FEHLERKORRIGIERENDE CODES

Jürgen Koslowski

SS 1999

Inhaltsverzeichnis

0 QUELLCODIERUNG UND SHANNONS 1. SATZ

0.0 Einleitung und Beispiele

Motiviert durch die Bedürfnisse der Nachrichtenübertragung, stehen die Arbeiten von Claude Shannon (1948) am Beginn der modernen Informationstheorie. Diese konzentriert sich aber nur auf den statistisch-syntaktischen Aspekt, d.h., auf die Wahrscheinlichkeitsverteilung der informationstragenden Elemente (Zeichen) und verzichtet auf die Berücksichtigung der Semantik, d.h., der Bedeutung der Information. Daher spricht man oft von *statistischer Informationstheorie*.

Das zugrundeliegende Modell der Nachrichtenübertragung ist das des störungsanfälligen Kanals:

$$\begin{tikzcd}
\boxed{\text{Quelle}} \arrow[r] & \boxed{\text{Codierer}} \arrow[r] & \boxed{\text{Kanal}} \arrow[r] & \boxed{\text{Decodierer}} \arrow[r] & \boxed{\text{Senke}} \\
 & & \text{Störung} \arrow[u, dotted] & &
\end{tikzcd}$$

Die Quelle erzeugt "Nachrichten", also zufällige Folgen von Zeichen, die mittels des Kanals zur Senke zu übertragen sind. Die "Nachrichten" bestehen aus "Wörtern" über einen "Alphabet" und sind ggf. aufzubereiten, damit sie "durch den Kanal passen". Diese Aufbereitung kann man generell als Codierung bezeichnen. Es lassen sich dabei zwei Stufen unterscheiden:

- Überführung der Nachrichten in ein dem Kanal angemessenes Alphabet, möglichst unter Eliminierung von Redundanz, dann sprechen wir von "Quellcodierung";
- Aufbereitung der dem Kanal angemessenen Nachrichten, um sie gegen Störungen unempfindlich zu machen, ggf. auch vor unbefugtem Zugriff zu schützen, dann sprechen wir von "Kanalcodierung".

Nach der Übertragung sind diese Schritte wieder rückgängig zu machen, durch "Kanaldecodierung" und "Quelldecodierung". Speziell bei der Kanaldecodierung gilt es, eventuell aufgetretene Störungen herauszufiltern.

Es ergeben sich nun mindestens drei Zielrichtungen für eine mathematische Theorie:

- Datenkompression (bei der Quellcodierung)
- Fehlererkennung und -korrektur (bei der Kanaldecodierung)
- Kryptologie (bei beiden Codierungsschritten)

Die ersten beiden Kapitel der Vorlesung werden sich allgemeinen Grundlagen der Quellcodierung und der Kanalcodierung widmen, im Hinblick auf die beiden wichtigen Sätze von Shannon. In den Kapiteln 3 bis 5 geht es dann auf algebraischer Grundlage um Methoden der Kanalcodierung mit guten Fehlererkennungs- und -korrektureigenschaften.

0.0.00 Beispiel. Zur Quellcodierung: In den Nachrichten aus dem Ausgangsalphabet $\{0,1,2,3\}$ trete das Symbol 0 viel häufiger auf als die anderen Symbole. Eine binäre Codierung sollte dann 0 ein kürzeres Codewort zuordnen als den anderen Symbolen, z.B.,

x	$C(x)$
0	0
1	01
2	011
3	111

Wir decodieren eine Nachricht von rechts nach links, indem wir vom Ende her Einsen zählen. Nach drei Einsen oder wenn die erste Null auftritt werden die betrachteten Zeichen entfernt. Haben wir i Einsen gezählt, so ist das letzte Symbol i. Ein praktischer Nachteil dieser Codierung liegt darin, daß die Decodierung erst nach Empfang der gesamten Nachricht möglich ist. Dies kann man vermeiden, indem man die Nullen der Codewörter $C(1)$ und $C(2)$ an das hintere Ende verschiebt, d.h.,

x	$C'(x)$
0	0
1	10
2	110
3	111

Die Decodierung kann nun analog von vorne erfolgen.

0.0.01 Beispiele. Zur Kanalcodierung: Wir betrachten einen binären Kanal, d.h., die Übertragung von Nullen und Einsen.

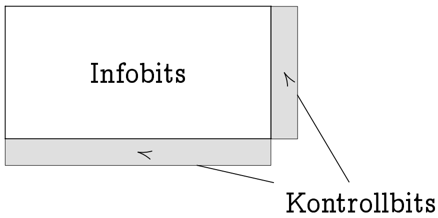
(0) Wir können alle zu übertragenden Zeichen verdoppeln, oder allgemeiner ver-n-fachen; die Decodierung erfolgt dann nach dem Majoritätsprinzip. Daher sollte n ungerade sein. Für $n=3$ erhalten wir beispielsweise

$$011010 \to 000111111000111000$$

Man spricht von einer $(3,1)$-Codierung, da von drei übertragenen Zeichen eines Information trägt, die anderen dienen der Sicherheit bzw. der Kontrolle. Pro 3er Block kann ein Fehler korrigiert werden. Solche Wiederholungscodierungen sind eher ineffizient.

(1) Paritätskontrolle: je n zu übertragenden Zeichen wird ein weiteres hinzugefügt, so daß eine gerade Anzahl von Einsen auftritt. Im obigen Beispiel ist an 011010 noch eine 1 anzuhängen. Dies ist eine $(7,6)$-Codierung, d.h., pro Block von sieben Zeichen treten sechs Informationszeichen auf. In jedem solchen Block wird eine ungerade Anzahl von Fehlern erkennt, Dreifach-, Fünffach- oder Siebenfachfehler können aber nicht von Einzelfehlern unterschieden werden. Wie kann man diese Codierung algebraisch beschreiben? Als Codewörter am Kanaleingang treten alle 7-Tupel $\langle x_0, x_1, \ldots, x_6 \rangle \in \{0,1\}^7$ auf, die $x_0 + x_1 + \cdots + x_6 = 0$ erfüllen, wobei diese Addition im Körper $\mathbb{Z}_2 = \{0,1\}$ erfolgt, in dem gilt $1+1=0=0+0$ und $1+0=1=0+1$. Eine solche Codierung ist nur für Kanäle mit geringer Störanfälligkeit geeignet.

(2) 2-dimensionale Paritätskontrolle: Denken wir uns die zu übertragenden Zeichen in einem Rechteckschema angeordnet, so können wir für jede Zeile und für jede Spalte ein (Paritäts-)Kontrollzeichen anfügen. Konkret sind etwa für 7×3 Informationszeichen $7+3$ Kontrollzeichen nötig: wir erhalten eine $(31,21)$-Codierung.



Einzelfehler können in diesem Fall nicht nur erkannt sondern sogar korrigiert werden: tritt ein Fehler bei den Informationszeichen auf, so zeigen eine Zeile und eine Spalte einen Paritätsfehler und lokalisieren dadurch das falsch empfangene Zeichen. Ein falsches Paritätszeichen identifiziert sich in Ermangelung einer weiteren Fehlermeldung (Zeile oder Spalte) selbst, und kann somit korrigiert werden. Treten zwei Fehler bei der Übertragung auf, so kann man das immer noch feststellen. Allerdings lassen sich die entstehenden Fehlermuster nicht in allen Fällen von denen gewisser Einzelfehler unterscheiden. Die Annahme, Einzelfehler seien wahrscheinlicher als Doppelfehler, führt dann zu einer falschen Decodierung. Damit lassen sich die Möglichkeit Einzelfehler zu korrigieren und die Möglichkeit bis zu zwei Fehler zu erkennen nicht gleichzeitig nutzen. Wir werden auf dieses Phänomen in Abschnitt 1.1 zurückkommen.

(3) Wie kann man die in (2) geschilderte Situation verbessern? Eine offensichtliche Möglichkeit besteht darin, ein zusätzliches Kontrollzeichen einzuführen, "in der unteren rechten Ecke" sozusagen, das die Gesamtparität überprüft. Damit verschlechtert sich zwar die "Coderate" auf $21 : 32$, aber Doppelfehler können nun zuverlässig an ihren Mustern erkannt werden. Wir können sogar Dreifachfehler erkennen, aber nicht immer eindeutig von Einzelfehlern unterscheiden (HA).

Des Weiteren stellt sich die Frage nach der praktischen Handhabung. Natürlich können wir auch in einem Rechteckschema angeordnete Zeichen nur sequentiell übertragen, d.h., es sind 32-Tupel zu senden und zu empfangen. Wie testen wir diese auf eventuell aufgetretene Fehler? Eine einfache Möglichkeit bietet die sogenannte *Kontrollmatrix* H, die in unserem speziellen Fall für elf verschiedenen Kontrollen je eine Spalte der Länge 32 besitzt. Man beachte: das Gesamtparitätsbit kann als Summe der Spaltenparitätabits oder als Summe der Zeilenparitätsbits berechnet werden; seine Überprüfung bei evtl. fehlerhaft übertragenen Daten erfordert aber die Summe über sämtliche Bits.

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

Ein Zeilenvektor $x \in \{0,1\}^{32}$ ist genau dann ein Codewort, wenn gilt $x \cdot H = 0$. Ein vom Nullvektor verschiedenes Ergebnis signalisiert mindestens einen Fehler. Bei einem Einzelfehler enthält $x \cdot H$ genau zwei Einsen, aus deren Position wir das fehlerhafte Zeichen bestimmen können. Ein praktischer Nachteil ist, daß die Kontrollzeichen nicht als zusammenhängender Block am Anfang oder Ende der 32 Gesamtzeichen auftreten. Dies wird uns später zum Begriff der "systematischen Codierung" führen.

(4) Mit Blöcken der Länge 31 lassen sich aber mehr als 21 Informationszeichen übertragen, bei gleichen Fehlerkorrekturmöglichkeiten. Dafür eignet sich z.B. eine sogenannte "Hamming-Codierung". Hierzu zäumen wir das Pferd von hinten auf: wir beginnen mit einer (31×5)-Kontrollmatrix H, deren Zeilen gerade die aufsteigenden Binärdarstellungen der Zahlen von 1 bis 31 sind. Codewörter sind diejenigen Vektoren $x \in \{0,1\}^{31}$, die $x \cdot H = 0$ erfüllen. Tritt ein Einzelfehler in Position n auf, so liefert $x \cdot H$ genau die Binärdarstellung von n und wir können diese Position korrigieren. Doppelfehler werden zwar erkannt, können aber nicht anhand ihrer Muster von Einzelfehlern unterschieden werden.

Wie codieren wir die $31 - 5 = 26$ Informationszeichen zu 31 Gesamtzeichen? Die durch $x \cdot H = 0$ beschriebenen fünf Gleichungen lassen sich zwar nach fünf Variablen auflösen, so daß diese unabhängig voneinander sind, aber leider ist dies nicht für die ersten fünf Variablen $x_0, \ldots, x_4$ oder die letzten fünf $x_{26}, \ldots, x_{30}$ möglich (HA). Man erhält zum Beispiel

$$\begin{aligned}
x_{30} &= x_0 + x_1 + x_3 + x_6 + x_{10} + x_{13} + x_{15} + x_{18} + x_{20} + x_{21} + x_{24} + x_{25} + x_{27} \\
x_{29} &= x_0 + x_1 + x_2 + x_5 + x_9 + x_{14} + x_{15} + x_{17} + x_{20} + x_{22} + x_{24} + x_{26} + x_{27} \\
x_{28} &= x_1 + x_2 + x_3 + x_4 + x_8 + x_{12} + x_{13} + x_{14} + x_{15} + x_{16} + x_{21} + x_{25} + x_{26} + x_{27} \\
x_{23} &= x_1 + x_4 + x_5 + x_6 + x_7 + x_{11} + x_{13} + x_{14} + x_{15} + x_{16} + x_{17} + x_{18} + x_{22} + x_{24} \\
&\qquad + x_{25} + x_{26} \\
x_{19} &= x_2 + x_3 + x_8 + x_9 + x_{10} + x_{11} + x_{13} + x_{14} + x_{15} + x_{16} + x_{17} + x_{18} + x_{20} + x_{21}
\end{aligned}$$

Wir müssen uns also wieder mit verstreut auftretenden Kontrollzeichen zufrieden geben.

