alpha^{q-1} = 1$). Die Potenzen von α werden durch die Multiplikation in K gebildet.

7.4 Definition.
Sei K ein endlicher Körper, $|K| = q$, $q \geq 3$, $n = q - 1$.
Sei $d \in \mathbb{N}$ mit $2 \leq d \leq n - 1$.
Setze $\mathcal{M} = K^* = \{\alpha^0, \alpha^1, \ldots, \alpha^{q-2}\}$ (geordnet).
Dann heißt

$$\mathcal{C}_{\mathcal{M}} = \{\big(f(\alpha^0), \ldots, f(\alpha^{q-2})\big) : f \in K[x]_{n-d+1} = K[x]_{q-d}\}$$

Reed-Solomon-Code $RS_q(d)$.
($d = 1$, $d = n$ liefern triviale Codes; daher werden sie hier ausgeschlossen.)

7.5 Satz.
$\mathcal{C} = RS_q(d)$ ist ein zyklischer $[n, n - d + 1]$-Code mit $\mathrm{d}(\mathcal{C}) = d$.
$g(x) = (x - \alpha) \cdot \ldots \cdot (x - \alpha^{d-1})$ ist das Erzeugerpolynom und
$h(x) = (x - 1) \prod_{j=d}^{n-1} (x - \alpha^j)$ ist das Kontrollpolynom von $\mathcal{C}$.

$$H = \begin{pmatrix} 1 & \alpha & \alpha^2 & \ldots & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^4 & \ldots & \alpha^{(n-1)\cdot 2} \\ 1 & \alpha^3 & \alpha^6 & \ldots & \alpha^{(n-1)\cdot 3} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \alpha^{d-1} & \alpha^{2(d-1)} & \ldots & \alpha^{(n-1)\cdot(d-1)} \end{pmatrix}$$

ist eine Kontrollmatrix von $\mathcal{C}$.
(Das ist nicht die Kontrollmatrix zu $h(x)$ aus 6.8 (b).)

Beweis.
Dass $\mathcal{C}$ ein $[n, n-d+1, d]$-Code ist, folgt aus 7.2 (setze $k = n-d+1 = \dim(\mathcal{C})$).
Sei $f \in K[x]_k$. Dann ist $(f(\alpha^0), \ldots, f(\alpha^{q-2})) \in \mathcal{C}$.
$(f(\alpha^{q-2}), f(\alpha^0), \ldots, f(\alpha^{q-3})) = (h(\alpha^0), \ldots, h(\alpha^{q-2}))$ mit $h(x) = f(\alpha^{-1}x) \in K[x]_k$ ($\alpha^{-1} = \alpha^{q-2}$). Also ist $\mathcal{C}$ zyklisch.
Allgemein gilt:

$$x^n - 1 = x^{q-1} - 1 = \prod_{j=0}^{q-2} (x - \alpha^j). \qquad (*)$$

(Denn jedes α^j ist wegen $(\alpha^j)^{q-1} = (\alpha^{q-1})^j = 1$ eine Nullstelle von $x^{q-1} - 1$.)
Sei $\widetilde{\mathcal{C}}$ ein zyklischer Code mit Erzeugerpolynom $g(x)$. Beachte: $g(x) | x^n - 1$.

Grad $g(x) = d-1$, also $\dim(\tilde{\mathcal{C}}) = n-(d-1) = k = \dim(\mathcal{C})$.
Nun gilt $c(x) = \sum_{i=0}^{n-1} c_i x^i \in \widetilde{\mathcal{C}} \Leftrightarrow g(x)|c(x) \Leftrightarrow x-\alpha^j|c(x)$ für alle $j = 1,\ldots,d-1 \Leftrightarrow c(\alpha^i) = c_0+c_1\alpha^j+\ldots+c_{n-1}\alpha^{j(n-1)} = 0$ für alle $j = 1,\ldots,d-1$.
Also ist $H = \begin{pmatrix} 1 & \alpha & \alpha^2 & \ldots & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^4 & \ldots & \alpha^{(n-1)\cdot 2} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \alpha^2 & \alpha^4 & \ldots & \alpha^{(n-1)\cdot 2} \end{pmatrix}$ eine Kontrollmatrix von $\tilde{\mathcal{C}}$.
Wir zeigen nun: H ist auch eine Kontrollmatrix von $\mathcal{C}$. Dann ist nämlich $\widetilde{\mathcal{C}} = \mathcal{C}$ und wir sind fertig.
Sei $c = (f(1), f(\alpha), \ldots, f(\alpha^{n-1})) \in \mathcal{C}$ $(n-1 = q-2)$ für $f(x) = \sum_{j=0}^{n-d} a_j x^j$.
Multipliziere die i-te Zeile von H mit c^t. Das liefert:

$$\begin{aligned} \sum_{s=0}^{n-1} f(\alpha^s)\alpha^{is} &= \sum_{j=0}^{n-d} a_j + \sum_{j=0}^{n-d} a_j\alpha^{j+i} + \sum_{j=0}^{n-d} a_j\alpha^{2(j+i)} + \ldots + \sum_{j=0}^{n-d} a_j\alpha^{(n-1)(j+i)} \\ &= \sum_{j=0}^{n-d} a_j[1+\alpha^{j+i}+\alpha^{2(j+i)}+\ldots+\alpha^{(n-1)(j+i)})] \end{aligned}$$