(5) Anfügen eines Paritätskontrollzeichens und die Hamming-Codierung aus (4) liefert eine $(32, 26)$-Codierung, die gleichzeitig Einzelfehler korrigiert und Doppelfehler erkennt (HA).

(6) Verbesserungen der Codierung aus (3) sind noch in anderer Hinsicht möglich: es existiert eine $(31, 21)$-Codierung, die Doppelfehler korrigiert. Dabei handelt es sich um eine sogenannte "BCH-Codierung", die wir in Kapitel 5 behandeln wollen.

0.1 Codierung in Kommunikationssystemen

Wir legen das folgende Modell der diskreten Nachrichtenübertragung zugrunde

$$\begin{tikzcd}
\boxed{\text{Quelle}} \arrow[r] & \boxed{\text{Quellcodierer}} \arrow[r] & \boxed{\text{Kanalcodierer}} \arrow[d] & \\
& & \boxed{\text{Kanal}} \arrow[d] & \text{Störung} \arrow[l, dotted] \\
\boxed{\text{Senke}} & \boxed{\text{Quelldecodierer}} \arrow[l] & \boxed{\text{Kanaldecodierer}} \arrow[l] &
\end{tikzcd}$$

0.1.00 Beispiele. (0) Text, ASCII (American Standard Code for Information Interchange): Das Quellalphabet besteht aus 128 Buchstaben, Ziffern und diversen Sonderzeichen. Codewörter sind die 7-stelligen Binärzahlen von 0 bis 128, gefolgt von einem Paritätsbit.

(1) Buchwunsch, ISBN (International Standard Book Number): Das Quellalphabet besteht aus Author, Titel, Verlag, etc. Codewörter gehören zu $\{0, 1, \ldots, 9, X\}^{10}$, wobei das erste Symbol das Erscheinungsland (z.B. 0 = USA), die nächsten 2 oder 3 Symbole den Verlag (z.B. 13 = Prentice Hall, 471 = Wiley-Interscience) und die folgenden 6 oder 5 Symbole das Buch codieren. Hier treten nur die Ziffern von 0 bis 9 auf. Schießlich soll für ein Codewort $a_0a_1 \ldots a_9$ die folgende Summe durch 11 teilbar sein

$$\sum_{i \in 10} (i + 1)a_{9-i} = 10a_0 + 9a_1 + 8a_2 + \cdots + 2a_8 + a_9$$

Die Darstellung einer “Nachricht” erfolgt mittels “Wörtern” über einem “Alphabet”. Wir wollen diese Begriffe mathematisch präzisieren.

0.1.01 Definition. Unter einem *Alphabet* verstehen wir einfach eine Menge (von Zeichen). Ist A ein Alphabet, so setzen wir

$$A^* := \bigcup \{ A^n : n \in \mathbb{N} \} ,$$

d.h., A^* besteht aus allen endlichen Tupeln von Elementen aus A, insbesondere auch dem *leeren Wort*, dem einzigen Element von A^0 , das wir mit ε bezeichnen. Wir lassen normalerweise die Klammern um die Tupel weg und nennen die Elemente von A^* *Wörter*.

Die Konkatenation von Wörtern und das leere Wort versehen die Menge A^* mit einer algebraischen Struktur, der eines “Monoids”. Da wir in Kapitel 3 weitere algebraische Strukturen detailliert betrachten wollen, nutzen wir die Gelegenheit zu einem kleinen Ausflug in die wunderbare Welt der Monoide:

0.1.02 Definition. Ein *Monoid* $\mathcal{M} = \langle M, m, e \rangle$ besteht aus einer Menge M , einer binären Operation (inneren Verknüpfung) $M \times M \xrightarrow{m} M$ und einer Abbildung $1 \xrightarrow{e} M$ (auch als Element $e \in M$ aufzufassen), die folgenden Bedingungen genügen:

- m ist *assoziativ*, d.h. $m(m(a,b),c) = m(a,m(b,c))$ für alle $a,b,c \in M$, oder eleganter

$$\begin{tikzcd} M \times M \times M \arrow[r, "M\times m"] \arrow[d, "m\times M"'] & M \times M \arrow[d, "m"] \\ M \times M \arrow[r, "m"'] & M \end{tikzcd}$$

- e ist *neutral* bzgl. m, d.h. $m(a,e) = a = m(e,a)$ für alle $a \in M$, oder eleganter

$$\begin{tikzcd} M \times 1 \arrow[r, "M\times e"] \arrow[dr, "\cong"'] & M \times M \arrow[d, "m"] & 1 \times M \arrow[l, "e\times M"'] \arrow[dl, "\cong"] \\ & M & \end{tikzcd}$$

Ein *Monoidhomomorphismus* von $\mathcal{M}$ zu einem weiteren Monoid $\mathcal{M}' = \langle M', m', e' \rangle$ ist eine Abbildung $M \xrightarrow{f} M'$, die mit der Monoidstruktur verträglich ist, d.h., es gilt $f(e) = e'$ und $m'(f(a), f(b) = f(m(a,b))$ für alle $a,b \in M$, oder eleganter

$$\begin{tikzcd} & 1 \arrow[dl, "e"'] \arrow[dr, "e'"] & \\ M \arrow[rr, "f"'] & & M' \end{tikzcd} \quad \text{und} \quad \begin{tikzcd} M \times M \arrow[r, "m"] \arrow[d, "f\times f"'] & M \arrow[d, "f"] \\ M' \times M' \arrow[r, "m'"'] & M' \end{tikzcd}$$

0.1.03 Proposition. *Das neutrale Element eines Monoids ist eindeutig bestimmt.* □

0.1.04 Bemerkung. Eine Abbildung zwischen Monoiden, die mit den binären Operationen verträglich ist, muß nicht notwendig das neutrale Element erhalten (vergl. HA)

Monoide der Form A^*, mit der Konkatenation von Worten als binärer Operation und dem leeren Wort als neutralem Element, spielen unter allen Monoiden eine besondere Rolle, daher werden sie auch *freie Monoide* genannt.

0.1.05 Satz. *Ist $\langle M, m, e \rangle$ ein Monoid, so läßt sich jede Abbildung $A \xrightarrow{f} M$ von einer Menge A nach M eindeutig zu einem Monoidhomomorphismus $A^* \xrightarrow{\bar{f}} M$ fortsetzen.* □

Außer der algebraischen trägt A^* auch noch eine ordnungstheoretische Struktur.

0.1.06 Definition. Sind u, w Wörter aus A^*, so heißt u ein *Präfix* von w, wenn ein $v \in A^*$ existiert mit $uv = w$. Wir schreiben dann $u \leq w$.

0.1.07 Proposition. *Die Präfix-Relation $\leq$ auf A^* ist eine Ordnungsrelation, d.h., sie ist reflexiv, antisymmetrisch und transitiv.* □

Nach diesem ersten Ausflug in allgemeinere mathematische Bereiche wollen wir uns wieder der Codierungstheorie zuwenden.

0.1.08 Definition. Eine *Codierung* von einem Alphabet A in ein Alphabet B ist eine injektive Abbildung $A \xrightarrow{C} B^*$ (d.h., wir bilden *Zeichen* aus A auf *Wörter* über B ab). Der zugehörige *Code* ist das Bild von A unter C, d.h., eine Teilmenge von B^*. Eine *Blockcodierung* der Länge $n \in \mathbb{N}$ ist eine injektive Abbildung $A \xrightarrow{C} B^n$.

Wegen der Inklusion $B^n \subseteq B^*$ können wir Blockcodierungen als spezielle Codierungen auffassen. Nicht überall in der Literatur wird streng zwischen der Codierung und dem zugehörigen Code unterschieden.

0.1.09 Beispiele. (0) Morse-Codierung (wird in der Praxis nicht mehr benutzt)

(1) CCIT-Codierung

(2) Da der Binomialkoeffizient $\binom{5}{2}$ den Wert 10 hat, lassen sich die Dezimalziffern durch binäre Codewörter der Länge 5 mit genau zwei Einsen (oder analog mit genau zwei Nullen) codieren. Speziell ist

x	$C(x)$
1	11000
2	10100
3	01100
4	10010
5	01010
6	00110
7	10001
8	01001
9	00101
0	00011

als *2-aus-5-Codierung* bekannt. Man decodiert eine binären 5er Block, indem man das "Skalarprodukt" mit 01247 bildet; leider ist die Ziffer 0 eine Ausnahme.

Die Morse-Codierung hat die spezielle Eigenschaft, daß das Blank-Zeichen als Endsymbol aller Codewörter reserviert ist. Im allgemeinen Fall nennt man solch ein spezielles Endsymbol *Komma*, was zum Begriff der *Komma-Codierung* führt. Man kann sogar darauf verzichten, die Codewörter maximaler Länge mit dem ausgewählten Komma abzuschließen, da sie auch so erkannt werden, vergl. etwa die Codierung C' in Beispiel 0.0.00. Dann spricht man von einer *schwachen Komma-Codierung*.

0.2 Decodierung: Präfix-Codierungen

Die Injektivität einer Codierung $A \xrightarrow{C} B^*$ ist sicherlich notwendig, um eine codierte Nachricht wieder decodieren zu können, aber nicht hinreichend. Und selbst wenn die Decodierbarkeit gewährleistet ist, kann sie sich schwierig gestalten. Erfreulicherweise läßt sich dieses Problem in der Praxis immer umgehen.

0.2.00 Definition. Wir betrachten eine Codierung $A \xrightarrow{C} B^*$.

(0) C heißt *eindeutig decodierbar*, falls die Erweiterung $A^* \xrightarrow{\bar{C}} B^*$ injektiv ist.

(1) C heißt *unverzögert decodierbar*, falls für jedes $a \in A$ und jedes $w \in A^*$ aus $C(a) \leq \bar{C}(w)$ schon $a \leq w$ folgt.

(2) C hat die *Präfix-Eigenschaft*, falls für alle $a, a' \in A$ aus $C(a) \leq C(a')$ schon $a = a'$ folgt (d.h., kein Codewort ist Präfix eines anderen Codewortes).

0.2.01 Proposition. (0) *Jede unverzögert decodierbare Codierung hat die Präfix-Eigenschaft und ist eindeutig decodierbar.*

(1) *Eine Codierung $A \xrightarrow{C} B^*$ mit der Präfix-Eigenschaft ist unverzögert decodierbar, falls $|A| \geq 2$ gilt.*

Beweis: (0) Die erste Behauptung folgt aus $A \subseteq A^*$ und die zweite ist trivial.

(1) Wir betrachten $a \in A$ und $w \in A^*$ mit $C(a) \leq \bar{C}(w)$. Falls $w = \varepsilon$, so folgt aus $C(a) \leq \bar{C}(w) = \varepsilon$ sofort $C(a) = \varepsilon \leq C(a')$ für jedes $a' \in A$. Wegen der Präfix-Eigenschaft gilt dann $A = \{a\}$, im Widerspruch zur Voraussetzung.

Damit ist w nichtleer und hat ein erstes Symbol, etwa a'. Wegen $C(a) \leq \bar{C}(w)$ folgt nun entweder $C(a) \leq C(a')$ oder $C(a') \leq C(a)$, wegen der Präfix-Eigenschaft auf jeden Fall also $a = a'$ und somit $a \leq w$. □

0.2.02 Beispiele. (0) Jede Blockcodierung der Länge $n > 0$ hat die Präfix-Eigenschaft und ist unverzögert decodierbar, also auch eindeutig decodierbar.

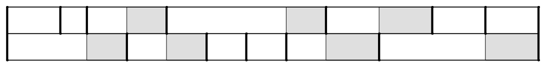
(1) Eine Codierung mit ε als Codewort ist nicht eindeutig decodierbar: aus $C(a) = \varepsilon$ folgt $\bar{C}(a^n) = \varepsilon$ für alle $n > 0$. Aus diesem Grund ist es nützlich, für die nichtleeren Wörter über einem Alphabet eine spezielle Bezeichnung einzuführen. Wir setzen $B^+ := B^* \setminus \{\varepsilon\}$.

(2) Die Codierung

x	$C(x)$
0	0
1	01
2	011
3	111

aus Beispiel 0.0.00 ist eindeutig, aber nicht unverzögert decodierbar: erst nach Empfang der gesamten Nachricht können wir vom Ende her 3-er Gruppen von Einsen wegstreichen, bevor wir entscheiden können, ob der letzte Übergang von Nullen auf Einsen von einer 1 oder einer 2 herrührt. Eine solche Codierung ist in der Praxis unbrauchbar.

Wie kann man eine Codierung $A \xrightarrow{C} B^*$ auf eindeutige Decodierbarkeit, d.h., die Erweiterung $A^* \xrightarrow{\bar{C}} B^*$ auf Injektivität prüfen? Falls $\bar{C}$ nicht injektiv ist, muß es ein kürzestes Wort in B^* geben, das auf zwei verschiedene Weisen aus Codewörtern zusammengesetzt werden kann, schematisch etwa



$K \subseteq B^*$ bezeichne die Menge aller Codewörter, also das Bild von A unter C. OBdA können wir $K \subseteq B^+$ annehmen, denn sonst wissen wir schon, daß C nicht eindeutig decodierbar ist. Außerdem sei $\bar{K}$ der von K erzeugte Untermonoid von B^*, also die Menge der aus Codewörtern zusammensetzbaren Wörter. Nun definieren wir rekursiv folgende Mengen

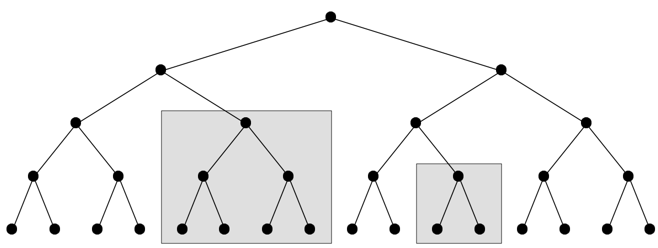
$$S_0 = \{\, w \in B^+ : \exists v \in \bar{K}.\, v \neq \varepsilon \wedge vw \in K \,\}$$
$$S_{n+1} = \{\, w \in B^+ : \exists u \in S_n \, \exists v \in \bar{K}.\, uvw \in K \,\}$$

Man überzeugt sich (leicht?) davon, daß die Folge $\langle S_n \rangle_{n \in \mathbb{N}}$ nach endlich vielen Schritten stationär wird, sofern die Menge A endlich ist.

0.2.03 Satz. *C ist genau dann eindeutig decodierbar, wenn keine der Mengen S_n, $n \in \mathbb{N}$ ein Codewort enthält.* □

Wir wenden uns nun der Frage zu, ob binäre Präfix-Codierungen mit vorgegebenen Wortlängen existieren können. Besteht das Alphabet A aus N Symbolen $a_0, a_1, \ldots, a_{N-1}$, so ordnen wir diesen positive natürliche Zahlen $l_0 \leq l_1 \leq \cdots \leq l_{N-1}$ als gewünschte Wortlängen zu. Dieses Vorgehen wird dadurch motiviert, daß häufig die Zeichen aus A mit spezifischen Wahrscheinlichkeiten auftreten und wir wahrscheinlichen Zeichen kurze und unwahrscheinlichen Zeichen längere Codewörter zuordnen wollen. Der Zusammenhang von Wahrscheinlichkeit und gewünschter Codewortlänge wird im nächsten Abschnitt hergestellt.

Betrachten wir die ersten l_{N-1} Stufen eines binären Baumes:



Eine binäre Codierung einer Menge A kann auch als Abbildung von A auf die von der Wurzel ausgehenden Pfade im Baum, oder äquivalent, auf die Knoten im Baum aufgefaßt werden. Die Präfix-Eigenschaft korrespondiert dann zu der Eigenschaft, daß die von den ausgewählten Knoten erzeugten Teilbäume disjunkt sind. Betrachten wir den Querschnitt der m-ten Stufe mit diesen disjunkten Teilbäumen, so stellen wir fest, daß für $k \leq m$ ein ausgewählter Knoten auf der k-ten Stufe 2^{m-k} Knoten "abschattet". Daher gilt für den Querschnitt auf Stufe l_{N-1}

$$\sum_{i<N} 2^{l_{N-1}-l_i} \leq 2^{l_{N-1}} \quad \text{oder äquivalent} \quad \sum_{i<N} 2^{-l_i} \leq 1 ,$$

die sogenannte *Kraft'sche Ungleichung* (KU). Sie ist eine notwendige, aber keine hinreichende Bedingung für die eindeutige Decodierbarkeit einer Codierung.

0.2.04 Beispiel. Wir betrachten vier Codierungen einer 6-elementigen Menge:

x	$C_0(x)$	$C_1(x)$	$C_2(x)$	$C_3(x)$
a_0	00	00	00	0
a_1	01	01	01	100
a_2	10	10	10	101
a_3	110	110	110	110
a_4	111	1110	1110	1110
a_5	1101	1101	1111	1111

Durch Inspektion stellen wir fest, daß C_2 und C_3 die Präfix-Eigenschaft erfüllen, C_0 und C_1 dagegen nicht (man betrachte die Codewörter für a_3 und a_5). Wie steht es mit der (KU)?

$$\begin{aligned}
C_0 &: \frac{3}{4} + \frac{2}{8} + \frac{1}{16} = \frac{17}{16} > 1 && \text{(KU) verletzt} \\
C_1 &: \frac{3}{4} + \frac{1}{8} + \frac{2}{16} = \frac{16}{16} = 1 && \text{(KU) erfüllt} \\
C_2 &: \frac{3}{4} + \frac{1}{8} + \frac{2}{16} = \frac{16}{16} = 1 && \text{(KU) erfüllt} \\
C_3 &: \frac{1}{2} + \frac{3}{8} + \frac{2}{16} = \frac{16}{16} = 1 && \text{(KU) erfüllt}
\end{aligned}$$

Man beachte: Die (KU) macht eine Aussage darüber, ob es zu einer vorgegebenen Verteilung der Codewortlängen eine Präfix-Codierung gibt, nicht darüber, ob eine vorgegebene Codierung die Präfix-Eigenschaft hat! Im Beispiel ist die (KU) für C_1 erfüllt, obwohl diese Codierung die Präfix-Eigenschaft verletzt. C_2 hat dieselbe Verteilung der Codewortlängen wie C_1 und erfüllt die Präfix-Eigenschaft.

0.2.05 Satz. (McMillan) *Jede eindeutig decodierbare binäre Codierung erfüllt die Kraft'sche Ungleichung. Insbesondere existiert zu jeder solchen Codierung eine Präfix-Codierung mit denselben Codewortlängen.*

Beweis: Die Codewortlängen einer eindeutig decodierbaren Codierung $A \xrightarrow{C} \{0,1\}^*$ seien gegeben durch $l_0, l_1, \ldots, l_{N-1}$. Wir setzen $l := \max\{l_i : i < N\}$. Um nachzuweisen, daß $s := \sum_{i<N} 2^{-l_i} \leq 1$ gilt, werden wir $\lim_{k\to\infty} s^k/k \neq \infty$ zeigen. Es gilt

$$s^2 = \Big(\sum_{i<N} 2^{-l_i}\Big)\Big(\sum_{j<N} 2^{-l_j}\Big) = \sum_{i,j<N} 2^{-(l_i+l_j)} \ ,$$

und allgemeiner

$$s^k = \sum_{\vec{i}\in N^k} 2^{-\sum_{m<k} l_{i_m}} = \sum_{j=k}^{kl} N_j 2^{-j}$$

wobei N_j die Anzahl der Summanden der Form $2^{-\sum_{m<k} l_{i_m}}$ mit $\sum_{m<k} l_{i_m} = j$ bezeichnet. Da alle Codewortlängen positiv sind, beginnt die neue Summation mit k und endet bei kl. Nun gilt es, N_j abzuschätzen. Zu jedem Summanden $2^{-\sum_{m<k} l_{i_m}}$ hat das Codewort $\bar{C}(a_{i_1} a_{i_2} \ldots a_{i_{k-1}})$ die Länge j. Wegen der Injektivität von $\bar{C}$ folgt somit $N_j \leq 2^j$ und daher

$$s^k = \sum_{j=k}^{kl} N_j 2^{-j} \leq \sum_{j=k}^{kl} 1 = k(l-1)$$

Damit ist s^k/k für $k \in I\!\!N$ nach oben durch $l-1$ beschränkt und folglich auch der Grenzwert dieser Quotienten. □

Sowohl die Kraft'sche Ungleichung wie auch der Satz von McMillan lassen sich leicht auf den Fall $|B| > 2$ übertragen. Die entsprechend verallgemeinerte Kraft'sche Ungleichung lautet:

$$\sum_{i<N} |B|^{-l_i} \leq 1$$

0.3 Informationsquellen und Entropie

Im vorigen Abschnitt haben wir für die Symbole a_i des Ausgangsalphabets Codewortlängen l_i vorgeschrieben, die nicht notwendig alle gleich sein müssen. Vielmehr sollten diese von den Auftrittswahrscheinlichkeiten der Symbole a_i abhängen – kurze Codewörter für häufige Symbole und längere Codewörter für seltene Symbole. Dies soll nun präzisiert werden.

0.3.00 Definition. (0) Unter einer *Informationsquelle* $S = \langle A, \pi \rangle$ verstehen wir ein endliches Alphabet A zusammen mit einer Wahrscheinlichkeitsverteilung $A \xrightarrow{\pi} [0,1]$ (d.h., einer Funktion π, die der Bedingung $\sum_{a \in A} \pi(a) = 1$ genügt).

(1) Ist $S = \langle A, \pi \rangle$ eine Informationsquelle und $A \xrightarrow{C} B^*$ eine Codierung, so bezeichnen wir die Summe $l(C) := \sum_{a \in A} \pi(a)|C(a)|$ als *mittlere Codewortlänge*.

0.3.01 Bemerkungen. (0) Da das Intervall $[0,1]$ bezüglich der Multiplikation ein Monoid mit neutralem Element 1 ist, läßt sich $A \xrightarrow{\pi} [0,1]$ eindeutig zu einem Monoidhomomorphismus $A^* \xrightarrow{\bar{\pi}} [0,1]$ fortsetzen. Dies ist die einfachste Möglichkeit, π zu einer Wahrscheinlichkeitsverteilung auf A^* fortzusetzen und ist als *stochastische Unabhängigkeit* zu interpretieren. In realen Anwendungen ist dies fast nie erfüllt. Z.B. tritt der Buchstabe 'q' im Deutschen und im Englischen fast nur in Verbindung mit dem Buchstaben 'u' auf. Auch bei der Übertragung von Satelliten- oder Fernsehbildern sind die Pixel aufeinanderfolgender Bilder meist nicht voneinander unabhängig. Leider ist die mathematische Handhabung solcher Komplikationen hier für uns zu aufwendig.

(1) Zur Notation: häufig denken wir uns die Elemente eines endlichen Alphabets A durchnummeriert, also $A = \{a_i : i \in N\}$ mit $a_i \neq a_j$ für $i \neq j$. Abkürzend setzen wir $\pi_i := \pi(a_i)$ und $l_i := |C(a_i)|$, $0 \leq i < N$. Die beiden Bedingungen in Definition 0.3.00 lauten nun

$$\sum_{i<N} \pi_i = 1 \qquad \text{sowie} \qquad l(C) = \sum_{i<N} \pi_i l_i$$

In der Praxis stellt sich nun die Aufgabe, eindeutig decodierbare Codierungen mit möglichst kleiner mittlerer Codewortlänge zu finden. Nach dem Satz von McMillan können wir uns dabei auf Präfix-Codierungen beschränken.

Allerdings eignet sich die mittlere Codewortlänge $l(C)$ nur zum Vergleich von Codierungen mit Codealphabeten gleicher Größe. Stehen mehr Codierungssymbole zur Verfügung, so sollten sich auch kürzere mittlere Codewortlängen realisieren lassen. Um die Vergleichbarkeit beliebiger Codierungen zu gewährleisten, wird es nötig sein, die mittlere Codewortlänge in geeigneter Weise zu normieren.