Nun ist $\alpha^{j+i} \neq 1$, da $0 \leq j \leq n-d$ und $1 \leq i \leq d-1$, also $1 \leq j+i \leq n-1 = q-2$.
$(\alpha^{j+1})^{q-1} = (\alpha^{q-1})^{j+1} = 1$.
Also ist α^{j+i} eine Nullstelle des Polynoms $x^{q-1}-1 \in K[x]$.
$x^{q-1}-1 = (x-1)(x^{q-2}+x^{q-3}+\ldots+1) = (x-1)(x^{n-1}+x^{n-2}+\ldots+1)$.
Wegen $\alpha^{j+i} \neq 1$, muss α^{j+i} eine Nullstelle von $(x^{n-1}+x^{n-2}+\ldots+1)$ sein, d.h. $1+\alpha^{j+i}+\alpha^{2(j+1)}+\ldots+\alpha^{(n-1)(j+i)} = 0$.
Damit gilt $H \cdot c^t = 0$.

Wegen $x^n-1 = g(x)h(x)$ und $x^n-1 = \prod_{i=0}^{n-1}(x-\alpha^i)$ (vgl. $(*)$) folgt aus $h(x) = \frac{x^n-1}{g(x)}$ die Behauptung über $h(x)$. □

7.6 Beispiel.
Sei $q = 5$, also $n = 4$ und $d = 3$. Dann ist $k = 2$.
$\mathbb{Z}_5^* = \{1,2,3,4\}$; $\alpha = 2$ oder $\alpha = 3$ sind möglich. Wähle $\alpha = 2$.
Erzeugerpolynom von $RS_5(3)$ nach 7.5: $g(x) = (x-2)(x-4) = x^2+4x+3$
Erzeugermatrix nach 6.8: $G = \begin{pmatrix} 3 & 4 & 1 & 0 \\ 0 & 3 & 4 & 1 \end{pmatrix}$
Kontrollmatrix nach 7.5: $H = \begin{pmatrix} 1 & 2 & 4 & 3 \\ 1 & 4 & 1 & 4 \end{pmatrix}$
Kontrollpolynom nach 7.5: $h(x) = (x-1)(x-3) = x^2+x+3$

Kontrollmatrix nach 6.8: $\widetilde{H} = \begin{pmatrix} 1 & 1 & 3 & 0 \\ 0 & 1 & 1 & 3 \end{pmatrix}$

Beachte: $|RS_5(3)| = 25$.

7.7 Bemerkung.
Reed-Muller-Codes werden häufig eingesetzt.

(a) Z.B. Deep Space Missions, etwa Voyager zu Jupiter (1979), Saturn (1981), Uranus und Neptun (Start 1977).
(Mehr unter http://www.nasa.gov/)
Hier wurde zur Übertragung ein $RS_{2^8}(223)$ eingesetzt (in Kombination mit einem Faltungscode; die Konstruktion beruht auf einem Produktcode. Auf Faltungscodes und Produktcodes werden wir später genauer eingehen).

(b) Auch für die Audio-CD- oder DVD-Codierung werden Reed-Solomon-Codes verwendet. Wir werden hier nur die Grundidee skizzieren. (Einzelheiten s. Skript WS 05/06)

Ein wesentlicher Fehlertyp bei CD's und DVD's sind Bündelfehler („bursts"), die auch Auslöschungen beinhalten können (d.h. es ist an der betreffenden Stelle kein Bit mehr zu identifizieren). Es ist nicht schwer zu zeigen: Ein $[n, k, d]$-Code kann bis zu $d-1$ Auslöschungen korrigieren, wenn keine weiteren Fehler aufgetreten sind. Die Technik zur Korrektur solcher Bündelfehler ist das sogenannte *Interleaving* (Spreizung) in der Form von *Produktcodes*.