Zu diesem Zweck wollen wir nun, unabhängig von Codierungen, ein Maß für den "mittleren Informationsgehalt pro Ereignis" einer Informationsquelle einführen, wobei ein Ereignis aus dem Aussenden eines Zeichens besteht. Zur Erinnerung: wir beschränken uns hier auf den statistisch-syntaktischen Aspekt von "Nachrichten".

Wie groß ist also unser “Informationsgewinn” oder unsere “Überraschung” beim Empfang eines Zeichens $a \in A$? Offenbar ist unsere Überraschung um so größer, je kleiner der Wert $\pi(a)$ ist – umgekehrt sind wir von Zeichen hoher Wahrscheinlichkeit nicht überrascht. Slogan: “Information ist beseitigte Unbestimmtheit.”

Für eine Informationsquelle $S = \langle A, \pi \rangle$ sollte ein solches Maß allein von der Wahrscheinlichkeitsverteilung π abhängen, und nicht von der Natur der Zeichen in A. Wir setzen wieder $|A| = N$. Ist $W(A)$ die Menge aller Wahrscheinlichkeitsverteilungen auf A, d.h.,

$$W(A) = \{ \pi \in [0,1]^A : \sum_{a \in A} \pi(a) = 1 \}$$

so suchen wir eine Funktion $W(A) \xrightarrow{h_N} I\!R$. Welche Eigenschaften sollte diese Funktion sinnvollerweise haben? Hier sind drei solche Eigenschaften:

(0) Stetigkeit: Grob gesprochen “geringe Änderungen im Argument π führen zu geringen Änderungen im Funktionswert $h_N(\pi)$”;

(1) Symmetrie: Ist $A \xrightarrow{\alpha} A$ eine Permutation der Quellzeichen, so gilt $h_N(\pi \circ \alpha) = h_N(\pi)$ für alle $p \in W(A)$;

(2) Kohärenz: Gilt $N > 2$, so gibt es wegen $\sum_{i<N} \pi_i = 1$ zwei Indizes $i \neq j$ mit $\pi_i + \pi_j \neq 0$. Wegen (1) können wir $i = 0$ und $j = N-1$ annehmen. Die *reduzierte Quelle* $S' = \langle A', \pi' \rangle$ entsteht durch Zusammenfassen des ersten und des letzten Symbols, d.h., $A' = A - \{a_{N-1}\}$ und $\pi'_0 = \pi_0 + \pi_{N-1}$ sowie $\pi'_k = \pi_k$ für $k \geq 1$. Nun fordern wir
$$h_N(\pi) = h_{N-1}(\pi') + (\pi_0 + \pi_{N-1}) h_2\Big(\Big\langle \frac{\pi_0}{\pi_0 + \pi_{N-1}}, \frac{\pi_{N-1}}{\pi_0 + \pi_{N-1}} \Big\rangle\Big)$$

Die Forderung nach Kohärenz bedarf einer Begründung: wir können eine Nachricht zweimal lesen – “mit verschiedenen Brillen”: beim ersten Mal werden die Zeichen a_0 und a_{N-1} nicht unterschieden, während beim zweiten Mal die Zeichen a_i mit $2 \leq i \leq N-2$ ignoriert werden. Da wir auf diese Weise die ursprüngliche Nachricht rekonstruieren können, sollte das auch für den mittlere Informationsgehalt pro Zeichen der Gesamtquelle S gelten. Achtung: beim zweiten Lesen ist die Nachricht um den Faktor $\pi_0 + \pi_{N-1}$ kürzer geworden. Mittels Kohärenz können wir nun alle Funktionen h_N für $N \geq 2$ auf die Funktion h_2 zurückführen.

0.3.02 Satz. *Alle Funktionen, die obige drei Bedingungen erfüllen, unterscheiden sich nur um einen konstanten Faktor. Insbesondere erfüllt die stetige Fortsetzung h_N der durch*

$$\sum_{i<N} \pi_i \ln \frac{1}{\pi_i} = -\sum_{i<N} \pi_i \ln \pi_i$$

auf der Menge $\{ \pi \in]0,1]^A : \sum_{a \in A} \pi(a) = 1 \}$ der positiven Wahrscheinlichkeitsverteilungen definierten Funktion alle drei Bedingungen.

Beweis: Für uns leider zu technisch. $\square$

0.3.03 Definition. Die Funktion h_N heißt die *natürliche Entropiefunktion einer* N*-elementigen Quelle*. Für $b \geq 2$ nennen wir $H_N^{(b)} := h_N / \ln b$ die b*-näre Entropiefunktion*. Im Fall $b = 2$ sprechen wir einfach von der *Entropiefunktion* $H_N := H_N^{(2)}$. Statt $h_N(\pi)$ werden wir meist $h(S)$ schreiben, für $H_N^{(b)}$ und H_N verfahren wir entsprechend.

Daß die Funktionen, die alle drei unserer Bedingungen erfüllen, bis auf einen konstanten Faktor eindeutig bestimmt sind, kann als freie Wahlmöglichkeit einer Einheit für unseren gesuchten "mittleren Informationsgehalt pro Zeichen" interpretiert werden. Der Faktor für $H_N^{(b)}$ ist gerade so gewählt, daß die natürlichen Logarithmen in h_N durch Logarithmen zur Basis b ersetzt werden können. Die dadurch bestimmte Einheit ist gerade "1 b-näres bit pro Ereignis", wie man an der Quelle "fairer b-närer Münzwurf" leicht sehen kann: $H_b^{(b)}(\langle 1/b, \ldots, 1/b\rangle) = 1$.

Unerfreulich ist, daß die explizite Formel in Satz 0.3.02 nur für positive Wahrscheinlichkeitsverteilungen gilt. Wir erinnern uns aber an die Regel von l'Hospital

$$\lim_{x\to 0} x \ln x = \lim_{x\to 0} \frac{\ln x}{x^{-1}} = \lim_{x\to 0} \frac{x^{-1}}{-x^{-2}} = -\lim_{x\to 0} x = 0$$

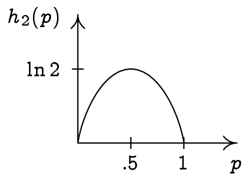
woraus wir für beliebiges $\pi \in W(A)$ schließen

$$h_N(\pi) = \sum_{i<N, \pi_i \neq 0} \pi_i \ln \frac{1}{\pi_i} = -\sum_{i<N \pi_i \neq 0} \pi_i \ln \pi_i$$

Etwas unpräzise findet man in der Literatur manchmal die "Definition" $0 \ln(1/0) = 0 = 0 \ln 0$, nun ja... Mit der Stetigkeit der Funktion h_N läßt sich das Verschwinden der Summanden mit $\pi_i = 0$ jedoch vollständig rechtfertigen.

0.3.04 Beispiele. (0) Für $A = \{a\}$ gibt es nur eine Wahrscheinlichkeitsverteilung: $\pi(a) = 1$. Dann gilt $h_1(\pi) = -\ln 1 = 0$, denn wir gewinnen pro Zeichen keine Information.

(1) $|A| = 2$: wegen $\pi_1 = 1 - \pi_0$ können wir die Funktion h_2 als Funktion einer Variablen $p \in [0,1]$ auffassen. Ihr Graph hat folgendes Aussehen:



(2) Ist π gleichverteilt, d.h., gilt $\pi_i = 1/N$ für alle $i < N$, dann folgt $h_N(\pi) = \ln N$.

Um zu zeigen, daß die Entropiefunktion genau für Gleichverteilungen ihr Maximum annimmt, brauchen wir ein Lemma.

0.3.05 Lemma. *Für alle positiven reellen Zahlen* x *gilt* $\ln x \leq x - 1$, *wobei Gleichheit nur für* $x = 1$ *auftritt.*

Beweis: Die Ableitung der durch $x-1-\ln x$ definierte Funktion, ist $1-1/x$. Sie verschwindet genau für $x=1$, ist für $x<1$ negativ, und für $x>1$ positiv. $\square$

0.3.06 Satz. *h_N nimmt genau für die Gleichverteilung den Maximalwert $\ln N$ an*

Beweis:

$$\begin{aligned} h_N(\pi)-\ln N &= \sum_{i<N,\pi_i\neq 0} \pi_i\Big(\ln\frac{1}{\pi_i}-\ln N\Big) = \sum_{i<N,\pi_i\neq 0} \pi_i \ln\frac{1}{N\pi_i} \\ &\leq \sum_{i<N,\pi_i\neq 0} \pi_i\Big(\frac{1}{N\pi_i}-1\Big) = \sum_{i<N,\pi_i\neq 0} \Big(\frac{1}{N}-\pi_i\Big) \leq \Big(\sum_{i<N}\frac{1}{N}-\sum_{i<N}\pi_i\Big) = 0 \end{aligned}$$

Gleichheit tritt genau dann ein wenn für alle $i<N$ gilt $N\pi_i=1$, d.h., $\pi_i=1/N$. $\square$

0.3.07 Proposition. *Die Entropie verschwindet genau dann, wenn die Quelle ein sicheres Ereignis enthält.*

Beweis: $h_N(\pi)$ ist Summe von $N-1$ Termen, die alle $h_2(\langle 1,0\rangle)$ beinhalten. Somit folgt die Behauptung, sobald wir sie für $h_2(\langle 1,0\rangle)$ zeigen können. Aber dies ist trivial. $\square$

0.3.08 Satz. *Ist $S=\langle A,\pi\rangle$ eine Informationsquelle, so beschränkt ihre natürliche Entropie die mit $\ln|B|$ gewichtete mittlere Codewortlänge jeder Präfix-Codierung $A \xrightarrow{C} B^*$ nach unten.*

Beweis: Wir betrachten eine Präfix-Codierung $A \xrightarrow{C} B^*$ und setzen $b:=|B|$.

$$\begin{aligned} h(S)-(\ln b)\cdot l(C) &= \sum_{i<N,\pi_i\neq 0} \pi_i\ln\frac{1}{\pi_i} - \sum_{i<N}\pi_i l_i \ln b = \sum_{i<N,\pi_i\neq 0} \pi_i\Big(\ln\frac{1}{\pi_i}+\ln b^{-l_i}\Big) \\ &= \sum_{i<N,\pi_i\neq 0} \pi_i \ln\frac{b^{-l_i}}{\pi_i} \leq \sum_{i<N,\pi_i\neq 0} \pi_i\Big(\frac{b^{-l_i}}{\pi_i}-1\Big) \\ &= \Big(\sum_{i<N,\pi_i\neq 0} b^{-l_i}-1\Big) \leq \Big(\sum_{i<N} b^{-l_i}-1\Big) \leq 0 \end{aligned}$$

wobei im letzten Schritt die verallgemeinerte KU anzuwenden ist. $\square$

Dieser Satz zeigt, daß die gewichteten mittleren Codewortlängen das eigentliche Kriterium zum Vergleich verschiedener Codierungen sein sollten.

0.3.09 Definition. Ist $S=\langle A,\pi\rangle$ eine Informationsquelle, so bezeichnen wir für eine Codierung $A \xrightarrow{C} B^*$ die gewichtete mittlere Codewortlänge $\ln|B|\cdot l(C)$ mit $L(C)$. Unter der *natürlichen Redundanz* verstehen wir die Differenz $R(C):=L(C)-h(S)$, während wir den Quotienten $E(C):=h(s)/L(C)=H^{(b)}(S)/l(C)$ als *Effizienz* bezeichnen.

Offenbar ist eine redundanzfreie Codierung möglich, wenn die auftretenden Wahrscheinlichkeiten π_i negative Potenzen von $|B|$ sind.

0.4 Konstruktion binärer Präfix-Codierungen

Es gibt drei Verfahren, zu einer Informationsquelle eine binäre Präfix-Codierung zu konstruieren. Davon ist das erste, nach Shannon benannte Verfahren nur noch von historischem Interesse, während das letzte Verfahren optimale sogenannte Huffman-Codierungen liefert.

Im Folgenden sei immer $S = \langle A, \pi \rangle$ eine Informationsquelle.

0.4.00 Algorithmus. Das *Shannon-Verfahren* für positive Wahrscheinlichkeitsverteilungen π arbeitet in vier Schritten:

(0) Umordnen der Informationsquelle nach fallenden Wahrscheinlichkeiten, so daß gilt

$$\pi_0 \geq \pi_1 \geq \ldots \geq \pi_{N-1}$$

(1) Bestimmen der Codewortlängen $l_i \in \mathbb{N}$, $i < N$, gemäß $2^{-l_i} \leq \pi_i < 2^{1-l_i}$, oder äquivalent $l_i \geq -\log_2 \pi_i > l_i - 1$, um die Gültigkeit der (KU) zu gewährleisten.

(2) Bestimmen der kumulativen Wahrscheinlichkeiten

$$\Pi_k = \sum_{i<k} \pi_i \quad \text{für } k < N$$

(3) Approximation von Π_k, $k < N$, durch den größtmöglichen Binärbruch mit l_k Stellen:

$$\frac{1}{2} \sum_{i<l_k} u_i 2^{-i} \leq \Pi_k$$

mit $u_i \in \{0, 1\}$; dann definiert man $C(a_k) := u_0 u_1 \ldots u_{l_k - 1}$. Wegen

$$\frac{1}{2} \sum_{i<l_k} u_i 2^{-i} + 2^{-l_k} \leq \frac{1}{2} \sum_{i<l_k} u_i 2^{-i} + \pi_k \leq \Pi_{k+1}$$

unterscheiden sich $C(a_k)$ und $C(a_{k+1})$ spätestens in der l_k-ten Stelle. Damit erhält man tatsächlich eine Präfix-Codierung.

0.4.01 Beispiel. Besteht A aus 7 Elementen mit der Wahrscheinlichkeitsverteilung

$$\pi = \langle 0.18;\ 0.10;\ 0.40;\ 0.08;\ 0.05;\ 0.05;\ 0.14 \rangle$$

so erhalten wir nach geeigneter Umordnung die folgende Codierung

i	π_i	$-\log_2 \pi_i$	l_i	Π_i	$C(a_i)$
0	0.40	1.322	2	0.00	00
1	0.18	2.474	3	0.40	011
2	0.14	2.837	3	0.58	100
3	0.10	3.322	4	0.72	1011
4	0.08	3.644	4	0.82	1101
5	0.05	4.322	5	0.90	11100
6	0.05	4.322	5	0.95	11110

mit $h(S) \cong 1.682$, $L(C) \cong 2.066$ und $R(C) \cong 0.383$. Offenbar ist diese Codierung nicht besonders effizient, da die Präfix-Eigenschaft erhalten bleibt, wenn einige der Codewörter um das letzte Symbol gekürzt werden, z.B., $C(a_1)$, $C(a_3)$, $C(a_4)$, $C(a_5)$ und $C_(a_6)$. Diese Modifikation liefert eine Codierung C' mit $L(C') \cong 1.747$ und $R('C) \cong 0.064$.

0.4.02 Algorithmus. Das *Shannon-Fano Verfahren* benötigt 3 Schritte und kann mechanischer ausgeführt werden als das Shannon-Verfahren:

(0) Umordnen der Informationsquelle nach fallenden Wahrscheinlichkeiten wie oben;

(1) Aufteilung der geordneten Wahrscheinlichkeiten in zwei zusammenhängende Gruppen möglichst gleich großer Summe; Aufbau der Codewörter durch Hinzufügen einer 0 für die Elemente der ersten Gruppe und einer 1 für Elemente der zweiten Gruppe.

(2) Wiederholung von Schritt (1) für die entstandenen Teilgruppen, bis alle entstandenen Gruppen nur noch ein Element enthalten.

0.4.03 Beispiel. Anwendung auf das obige Beispiel liefert

i	π_i				
0	0.40	0	0		
1	0.18	0	1		
2	0.14	1	0	0	
3	0.10	1	0	1	
4	0.08	1	1	0	
5	0.05	1	1	1	0
6	0.05	1	1	1	1

Dies ist genau die Codierung C' von Beispiel 0.4.01.

0.4.04 Algorithmus. Das *Huffman Verfahren* reduziert zunächst iterativ die Wahrscheinlichkeitsverteilung und konstruiert anschließend einen Codierungs-Baum.

(0) Umordnen der Informationsquelle nach fallenden Wahrscheinlichkeiten wie oben;

(1) Zusammenfassen der beiden kleinsten Werte zu einem $\boxed{\text{neuen Wert}}$.

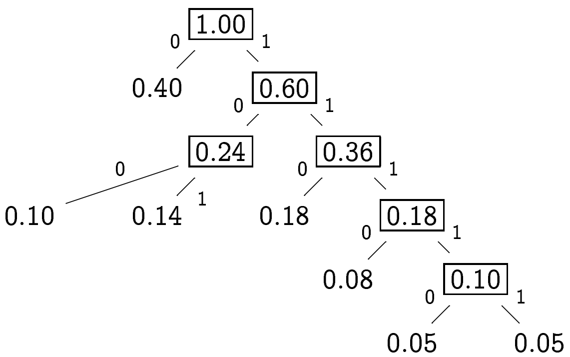
(2) Wiederholung der Schritte (0) und (1) bis der Wert 1 erreicht ist.

(3) Aufstellen eines Codebaumes mit Wurzel 1, der sich entsprechend der Zusammenfassung der Werte verzweigt; an jeder Verzweigung sind die Label 0 und 1 anzubringen

0.4.05 Beispiel. Anwendung auf das obige Beispiel liefert zunächst die Tabelle

π_i						
0.40	0.40	0.40	0.40	0.40	$\boxed{0.60}$	$\boxed{1.00}$
0.18	0.18	0.18	$\boxed{0.24}$	$\boxed{0.36}$	0.40	
0.14	0.14	$\boxed{0.18}$	0.18	0.24		
0.10	0.10	0.14	0.14			
0.08	$\boxed{0.10}$	0.10				
0.05	0.08					
0.05						

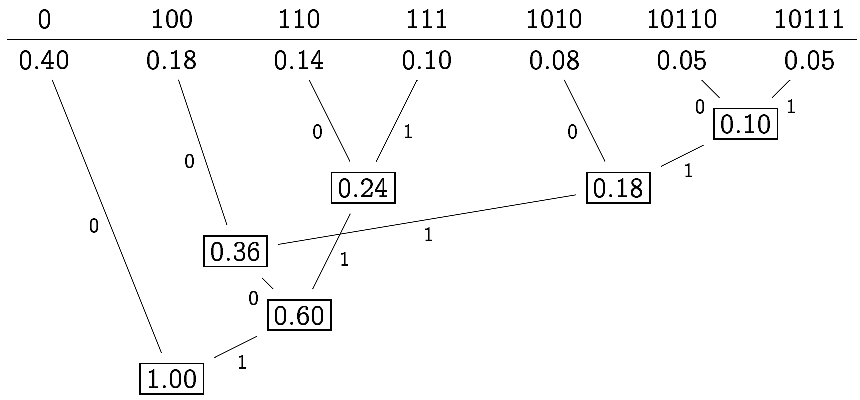
und dann den folgenden Codebaum sowie die daraus resultierende Codierung C



und

i	$C(a_i)$
0	0
1	110
2	101
3	100
4	1110
5	11110
6	11111

mit den Werten $l(C) = 2.48$, $L(C) \cong 1.719$ und $R(C) \cong 0.037$. Die Verteilung der Label im Codebaum unterliegt nur der Einschränkung, daß an jeder Verzweigung beide Label vorkommen müssen. Die Längen der Codewörter bleiben davon unbeeinflußt, es werden nur in bestimmten Positionen die Nullen und Einsen gegeneinander ausgetauscht. Das sieht man auch an folgender Abkürzung des Verfahrens, bei der allerdings nicht-planare Bäume vorkommen können:



Auffällig ist, daß die Summe der Kästchen gerade die mittlere Codewortlänge $l(C)$ ist.

Kann man noch bessere Präfix-Codierungen finden, mit kleinerer Redundanz? Der folgende Satz zeigt, daß dies nicht möglich ist; insofern ist das Huffman-Verfahren optimal.

0.4.06 Satz. *Für eine Informationsquelle $S = \langle A, \pi \rangle$ mit $|A| = N > 2$ und nicht steigenden Wahrscheinlichkeiten definieren wir die reduzierte Quelle $S' = \langle A', \pi' \rangle$ durch*

$$A' = A \setminus \{a_{N-1}\} \qquad \text{und} \qquad \pi'_i = \pi_i \quad \text{für } i < N-2\,, \text{ sowie } \quad \pi'_{N-2} = \pi_{N-2} + \pi_{N-1}$$

Ist $A' \xrightarrow{C'} \{0,1\}^*$ eine Präfix-Codierung minimaler Redundanz für S', so hat auch die folgendermaßen definierte Codierung $A \xrightarrow{C} \{0,1\}^*$ diese Eigenschaft für S:

$$C(a_i) = C'(a_i) \quad \text{für } i < N-2\,, \text{ sowie } \quad C(a_{N-2}) = C'(a_{N-2})0 \quad , \quad C(a_{N-1}) = C'(a_{N-2})1$$

Beweis: Daß es sich bei C um eine Präfix-Codierung handelt ist klar. Nachzuweisen bleibt die Minimalität der Redundanz. Dazu betrachten wir eine Präfix-Codierung $A \xrightarrow{D} \{0,1\}$ minimaler Redundanz für S. Offenbar ist dann auch $l(D)$ minimal.

(0) Falls für ein $i < N-1$ gilt $|D(a_i)| > |D(a_{i+1})|$, so liefert das Vertauschen der Codewörter $D(a_i)$ und $D(a_{i+1})$ eine Präfix-Codierung $\tilde{D}$ mit

$$\begin{aligned} l(D) - l(\tilde{D}) &= |D(a_i)|\pi_i + |D(a_{i+1})|\pi_{i+1} - |D(a_i)|\pi_{i+1} - |D(a_{i+1})|\pi_i \\ &= (|D(a_i)| - |D(a_{i+1})|)(\pi_i - \pi_{i+1}) \geq 0 \end{aligned}$$

woraus wegen der Minimalität von $l(D)$ folgt $l(\tilde{D}) = l(D)$. Also können wir oBdA annehmen, daß die Codewortlängen von D nicht fallen.

(1) Wäre das Codewort $D(a_{N-1})$ echt länger als $D(a_{N-2})$, könnten wir die überzähligen Zeichen von $D(a_{N-1})$ weglassen und so die mittlere Codewortlänge verringern, ohne die Präfix-Eigenschaft zu zerstören. Also gilt $|D(a_{N-1})| = |D(a_{N-2})|$.

(2) Falls das von $D(a_{N-2})$ nur im letzten Zeichen verschiedene Wort als Codewort von D vorkommt, vertauschen wir es mit $D(a_{N-1})$, andernfalls ersetzen wir $D(a_{N-1})$ durch dieses Wort. In beiden Fällen ändert sich nichts an der Präfix-Eigenschaft und den beiden Eigenschaften in (0) und (1). Also können wir oBdA annehmen, daß sich die letzten beiden Codewörter nur an der letzten Stelle unterscheiden.

Nun betrachten wir die durch D auf A' induzierte Codierung D', die für $i < N-2$ mit D übereinstimmt und a_{N-2} auf das um das letzte Zeichen reduzierte Wort $D(a_{N-2})$ abbildet. Wegen (0) und (2) kann $D'(a_{N-2})$ nicht Präfix eines anderen Codeworts von D' sein, also ist D' wieder eine Präfix-Codierung. Nach Konstruktion gilt nun

$$l(C) = \sum_{i<N-2} \pi_i |C'(a_i)| + (\pi_{N-2} + \pi_{N-1})(|C'(a_{N-2})| + 1) = l(C') + \pi_{N-2} + \pi_{N-1}$$

Ganz analog erhalten wir $l(D) = l(D') + \pi_{N-2} + \pi_{N-1}$. Aber nach Voraussetzung gilt $l(C') \leq l(D')$ sowie $l(D) \leq l(C)$, woraus nun $l(D) = l(C)$ folgt. Also ist C eine Präfix-Codierung mit minimaler Redundanz. □

Das Huffman-Verfahren benutzt diesen Satz wiederholt: Zur Konstruktion einer binären Präfix-Codierung minimaler Redundanz auf S benötigen wir eine solche für die reduzierte Quelle S'. Wir reduzieren solange, bis wir eine zwei-elementige Quelle haben. Dafür kennen wir eine triviale binäre Präfix-Codierung minimaler Redundanz. Dieses Ergebnis rechtfertigt die folgende Sprechweise:

0.4.07 Definition. Eine eindeutig decodierbare Präfix-Codierung minimaler Redundanz bezeichnen wir als *Huffman-Codierung*.