Wie funktioniert das?
Gegeben seien zwei Codes $\mathcal{C}_1$ und $\mathcal{C}_2$ vom Typ $[n_i, k_i]$, $i = 1, 2$, mit Erzeugermatrizen in Standardform.
Die Originaldaten werden mit dem sogenannten *äußeren Code* $\mathcal{C}_2$ codiert.
Dann werden k_1 aufeinander folgende Codewörter $c_1, \ldots, c_{k_1} \in \mathcal{C}_2 \subseteq K^{n_2}$ gesammelt. Schreibe sie spaltenweise:

$$\begin{matrix} c_{11} & c_{12} & \ldots & c_{1k_1} \\ \vdots & \vdots & & \vdots \\ c_{n_2 1} & c_{n_2 2} & \ldots & c_{n_2 k_1} \end{matrix} .$$

(Die Originaldaten stehen jeweils an den Positionen $c_{1i}, \ldots, c_{k_2 i}$, $i = 1, \ldots, k_1$.)
Codiere die Zeilen mit der Erzeugermatrix (E_{k_1}, A) von $\mathcal{C}_1$ zu Codewörtern von $\mathcal{C}_1$ (dem *inneren Code*).

Originaldaten

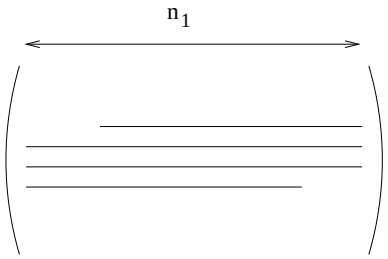
$$\begin{pmatrix} c_{11} & c_{12} & \dots & c_{1k_1} & * & \dots & * \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ c_{k_2 1} & c_{k_2 2} & \dots & c_{k_2 k_1} & * & \dots & * \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ c_{n_2 1} & c_{n_2 2} & \dots & c_{n_2 k_1} & * & \dots & * \end{pmatrix} \begin{matrix} \in \mathcal{C}_1 \\ \in \mathcal{C}_1 \\ \in \mathcal{C}_1 \end{matrix}$$

$$\underbrace{\phantom{c_{n_2 1}}}_{\in \mathcal{C}_2} \underbrace{\phantom{c_{n_2 2}}}_{\in \mathcal{C}_2} \quad \underbrace{\phantom{c_{n_2 k_1}}}_{\in \mathcal{C}_2} \underbrace{}_{\in \mathcal{C}_2} \underbrace{}_{\in \mathcal{C}_2}$$

(Wegen der Multiplikation von $\begin{pmatrix} c_{11} & \dots & c_{1k_1} \\ \vdots & & \vdots \\ c_{n_2 1} & \dots & c_{n_2 k_1} \end{pmatrix}$ mit (E_{k_1}, A) sind die letzten $n_1 - k_1$ Spalten Linearkombinationen der ersten k_1 Spalten, liegen also auch in $\mathcal{C}_2$.)

Jetzt liest man zeilenweise aus. (Produktcode, Crossinterleaving)

Angenommen ein Bündelfehler von Auslöschungen der Länge b entsteht, $b \leq (d_2 - 1)n_1$. ($d_2 = \mathrm{d}(\mathcal{C}_2)$)



$\mathcal{C}_1$ erkennt, dass Fehler in der Zeile aufgetreten sind. In jeder Spalte sind dann maximal $d_2 - 1$ Auslöschungen aufgetreten, die von $\mathcal{C}_2$ korrigiert werden können.
(Bei auftretenden Einzelfehlern in einer Zeile korrigiert $\mathcal{C}_1$ selbst, ansonsten wird die gesamte Zeile als gelöscht markiert, und dann von $\mathcal{C}_2$ wieder korrigiert.)

Nachteil: Man muss immer erst eine gesamte Matrix aufbauen, bevor man auslesen kann (entsprechend bei Decodierung). Daher werden andere Auslesetechniken verwendet mit denen die Spreizung der Fehler in den Codewörtern aus $\mathcal{C}_2$ sogar noch größer werden kann. Damit kann man Burst-Auslöschungen der Länge $(d_2 - 1) \cdot t \cdot n_1$ ($t \in \mathbb{N}$ fest, frei wählbar) korrigieren. Bei CD's wird $t = 4$ gewählt.

Bei Audio-CD-Codierung wird als Ausgangscode $RS_{2^8}(5)$ gewählt, der dann um 233 bzw. 227 Stellen verkürzt wird. Das liefert MDS-Codes $\mathcal{C}_1$ vom Typ $[32, 28, 5]$ und $\mathcal{C}_2$ vom Typ $[28, 24, 5]$ die als innere bzw. äußere Codes verwendet werden.

(Verkürzung eines Codes $\mathcal{C}$ an der Stelle i liefert den Code $\widehat{C}_i$, wobei

$$\widehat{C}_i := \{(c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_n) : (c_1, \ldots, c_{i-1}, 0, c_{i+1}, \ldots, c_n) \in \mathcal{C}\}.)$$

(CIRC: Cross-Interleaved-Reed-Solomon-Codes)

Mit zusätzlichen Techniken, Interpolationen, etc. lassen sich dann Bursts bis zu 7 mm Länge korrigieren.
Einzelheiten: Skript Codierungstheorie WS 05/06
oder Willems, Codierungstheorie

DVD's werden mit Produktcodes aus $\mathcal{C}_1$ und $\mathcal{C}_2$ codiert, wobei $\mathcal{C}_1$ ein $[172, 162, 11]$ und $\mathcal{C}_2$ ein $[208, 192, 17]$-Code über $\mathbb{F}_{2^8}$ ist. Sie entstehen durch Verkürzung aus $RS_{2^8}(11)$ und $RS_{2^8}(17)$.

7.8 Bemerkung.
Für Reed-Solomon-Codes (oder allgemeiner für sogenannte BCH-Codes, Bose, Ray-Chandhuri (1960), Hocquenghem (1959)) gibt es mehrere schnelle Decodierungsverfahren, z.B. beruhend auf dem Euklidischen Algorithmus in $K[x]$.
Wir können hierauf hier nicht eingehen und verweisen auf das oben zitierte Buch von Willems, Kapitel 9.

8 Faltungcodes

Faltungscodes (Convolutional Codes): D. Elias, 1955
Vorteil: *Soft-Decision-Decoding*

Nach der Codierung wird das diskrete Signal durch einen Modulator in ein kontinuierliches Signal umgewandelt (Aufprägung auf Welle). Der Demodulator kann nach Störung im Kanal oft nur mit einer gewissen Wahrscheinlichkeit sagen, welches Bit (oder welche Folge von Bits, die einem Inputsymbol entspricht) er empfangen hat.

Hard-Decision:	Der Demodulator muss eine Entscheidung treffen. Dann wird decodiert.
Soft-Decision:	Der Demodulator kann dem Decodierer zusätzliche Informationen geben, z.B. mit welcher Wahrscheinlichkeit 0 oder 1 empfangen wurde. Z.B. 3-Bit-Quantifizierung: Inputalphabet $\{0,1\}$, Outputalphabet $\{0,1,?\}$.

Bei Faltungscodes wird der Datenstrom (über $\mathbb{Z}_2$) in kleine Infoblöcke unterteilt, diese werden codiert, wobei die Codierung von den vorangegangenen Infoblöcken abhängt.

8.1 Definition.
Sei $K = \mathbb{Z}_2$. Ein *Faltungscode der Rate* $\frac{k}{n}$ wird durch einen Codierer mit Gedächtnis beschrieben:
Ein Datenstrom wird in *Infoblöcke* der Länge k unterteilt: $u_0, u_1, u_2, \ldots$.
Jeder Infoblock wird in ein *Codewort* der Länge n codiert: $c_0, c_1, c_2, \ldots$.
Dabei existieren $k \times n$-Matrizen $G_0, \ldots, G_m$ über $\mathbb{Z}_2$ mit

$$c_r = u_r G_0 + u_{r-1} G_1 + \ldots + u_{r-m} G_m$$

für alle $r = 0, 1, 2, \ldots$ ($u_{-1} = \ldots = u_{-m} = 0$)
(d.h. die Codierung ist linear.)
m heißt *Gedächtnislänge*, $m+1$ *Einflusslänge* des Faltungscodes ($m = 0$: normale Blockcodierung).
Üblicherweise sind k und n klein, z.B. $k = 1$ und $n = 2$ oder 3.

Besonders wichtig sind $\frac{1}{n}$-*Faltungscodes*. Hier gilt

$$G_0 = (g_{10}, \ldots, g_{n0}), \ldots, G_m = (g_{1m}, \ldots, g_{nm}).$$

Seien die Bits $u_r, \ldots, u_{r-m}$ gegeben. Dann wird das Bit u_r zu $c_r = (c_{r1}, \ldots, c_{rn})$ codiert, wobei

$$c_{ri} = g_{i0} u_r + g_{i1} u_{r-1} + \ldots + g_{im} u_{r-m} \quad (i = 1, \ldots, n).$$

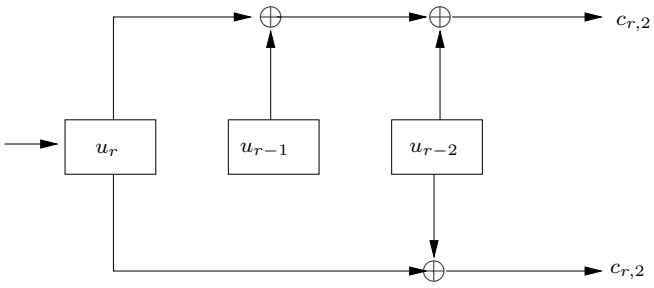
Das ist eine Faltung.
(Setzt man $G = \begin{pmatrix} G_0 \\ \vdots \\ G_m \end{pmatrix}_{(m+1)\times n}$, so ist $c_r = (u_r, \ldots, u_{r-m}) \cdot G$.)