0.4.08 Bemerkungen. (0) Für $N = 2$ ist der Satz nicht anwendbar: die Reduktion einer zwei-elementigen Quelle hat die Form $\langle \{a_0\}, 1 \rangle$. Die Codierung $a_0 \to 1$ ist trivialerweise eine Huffman-Codierung, aber ihre durch $a_0 \to 10$ und $a_1 \to 11$ gegebene Erweiterung ist keine.

(1) Alle drei Verfahren lassen sich auch für Codealphabete mit $b > 2$ Zeichen anwenden. Speziell beim Huffman-Verfahren sind dabei jeweils die b geringsten Wahrscheinlichkeiten zusammenzufassen, bis auf den ersten Schritt, in dem b_0 Werte addiert werden: dabei gilt $2 \leq b_0 \leq b$ und $b - 1$ teilt $N - b_0$.

(2) Zwei Huffman-Codierungen derselben Quelle müssen nicht dieselbe Verteilung der Codewortlängen haben!

0.5 Der 1. Satz von Shannon

Wie wir im vorigen Abschnitt gesehen haben, minimieren Huffman-Codierungen die gewichtete mittlere Codewortlänge, oder äquivalent, die Redundanz. Anders ausgedrückt: Huffman-codierte "Nachrichten" übertragen pro Zeichen ein Maximum an "Information".

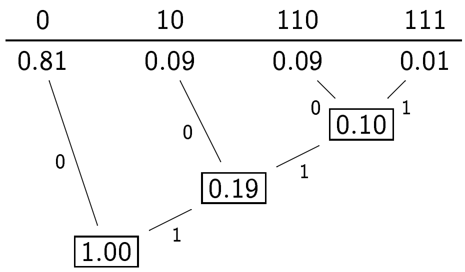
Weiterhin ist klar, daß die Redundanz auf die Abweichungen der Wahrscheinlichkeiten π_i von negativen Potenzen von $b := |B|$ zurückzuführen ist. Wir können einerseits hoffen, die Redundanz zu verringern, indem wir b vergrößern und somit gewissermaßen die "Auflösung" verbessern. Sobald b den Wert $|A| = N$ erreicht hat, können wir eine gewichtete mittlere Codewortlänge von $\ln N$ realisieren, dem Maximalwert der natürlichen Entropiefunktion h_N. In der Praxis steht uns diese Möglichkeit normalerweise nicht zur Verfügung. Um bei festem b dennoch die Redundanz zu verringern, bleibt uns nur, die Quelle künstlich zu modifizieren. Statt einzelner Zeichen kann man Gruppen von jeweils k Zeichen betrachten, deren Wahrscheinlichkeit gerade das Produkt der Einzelwahrscheinlichkeiten ist. Dies vergrößert die Anzahl der zu codierenden Elemente und verringert deren Wahrscheinlichkeiten – wir hoffen, daß sie jetzt näher an den negativen Potenzen von b liegen als vorher.

0.5.00 Definition. Ist $S = \langle A, \pi \rangle$ eine Informationsquelle und $k \in \mathbb{N}$, so verstehen wir unter der *k-ten Erweiterung* S^k die Quelle $\langle A^k, \pi^{(k)} \rangle$, wobei $\pi^{(k)}$ die Einschränkung der Erweiterung $A^* \xrightarrow{\bar{\pi}} [0,1]$ auf $A^k \subseteq A^*$ ist.

Während uns das folgende Beispiel den potentiellen Nutzen solcher Erweiterungen vor Augen führt, soll ein geringer Nachteil nicht verschwiegen werden: Die Decodierung der ursprünglichen Quelle erfolgt nicht mehr unverzögert.

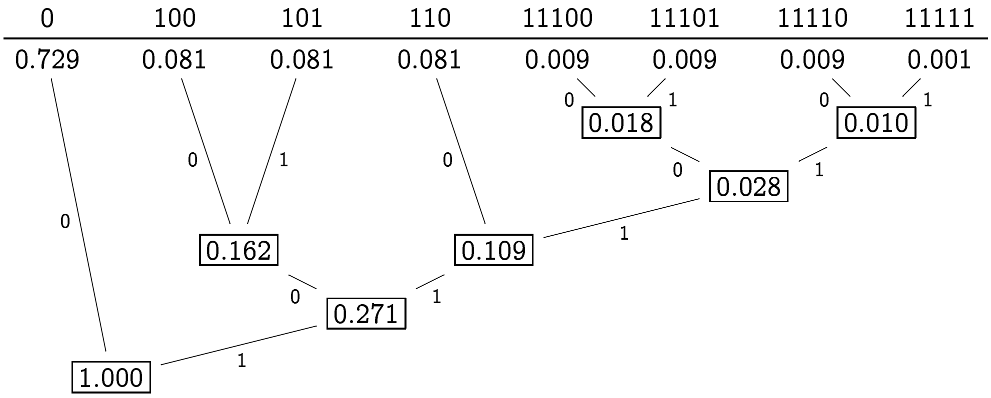
0.5.01 Beispiel. Wir betrachten die binäre Quelle $S = \langle \{a_0, a_1\}, (0.9, 0.1) \rangle$. Eine direkte binäre Huffman-Codierung C_1 erfüllt offenbar $l(C_1) = 1$, also gilt für die gewichtete mittlere Codewortlänge $L(C_1) = \ln 2$. Die natürliche Entropie dieser Quelle ist gegeben durch $h(S) \cong 0.325$, was eine Redundanz von $R(C_1) \cong 0.368$ liefert.

Die erweiterte Quelle $S^2 = \langle \{a_0a_0, a_0a_1, a_1a_0, a_1a_1\}, (0.81, 0.09, 0.09, 0.01) \rangle$ hat eine natürliche Entropie von $h(S^2) = 2h(S) \cong 0.650$ (dies ist natürlich kein Zufall, siehe Proposition 0.5.02). Der Baum



liefert nun eine Huffman-Codierung C_2 der Erweiterung mit $l(C_2) = 1.29$ und $L(C_2) \cong 0.894$, was einer natürlichen Redundanz von $R(C_2) \cong 0.244$ entspricht. Angesichts der Tatsache, daß jedes Codewort zwei Zeichen aus A überträgt, sind diese Werte noch durch den Faktor 2 zu dividieren, um einen sinnvollen Vergleich mit C_1 zu ermöglichen.

Für S^3 liefert der Baum



eine Huffman-Codierung C_3 mit $l(C_3) = 1.598$, $L(C_3) \cong 1.108$ und $R(C_3) \cong 0.132$, Werte, die noch durch den Faktor 3 zu dividieren sind.

0.5.02 Proposition. *Für jede Quelle S und jedes $k \in \mathbb{N}$ gilt $h(S^k) = k \cdot h(S)$.*

Beweis: Vollständige Induktion über k : für $k = 0$ ist die Behauptung trivial, nehmen wir also an, sie sei für ein $k \geq 1$ erfüllt. Zur Abkürzung schreiben wir $\pi_{\vec{\imath}} := \prod_{l<k} \pi_{i_l}$ für $\vec{\imath} \in N^k$.

$$\begin{aligned}
h(S^{k+1}) &= -\sum_{\vec{\imath}\in N^k}\sum_{j<N} \pi_{\vec{\imath}}\pi_j \ln(\pi_{\vec{\imath}}\pi_j) = -\sum_{\vec{\imath}\in N^k}\sum_{j<N} \pi_{\vec{\imath}}\pi_j (\ln \pi_{\vec{\imath}} + \ln \pi_j) \\
&= -\sum_{\vec{\imath}\in N^k} \pi_{\vec{\imath}} \ln \pi_{\vec{\imath}} \sum_{j<N} \pi_j - \sum_{j<N} \pi_j \ln \pi_j \sum_{\vec{\imath}\in N^k} \pi_{\vec{\imath}} \\
&= -\sum_{\vec{\imath}\in N^k} \pi_{\vec{\imath}} \ln \pi_{\vec{\imath}} - \sum_{j<N} \pi_j \ln \pi_j \\
&= h(S^k) + h(S) = k \cdot h(S) + h(S) = (k+1)h(S)
\end{aligned}$$

0.5.03 Satz. (Shannons 1. Satz) *Jede Huffman-Codierung $A^k \xrightarrow{C} B^*$ der k-ten Erweiterung einer Informationsquelle $S = \langle A, \pi \rangle$ genügt der Bedingung*

$$h(S) \leq \frac{L(C)}{k} \leq h(S) + \frac{\ln b}{k}$$

Beweis: Für eine Codierung $A \xrightarrow{C} B^*$ zeigen wir $h(S) \leq L(C) \leq h(S) + \ln b$ und wenden dieses Ergebnis auf S^k an. Die erste Ungleichung folgt nach Satz 0.4.06. Zum Nachweis der zweiten Ungleichung betrachten wir für alle $i < N$ mit $\pi_i \neq 0$ die durch

$$\log_b \frac{1}{\pi_i} < l_i \leq 1 + \log_b \frac{1}{\pi_i}$$

eindeutig bestimmten natürlichen Zahlen l_i . Wegen

$$\sum_{i<N,\pi_i\neq 0} b^{-l_i} < \sum_{i<N,\pi_i\neq 0} b^{-\log_b \frac{1}{\pi_i}} = \sum_{i<N,\pi_i\neq 0} \pi_i = 1$$

können wir nun für alle $i < N$ mit $\pi_i = 0$ weitere positive Zahlen l_i derart wählen, daß die (KU) bzgl. b erfüllt ist. Also gibt es eine Präfix-Codierung $A \xrightarrow{D} B^*$ mit $l_i = |D(a_i)|$, $i < N$. Da C eine Huffman-Codierung ist, folgt nun

$$\begin{aligned} l(C) \leq l(D) &= \sum_{i<N} \pi_i l_i = \sum_{i<N,\pi_i\neq 0} \pi_i l_i \\ &\leq \sum_{i<N,\pi_i\neq 0} \pi_i + \sum_{i<N,\pi_i\neq 0} \pi_i \log_b \frac{1}{\pi_i} = 1 + H^{(b)}(S) \end{aligned}$$

und somit $L(C) \leq \ln b + h(S)$. □

Mittels des Tricks der Erweiterung einer Quelle können wir somit die gewichtete mittlere Codewortlänge pro Originalsymbol beliebig der natürlichen Entropie der Originalquelle annähern.

1 KANALCODIERUNG UND SHANNONS 2. SATZ

1.0 Übersicht

Nachdem wir die Quellcodierung und -decodierung mittels Präfix-Codierungen mathematisch in den Griff bekommen haben, wenden wir uns nun dem mittleren Teil des Modells der Nachrichtenübertragung zu.

$$\begin{tikzcd}
\boxed{\text{Kanalcodierer}} \arrow[r] & \boxed{\text{Kanal}} \arrow[r] & \boxed{\text{Kanaldecodierer}} \\
 & \text{Störung} \arrow[u, dotted] &
\end{tikzcd}$$

In der Praxis finden als Kanalcodierung häufig sogenannte "Faltungscodierungen" Anwendung, deren mathematische Analyse allerdings schwierig ist. Hier werden wir nur Blockcodierungen der Form $B^k \xrightarrow{C} B^n$ betrachten, wobei B ein für den Kanal geeignetes Alphabet ist. Der vom Quellcodierer erzeugte Zeichenstrom wird also in Pakete der Länge k zerhackt, die als Eingabe für C dienen. Die Ausgaben werden beim Transport über den Kanal evtl. gestört, aber wir nehmen an, daß keine Zeichen verschluckt oder hinzugefügt werden. So können beim Kanaldecodierer beliebige Wörter aus B^n eintreffen. Diese sind zunächst mit Hilfe einer ggf. probabilistischen Abbildung $B^n \xrightarrow{D} C[B^k]$ in Codewörter umzuwandeln, die dann mittels der Umkehrabbildung zu C decodiert werden. Die Codewörter selbst bleiben unter D natürlich invariant.