Voraussetzung: $G_m \neq (0, \ldots, 0)$ (sonst ist die Gedächtnislänge m de facto kleiner) und $G_0 \neq (0, \ldots, 0)$ (sonst hängt die Codierung von u_r nicht von u_r ab)

8.2 Beispiele.
$\frac{1}{2}$-Faltungscode mit $m = 2$:

$$\begin{aligned} c_{r,1} &= u_r + u_{r-1} + u_{r-2} \\ c_{r,2} &= u_r + u_{r-2} \end{aligned}$$

d.h. $G = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}$
Die Codierung ist durch das folgende lineare Schieberegister (LSR) realisierbar:



Im nächsten Takt werden die Registerinhalte nach rechts verschoben, u_{r-2} fällt raus, statt u_r wird u_{r+1} in das linke Register geladen.

$$\underset{u_{-2}}{0}\ \underset{u_{-1}}{0}\ \underset{u_0}{1}\ \underset{u_1}{1}\ \underset{u_2}{0}\ \underset{u_3}{1}\ \underset{u_4}{0} \rightarrow \underbrace{11}_{c_0}\ \underbrace{01}_{c_1}\ \underbrace{01}_{c_2}\ \underbrace{00}_{c_3}\ \underbrace{10}_{c_4}$$

Bei Faltungscodes ist die Codiervorschrift einfach zu verstehen. Dagegen ist die Menge aller erzeugten Codebitfolgen in der Regel schwer zu bestimmen. (Unterschied zu Blockcodes: Algebraische Beschreibung der Codewörter zuerst; dann Codierer!)

Wir betrachten ab jetzt nur $\frac{1}{n}$-Faltungscodes.
Aus $c_0, c_1, \ldots$ lassen sich die $u_0, u_1, \ldots$ wie folgt zurückgewinnen:
Da $G_0 \neq (0, \ldots, 0)$, existiert ein i mit $g_{i0} \neq 0$, d.h. $g_{i0} = 1$. Also

$$u_r = c_{ri} - g_{i1}u_{r-1} - \ldots - g_{im}u_{r-m}.$$

Faltungscodes können beliebig lange Infobitfolgen codieren. Zur Verhinderung von Fehlerfortpflanzungen bei der Decodierung werden häufig sogenannte terminierte Faltungscodes eingesetzt:

8.3 Definition.
Bei einem *terminierten Faltungscode* mit Gedächtnislänge 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.
Das entspricht einem Blockcode, der L Infobits auf $(L + m) \cdot n$ 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.

Wir geben jetzt eine Beschreibung von Faltungscodes an, die sich für die Decodierung als vorteilhaft erweisen wird: *Trellisdiagramme* (trellis = Spalier), Fornay, 1973.

8.4 Definition.
Das *Trellisdiagramm* eines $\frac{1}{n}$-Faltungscodes mit Gedächtnislänge wird folgendermaßen erstellt:

(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 (Anordnung siehe (b)), $\xi_1 = (0, \ldots, 0)$ ist der *Nullzustand*.
Zum Zeitpunkt r befindet sich im LSR im linken Register das momentane Infobit u_r, und die übrigen Register bilden einen Zustand $(u_{r-1}, \ldots, u_{r-m}) = z_r \in \{\xi_1, \ldots, \xi_{2^m}\}$. Für $r = 0$ ist $z_0 = \xi_1$.