1.0.00 Definition. Für ein Codeworte $y \in C[B^k]$ bezeichne $P_{\text{err}}(y)$ die Wahrscheinlichkeit, daß D als Ergebnis y liefert, wenn ein Codewort $x \neq y$ gesendet wurde. Die *mittlere Fehlerwarhscheinlichkeit* ist dann gegeben durch

$$P_{\text{err}}(C, D) = \frac{1}{|B|^k} \sum_{y \in C[B^k]} P_{\text{err}}(y)$$

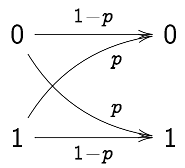
Häufig wird D nicht explizit erwähnt, da es in den meisten Fällen nur eine sinnvolle (probabilistische) Abbildung D gibt, die Nicht-Codewörter aus $B^n \setminus C[B^k]$ in Codewörter umwandelt (die sogenannte *maximum likelihood Decodierung*, vergl. Definition 1.2.05(1)).

Eine geringe mittlere Fehlerwahrscheinlichkeit $P_{\text{err}}(C)$ kann etwa durch eine Reduktion der *Coderate* $\rho(C) = k/n$ erkauft werden. Aber wie weit muß man mit dieser heruntergehen, um einen akzeptablen Wert für $P_{\text{err}}(C)$ zu erzielen? Der zweite Satz von Shannon garantiert die Existenz einer Folge $\langle C_n \rangle_{n \in \mathbb{N}}$ von Blockcodierungen, so daß die Coderaten $\rho(C_n)$ für $n \to \infty$ von unten gegen eine Kenngröße $\kappa > 0$ des Kanals, die sogenannte *Kapazität*, konvergieren, und gleichzeitig $P_{\text{err}}(C_n) \to 0$ gilt.

1.1 Der binäre symmetrische Kanal: ein Beispiel

Unser Hauptaugenmerk wird dem binären symmetrischen Kanal gelten, der das Alphabet $B = \{0, 1\}$ auf sich selbst abbildet, und dabei mit Wahrscheinlichkeit p ein Zeichen verändert:

schematisch:



oder formal: $P_{\text{err}}(0) = P_{\text{err}}(1) = p$

OBdA können wir $0 \le p \le 1/2$ annehmen, andernfalls vertauschen wir die Zeichen am Kanalausgang. Bei n empfangenen Zeichen ist die Wahrscheinlichkeit für j Fehler gegeben durch

$$\binom{n}{j} p^j (1-p)^{n-j}$$

Falls $p < 1/2$, so gilt $1 - p > 1/2$ und wir haben folgende Kette von echten Ungleichungen

$$(1-p)^n > p(1-p)^{n-1} > p^2(1-p)^{n-2} > \dots$$

Damit ist das Auftreten eines jeden k-fach Fehlers wahrscheinlicher als das Auftreten jedes $(k+1)$-fach Fehlers. Nach Empfang eines gestörten Wortes w wollen wir also nach einem Codewort suchen, das sich in möglichst wenigen Positionen von w unterscheidet (*maximum likelihood Decodierung*, kurz MLD). Für $p = 1/2$ ist das nicht sinnvoll, daher müssen wir $p \ne 1/2$ annehmen.

1.1.00 Beispiel. Für einen binären symmetrischen Kanal mit $p = 0.001$ soll eine Codierung C mit $P_{\text{err}}(C) \le 2 \cdot 10^{-6}$ gefunden werden.

Bei einer (ungeraden) Wiederholungscodierung erfolgt die Decodierung durch Majoritätsentscheidung. Im Fall von C_3 mir Coderate $1/3$ schlägt diese Decodierung fehl, wenn drei oder zwei Zeichen verfälscht wurden, d.h.,

$$P_{\text{err}}(C_3) = (0.001)^3 + 3(0.001)^2(0.999) \cong 3 \cdot 10^{-6}$$

Dies reicht noch nicht aus. Im Fall von C_5 mit Coderate $1/5$ resultieren Decodierungsfehler, wenn fünf, vier oder drei Übertragungsfehler auftreten, also gilt

$$P_{\text{err}}(C_5) = (0.001)^5 + 5(0.001)^4(0.999) + 10(0.001)^3(0.999)^2 \cong 1 \cdot 10^{-8}$$

Kann der Kanal nur doppelt soviele Zeichen versenden, wie der Quellcodierer liefert, sollten wir aus Effizienzgründen eine Coderate von $1/2$ oder größer anstreben. Wiederholungscodierungen können das nicht leisten. Die Idee ist nunmehr, die quellcodierte Nachricht in Blöcke der Länge k zu zerlegen, und pro Block $2k$ Zeichen zu versenden.

Für $k = 2$ kann man dies etwa durch Verdreifachen des zweiten Zeichens realisieren (Verdoppelungen sind nicht sinnvoll). Wir bezeichnen die entsprechende Codierung mit $C_{(4)}$:

w	$C_{(4)}(w)$
00	0000
01	0111
10	1000
11	1111

Bei der Decodierung eines 4er-Blocks nehmen wir an, das erste Zeichen sei korrekt, und bestimmen das zweite nach Majorität. Bei keinem Fehler oder einem Fehler in den letzten drei Positionen liefert dieses Verfahren das korrekte Ergebnis. Somit gilt

$$1 - P_{\text{err}}(C_{(4)}) = (1-p)^4 + 3(1-p)^3 p = (0.999)^4 + 3(0.999)^3(0.001) \cong 0.998997$$
$$\Longrightarrow P_{\text{err}}(C_{(4)}) \cong 1.003 \cdot 10^{-3}$$

Da dieses Ergebnis noch nicht ausreicht, versuchen wir es mit $k = 3$. Die folgende Codierung $C_{(6)}$ fügt jedem Binärtripel ein weiteres Tripel hinzu derart, daß sich je zwei Codewörter an mindestens drei Stellen unterscheiden. Insbesondere enthält jedes Codewort außer $C_{(6)}(0)$ drei oder vier Einsen.

w	$C_{(6)}(w)$
000	000000
100	100011
010	010101
001	001110
011	011011
101	101101
110	110110
111	111000

Jeder einzelne Fehler läßt sich daher korrigieren: es gibt genau ein Codewort, das sich vom empfangenen Wort an einer Stelle unterscheidet. Für Doppelfehler ist die Situation etwas unübersichtlich. Manche Doppelfehler sind von Einzelfehlern nicht zu unterscheiden und werden daher immer falsch decodiert, etwa wenn wir 110111 statt 100011 empfangen und 110110 decodieren. Andere Doppelfehler unterscheiden sich von mehreren Codewörtern in je zwei Stellen, etwa 101010 von 100011, 111000 und 001110. In einem Drittel dieser Fälle liefert hier die MLD das richtige Ergebnis. Insgesamt wird sicherlich weniger als ein Drittel aller Doppelfehler richtig decodiert, woraus wir schließen

$$(1-p)^6 + 6(1-p)^5 p < 1 - P_{\text{err}}(C_{(6)}) < (1-p)^6 + 6(1-p)^5 p + \frac{1}{3}\binom{6}{2}(1-p)^4 p^2$$
$$(0.999)^6 + 6(0.999)^5(0.001) < 1 - P_{\text{err}}(C_{(6)}) < (0.999)^6 + 6(0.999)^5(0.001) + 5(0.999)^4(0.001)^2$$
$$0.999985 < 1 - P_{\text{err}}(C_{(6)}) < 0.999990$$

Damit ergibt sich $9.980 \cdot 10^{-6} < P_{\text{err}}(C_{(6)}) < 1.496 \cdot 10^{-5}$, was immer noch nicht ausreicht.

Man könnte nun versucht sein anzunehmen, für $k = 4$ würde die gewünschte Schranke unterschritten, sofern wir eine Codierung konstruieren können, deren Codewörter der Länge 8 sich mindestens in jeweils vier Positionen unterscheiden. Leider ist dies ein Trugschluß, wie die Codierung $C_{(8)}$ zeigt, die sich aus $C_{(6)}$ wie folgt ergibt: Jedes Codewort soll eine durch 4 teilbare Anzahl von Einsen enthalten. Für $w \in \{0,1\}^3$ ergibt sich $C_{(8)}(w0)$ durch Anhängen von 01 bzw 00 an $C_{(6)}(w)$, während $C_{(8)}(w1)$ als "Komplement" von $C_{(8)}(w0)$ definiert wird, das sich durch Vertauschen der Nullen und Einsen ergibt:

w	$C_{(8)}(w0)$	$C_{(8)}(w1)$
000	00000000	11111111
100	10001101	01110010
010	01010101	10101010
001	00111001	11000110
011	01101100	10010011
101	10110100	01001011
110	11011000	00100111
111	11100001	00011110

Wie oben korrigiert die MLD Einzelfehler. Dreifachfehler sind von Einzelfehlern nicht zu unterscheiden und werden immer falsch decodiert. Dagegen können Doppelfehler als solche erkannt werden. Allerdings gibt es immer vier Codewörter, die sich von einem empfangenen Wort mit zwei Fehlern an zwei Stellen unterscheiden, so daß die MLD nur in einem Viertel der Fälle das richtige Ergebnis liefert. Damit erhalten wir

$$\begin{aligned} 1 - P_{\text{err}}(C_{(8)}) &= (1-p)^8 + 8(1-p)^7 p + 7(1-p)^6 p^2 \\ &= (0.999)^8 + 8(0.999)^7(0.001) + 7(0.999)^6(0.001)^2 \\ &\cong 0.999979 \\ \Longrightarrow P_{\text{err}}(C_{(6)}) &\cong 2.093 \cdot 10^{-5} \end{aligned}$$

Dieses Ergebnis ist sogar schlechter als das für $C_{(6)}$. Allerdings haben wir die Eigenschaft dieser Codierung, Doppelfehler als solche zu erkennen, überhaupt nicht ausgenutzt. Falls die Möglichkeit besteht, 8er-Blöcke mit zwei Fehlern einmal neu schicken zu lassen (um sie dann nach dem obigen Verfahren zu decodieren), läßt sich das Ergebnis erheblich verbessern:

$$\begin{aligned} 1 - \tilde{P}_{\text{err}}(C_{(8)}) &= (1-p)^8 + 8(1-p)^7 p + (0.99979)28(1-p)^6 p^2 \\ &= (0.999)^8 + 8(0.999)^7(0.001) + (0.999979)28(0.999)^6(0.001)^2 \\ &\cong 0.9999999436 \\ \Longrightarrow \tilde{P}_{\text{err}}(C_{(6)}) &\cong 1.409 \cdot 10^{-8} \end{aligned}$$

Allerdings sinkt jetzt die Coderate unter $1/2$.

1.2 Fehlererkennung und -korrektur

1.2.00 Definition. Unter einer *Metrik* oder *Abstandsfunktion* auf einer Menge X versteht man eine Abbildung $X \times X \xrightarrow{d} [0, \infty[$, die folgende Bedingungen erfüllt:

(0) Positivität: für alle $x, y \in X$ gilt $d(x, y) = 0 \iff x = y$;

(1) Symmetrie: für alle $x, y \in X$ gilt $d(x, y) = d(y, x)$;

(2) Dreiecks-Ungleichung: für alle $x, y, z \in X$ gilt $d(x, y) + d(y, z) \geq d(x, z)$.

1.2.01 Beispiele. (0) Die *Euklidische Metrik* auf dem $\mathbb{R}^n$ ist gegeben durch

$$d_2(\vec{x}, \vec{y}) = \left(\sum_{i<n} |x_i - y_i|^2 \right)^{\frac{1}{2}}$$

(1) Für $p \in]0, \infty[$ kann man allgemeiner die p-Metrik definieren durch

$$d_p(\vec{x}, \vec{y}) = \left(\sum_{i<n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

Je zwei Metriken d_p und d_q auf $\mathbb{R}^n$ sind äquivalent in dem Sinne, daß man zu jeder d_p-Kugel eine darin enthaltene d_q-Kugel finden kann und umgekehrt.