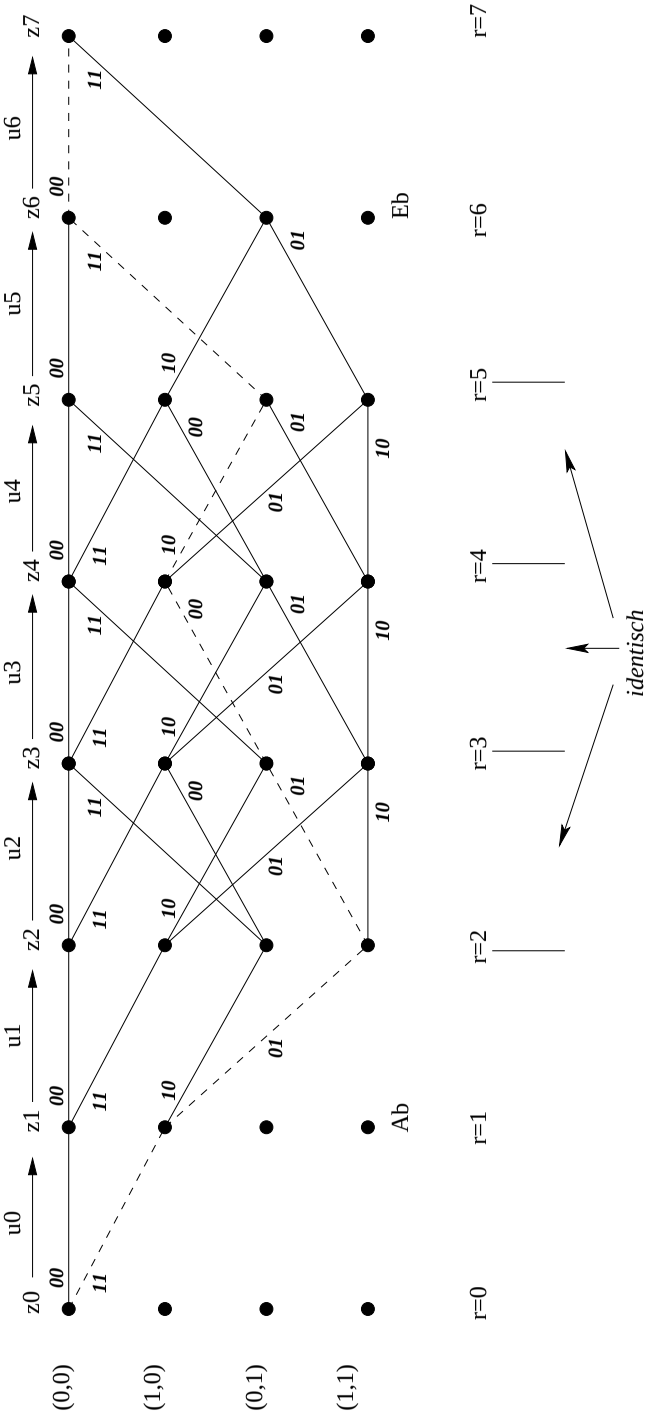
(b) Ein (vollständiges) *Trellissegment* besteht aus zwei vertikal angeordneten Punktreihen, jeweils mit 2^m Punkten, die allen möglichen Zuständen entsprechen.
Sie werden von oben nach unten in der lexikographischen Ordnung $(0 < 1)$ angeordnet, aber von rechts nach links gelesen.
($m = 2$: $(0,0) < (1,0) < (0,1) < (1,1)$)
Also steht $(0, u_{r-2}, \ldots, u_{r-m})$ immer oberhalb von $(1, u_{r-2}, \ldots, u_{r-m})$.
Von jedem Zustand $(u_{r-1}, \ldots, u_{r-m})$ der ersten vertikalen Punktreihe gehen immer 2 Kanten zu den Zuständen $(0, u_{r-1}, \ldots, u_{r-m-1})$ und

$(1, u_{r-1}, \ldots, u_{r-m-1})$.
Diese Kanten sind beschriftet mit dem Codewort, das sich ergibt, wenn die letzten m Infobits gerade $u_{r-1}, \ldots, u_{r-m}$ 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 höheren Punkt als die mit $u_r = 1$.
(Das Trellissegment beschreibt den Übergang von Zeitpunkt r zu Zeitpunkt $r + 1$).

(c) Das Trellisdiagramm ist die Hintereinanderreihung der Trellissegmente zu den Zeitpunkten $r = 0, 1, \ldots$.
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) bei jedem Zustand je zwei Kanten: In $(u_r, u_{r-1}, \ldots, u_{r-m-1})$ kommen Kanten von $(u_{r-1}, \ldots, u_{r-m-1}, 0)$ und $(u_{r-1}, \ldots, u_{r-m-1}, 1)$.
Jeder Pfad im Trellisdiagramm entspricht einer Inputfolge (obere Kante 0, untere Kante 1) und der zugehörigen Codewortfolge (ablesbar an den Kanten). Trellissegmente außerhalb der Anfangs- und Endbereiche sind identisch.

8.5 Beispiel.
Trellisdiagramm für das Beispiel 8.2. z_i = Zustand zur Zeit i.



8.6. Viterbi-Decodierung

(Viterbi, 1967) Gegeben sei ein $\frac{1}{n}$-Faltungscode.

(a) Der Viterbi-Algorithmus ist ein Maximum-Likelihood-Decodierer, das heißt zu einem empfangenen Wort $v = (v_0, \ldots, v_t)$, $v_i \in \mathbb{Z}_2^n$, wird eine Codewortfolge $c = (c_0, \ldots, c_t)$, $c_i \in \mathbb{Z}_2^n$, bestimmt, so dass $p(v|c)$ maximal wird. (vgl. 2.17 ML-Decodierung)
(Mit c ist dann die Infobitfolge eindeutig bestimmt.)

[Falls alle Quellensymbole gleich wahrscheinlich sind, minimiert ML-Decodierung die Wahrscheinlichkeit, in eine falsche Codewortfolge zu codieren.
Anderer Ansatz: Minimiere die Bit-Fehlerwahrscheinlichkeit in der Infobit-Folge z.B. mit dem BCJR-Algorithmus (Bahl, Cocke, Jelinek, Raviv); vgl. z.B. Lin, Costello, Error Control Coding]

Ist $v_i = (v_{i1}, \ldots v_{in})$, $c_i = (c_{i1}, \ldots, c_{in})$, so ist

$$p(v|c) = \prod_{i=0}^{t} p(v_i|c_i) = \prod_{i=0}^{t}\prod_{j=1}^{n} \underbrace{p(v_{ij}|c_{ij})}_{\substack{\text{W-keit, dass } v_{ij} \\ \text{empfangen wird, falls} \\ c_{ij} \text{ gesendet wird} \\ \text{(Übergangswahrscheinlichkeiten} \\ \text{des Kanals)}}}$$

(Diskreter gedächtnisloser Kanal)
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 zur Max-Likelihood-Decodierung und bedeutet die Maximierung von

$$\underbrace{\log p(v|c)}_{\substack{=:M(v|c) \\ \textit{Metrik} \text{ zur Codewortfolge } c \\ \text{(Pfad in Trellisdiagramm)} \\ \textit{Viterbi-Metrik}}} = \sum_{i=0}^{t} \underbrace{\log p(v_i|c_i)}_{\substack{=:M(v_i|c_i) \\ \textit{Kantenmetrik}, \text{ denn } c_i \\ \text{entspr. Kante in} \\ \text{Trellisdiagramm}}} = \sum_{i=0}^{t}\sum_{j=1}^{n} \underbrace{\log p(v_{ij}|c_{ij})}_{\substack{=:M(v_{ij}|c_{ij}) \\ \textit{Bitmetrik}}}$$

Kennt man die Übergangswahrscheinlichkeiten des Kanals, so kann man für jeden Weg über $t+1$ Segmente des Trellisdiagramms, also für jede Codewortfolge c, $M(v|c)$ bestimmen. Derjenige mit maximaler Metrik wird dann zur Decodierung verwendet.

Bei einem binär symmetrischen Kanal kann man zeigen:

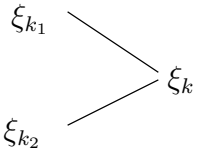
$$M(v|c) \text{ maximal } \Leftrightarrow \sum_{i=0}^{t}\sum_{j=1}^{n} v_{ij} \cdot c_{ij} =: v \cdot c \in \mathbb{N}_0 \text{ maximal}$$

(Summieren in $\mathbb{Z}$, nicht in $\mathbb{Z}_2$.)
Also wählt man dort als Metrik gerade $v \cdot c$.
ML-Decodierung bedeutet dann $\mathrm{d}(v, c)$ zu minimieren (wie bei Blockcodes).

(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, da nur bestimmte Pfade betrachtet werden müssen.

Viterbi-Algorithmus für terminierte Faltungscodes

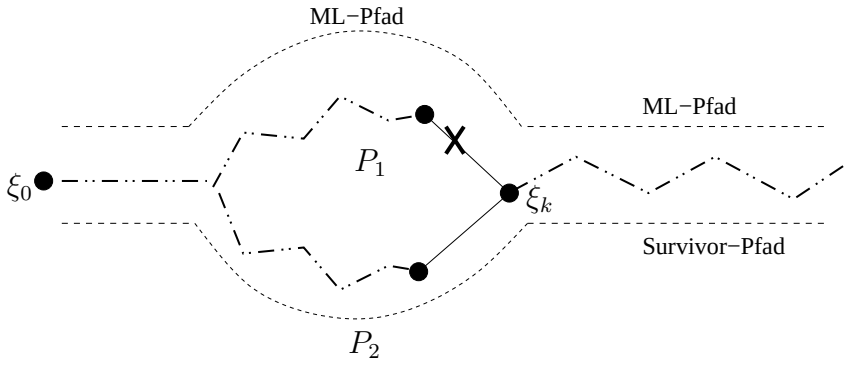
1. Starte bei $s = m$.
   Für jeden Zustand (in der vertikalen Reihe m im Trellis-Diagramm) berechne die Metrik aller eintreffenden Pfade.
2. $s := s + 1$
   Für jeden Zustand ξ_k (in der vertikalen Reihe s des Trellis-Diagramms) wird folgende Berechnung durchgefhrt:

   Bei ξ_k enden 2 Kanten von Zuständen ξ_{k1}, ξ_{k2} der vorangegangenen Reihe.
   Berechne: Metrik Survivor-Pfad bei ξ_{k_i} + Kantenmetrik $\overline{\xi_{k_i}\xi_k}$, $i = 1, 2$. Speichere die größere der beiden Summen und den entsprechenden Pfad als Survivor-Pfad (bei Mehrdeutigkeiten wähle einen der beiden aus).
3. Falls $s < L + m$ wiederhole 2., sonst stop.

8.7 Satz.
Der letzte Survivor-Pfad (entsprechend $c = (c_0, \ldots, c_{L+m-1})$) des Viterbi-Algorithmus für terminierte Faltungscodes ist der Maximum-Likelihood-Pfad (ML-Pfad), das heißt $M(v|c) \geq M(v|c')$ für alle Pfade $c' = (c'_0, \ldots, c'_{L+m-1})$.

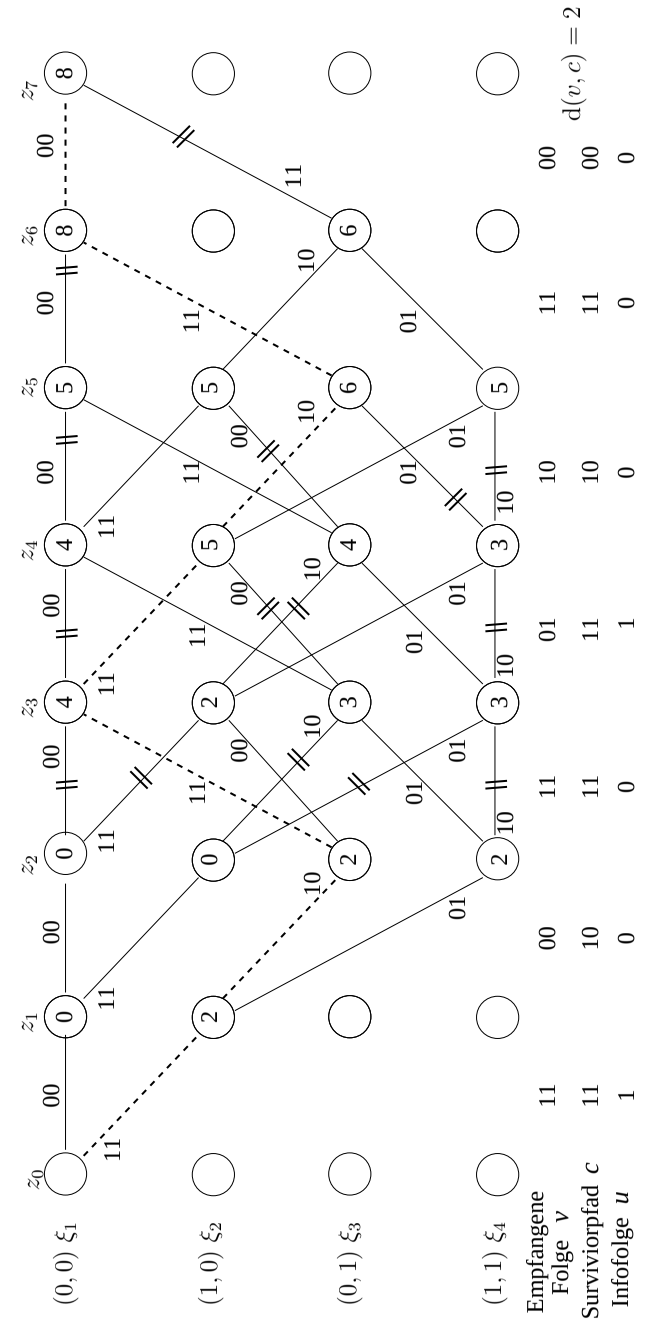
Beweis.
Angenommen der ML-Pfad wurde zum Zeitpunkt $s < L + m$ eliminiert.



Dann: Metrik von P_1 + Metrik von $\overline{P_1\xi_k}$ < Metrik von P_2 + Metrik von $\overline{P_2\xi_k}$ (nach Vorgehen Viterbi).
Also: Metrik ML-Pfad < Metrik Survivor-Pfad. Das ist ein Widerspruch.

□

8.8 Beispiel.



Viterbi–Decodierung für den Code aus Beispiel 8.2.

8.9 Bemerkung.

(a) Der Viterbi-Algorithmus verwendet Technik des dynamischen Programmierens

(b) Er kann auch für Soft-Decoding verwendet werden, d.h. die v_i sind aus einem größeren Alphabet als die c_i (siehe Skript WS 05/06).

(c) Er kann auch für nicht-terminierte Faltungscodes verwendet werden (siehe Skript WS 05/06).