(2) Der Grenzwert der p-Metriken für $p \to \infty$ ist die Maximums-Metrik

$$d_{\max}(\vec{x}, \vec{y}) = \max\{ |x_i - y_i| : i < n \}$$

die ebenfalls zu den p-Metriken äquivalent ist.

(3) Auf jeder Menge X gibt es eine sogenannte *diskrete Metrik* d_D , die durch $d_D(x, y) = 1$ gdw. $x \neq y$ definiert ist. Positivität und Symmetrie sind unmittelbar klar, während die Dreiecksungleichung daraus folgt, daß zwei verschiedene Punkte nicht beide zu einem dritten gleich sein können.

Schränkt man speziell die 1-Metrik auf die Menge $\{0, 1\}^n \subseteq \mathbb{R}^n$ ein, so erhalten wir für den Abstand zweier Vektoren genau die Anzahl der Komponenten, in denen sie sich unterscheiden. Es stellt sich die Frage, ob diese Anzahl auch für andere endliche Mengen eine Metrik liefert, wenn wir die Analogie mit dem $\mathbb{R}^n$ nicht mehr ausnutzen können. Zunächst wollen wir ausnutzen, daß das Intervall $[0, \infty[$ bzgl. der Addition ein Monoid bildet, mit neutralem Element 0 .

1.2.02 Proposition. *Ist d eine Metrik auf B , so ist für $n \in \mathbb{N}$ die Einschränkung von*

$$(B \times B)^* \xrightarrow{\bar{d}} [0, \infty[$$

auf $(B \times B)^n \cong B^n \times B^n$ eine Metrik auf B^n .

Beweis: Dies folgt unmittelbar aus der komponenenweisen Definition von $\bar{d}$: für $x, y \in B^n$ gilt $\bar{d}(x,y) = \sum_{i<n} d(x_i, y_i)$. □

1.2.03 Definition. Die gemäß Proposition 1.2.02 aus der diskreten Metrik auf B herleitbaren Metriken auf B^n, $n \in \mathbb{N}$, heißen *Hamming-Abstände*. Wir bezeichnen sie mit d_H.

Unter dem *Minimalabstand* $d(C)$ einer Codierung $A \xrightarrow{C} B^n$ verstehen wir den minimalen Hamming-Abstand zweier verschiedener Codewörter, d.h.,

$$d(C) := \min\{\, d_H(C(a), C(a')) : a, a' \in A \text{ verschieden} \,\}$$

1.2.04 Bemerkung. Die Monoide $B^* \times B^*$ (mit komponentenweiser Konkatenation) sowie $(B \times B)^*$ sind außer für $B = \emptyset$ nicht isomorph (warum?). Dennoch läßt sich die diskrete Metrik von B auch zu einer Metrik auf B^* fortsetzen. Für Wörter $u, v \in B^*$ der Längen m und n formen wir rekursiv eine $(m+1) \times (n+1)$-Matrix L wie folgt:

$$\begin{aligned}
L_{i0} &= i \quad \text{für} \quad i \leq m \\
L_{0j} &= j \quad \text{für} \quad i \leq n \\
L_{ij} &= \min\{\, L_{(i-1)j} + 1, L_{i(j-1)} + 1, L_{(i-1)(j-1)} + d_D(u_{i-1}, v_{j-1}) \,\} \quad \text{sonst}
\end{aligned}$$

Die Komponente L_{mn} dieser Matrix liefert den sogenannten *Levenshtein-Abstand* $d_L(u,v)$.

Der Hamming-Abstand ist das wesentliche Kriterium für die Decodierung gestörter Codewörter nach der maximum likelihood Methode, die wir jetzt präzise definieren können:

1.2.05 Definition. $B^k \xrightarrow{C} B^n$ sei eine Block-Codierung mit Minimalabstand d.

(0) C *erkennt bis zu s Fehler*, wenn $s < d$ gilt.

(1) Die *maximum likelihood* Decodierung (MLD) für C wählt zufällig ein Codewort mit minimalem Hamming-Abstand zum empfangenen Wort aus.

(2) C *korrigiert bis zu t Fehler*, falls die MLD jedes durch höchstens t Fehler gestörte Wort richtig decodiert.

(3) C *korrigiert maximal t Fehler*, wenn C bis zu t Fehler korrigiert und mindestens ein durch $t+1$ Fehler gestörtes Wort falsch decodiert werden kann.

1.2.06 Satz. *$B^k \xrightarrow{C} B^n$ sei eine Block-Codierung mit Minimalabstand d.*

(0) *C korrigiert genau dann $\leq t$ Fehler, wenn gilt $2t < d$.*

(1) *C korrigiert genau dann maximal t Fehler, wenn gilt $2t < d \leq 2(t+1)$.*

Beweis: Man betrachte die entsprechenden "offenen Kugeln" um die Codewörter.

(0) ($\Rightarrow$). Sind x, z Codewörter mit $d_H(x,z) = d$, so gilt $d_H(x,y) \le t \Longrightarrow d_H(y,z) > t$ für alle $y \in B^n$. Insbesondere haben wir $d = d_H(x,z) > t$. Wir wählen nun $y_0 \in B^n$ derart, daß es sich in den ersten t Positionen, in denen sich x von z unterscheidet, mit z, und sonst mit x übereinstimmt. Wegen $d_H(x, y_0) = t$ folgt nun $d_H(y_0, z) > t$, und wegen $d_H(x,y_0) + d_H(y_0,z) = d_H(x,z) = d$ erhalten wir $2t < d$.

($\Leftarrow$). Für verschiedene Codewörter x, z betrachte $y \in B^n$ mit $d_H(x,y) \le t$. Wegen $d_H(x,y) + d_H(y,z) \ge d_H(x,z) \ge d > 2t$ folgt $d_H(y,z) > 2t - d_H(x,y) \ge 2t - t = t \ge d_H(x,y)$. Damit decodiert die MLD y zu x und nicht zu z.

(1) ($\Rightarrow$). Es gibt Codewörter x, z sowie $y \in B^n$ mit $d_H(x,y) = t+1$ und $d_H(y,z) \le t+1$. Also folgt $d \le d_H(x,z) \le d_H(x,y) + d_H(y,z) \le 2t+2$.

($\Leftarrow$). Sind x, z Codewörter mit $d_H(x,z) = d \le 2(t+1)$, so wird das Wort $y_0 \in B^n$, das in den ersten $t+1$ derjenigen Positionen, in denen sich z von x unterscheidet, mit z, und sonst mit x übereinstimmt, gelegentlich als z decodiert, denn es gilt $d_H(x,y_0) + d_H(y_0,z) = d_H(x,z) = d \le 2(t+1)$ und somit $d_H(y_0,z) \le 2(t-1) - d_H(x,y_0) = 2(t+1) - (t+1) = t+1$. □

1.2.07 Beispiele. (0) Für eine Wiederholungscodierung $B \xrightarrow{C} B^n$ (n ungerade) gilt $d(C) = n$, somit können bis zu $n-1$ Fehler erkannt bzw. bis zu $\frac{n-1}{2}$ Fehler korrigiert werden.

(1) Anfügen eines Paritätsbits liefert nur einen Minimalabstand $d = 2$, also werden Einzelfehler erkannt aber nicht korrigiert.

(2) Um einen Fehler korrigieren zu können, ist ein Minimalabstand von mindestens 3 erforderlich. Codierungen mit dieser Eigenschaft heißen auch *SEC-Codierungen.* Für $C_{(6)}$ und $C_{(8)}$ in Beispiel 1.1.00 ist diese Bedingung erfüllt. Andererseits erkennt $C_{(6)}$ bis zu zwei Fehler und $C_{(8)}$ bis zu drei Fehler.

Die volle Fähigkeit zur Fehlererkennung von $C_{(6)}$ und $C_{(8)}$ läßt sich aber nicht gleichzeitig mit der Fehlerkorrektur nutzen, da im ersten Fall manche Doppelfehler von Einzelfehlern nicht unterscheidbar sind, während im zweiten Fall alle Dreifachfehler denselben Effekt haben wie Einzelfehler. Einen ähnlichen Effekt hatten wir bereits in der Einführung gesehen; er tritt z.B. auch für die Wiederholungcodierung der Blocklänge 5 auf, die entweder Doppelfehler korrigieren oder Vierfachfehler erkennen kann, aber nicht beides gleichzeitig.

Um wenigstens einen Teil der Fehlererkennungsfähigkeiten zu nutzen, können wir im Falle einer Rückmeldungsmöglichkeit für fehlerhafte Blöcke wie folgt verfahren: korrigiert eine Codierung maximal t Fehler, so bestimmen wir die Maximalzahl erkennbarer Fehler, deren Fehlermuster nicht von $\le t$ Fehlern erzeugt werden kann. Finden wir ein Fehlermuster dieses Typs, melden wir den entsprechenden Block zur nochmaligen Übertragung. Diese Überlegung führt zu folgender Begriffsbildung:

1.2.08 Definition. Eine Codierung $B^k \xrightarrow{C} B^n$ kann *gleichzeitig t Fehler korrigieren und $s > t$ Fehler erkennen*, wenn für alle Codewörter $x \neq z$ und für jedes $y \in B^n$ gilt

$$d_H(x,y) \leq t \implies d_H(y,z) > s \qquad \text{oder äquivalent} \qquad d_H(x,y) \leq s \implies d_H(y,z) > t$$

1.2.09 Satz. *Eine Codierung mit Minimalabstand d kann genau dann gleichzeitig t Fehler korrigieren und $s > t$ Fehler erkennen, wenn $t + s < d$ gilt.*

Beweis: $(\Rightarrow)$. Nach Definition klar: die abgeschlossene t-Kugel um ein Codewort schneidet die abgeschlossene s-Kugel um ein anderes Codewort nicht.

$(\Leftarrow)$. Ausgehend von $t + s < d$, $t < s$, einem Codewort x und einem Wort $y \in B^n$ mit $d_H(x,y) \leq t$, erfüllt jedes von x verschiedene Codewort z die Ungleichung $d_H(x,y) + d_H(y,z) \geq d_H(x,z) \geq d > t + s$, also auch $d_H(y,z) > t + s - d_H(x,y) \geq t + s - t = s$. □

1.2.10 Bemerkung. Wenn C gleichzeitig t Fehler korrigiert und $s > t$ Fehler erkennt, so kann für ein Wort y mit Hamming-Abstand $\geq t$ zu jedem Codewort nur geschlossen werden, daß zwischen $t+1$ und s Fehler aufgetreten sind. Nur im Fall $s = t + 1$ können wir die Fehlerzahl genauer fixieren.

1.3 Kanalkapazität und Shannons 2. Satz

Eine Informationsquelle ware als Wahrscheinlichkeitsverteilung π auf einer endlichen Mengen A eingeführt worden. Wie kann man π, interpretiert als Zeilenvektor, nun in eine Wahrscheinlichkeitsverteilung auf einem Alphabet B transformieren? Einfach, mittels einer sogenannten *stochastischen* $(A \times B)$-Matrix, deren Komponenten aus $[0,1]$ stammen und deren Zeilensummen alle den Wert 1 haben! Man beachte, daß solche Matrizen unter Multiplikation abgeschlossen sind. Sie liefern uns das mathematische Modell für den "Kanal":

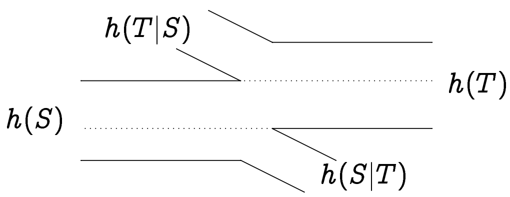
1.3.00 Definition. Eine stochastische $(A \times B)$-Matrix P heißt *diskreter gedächtnisloser Informationskanal* (oder kurz *Kanal*) von A nach B; wir schreiben dafür $A \overset{P}{\Longrightarrow} B$.

1.3.01 Bemerkungen. (0) Die Bezeichnungen in der Literatur sind in diesem Bereich etwas irritierend, da an Spaltenvektoren orientiert. So wird für die $\langle a,b\rangle$-Komponente $P_{a,b}$ der Matrix P meist $P(b|a)$ geschrieben (beachte die Reihenfolge der Argumente). Diese Zahl interpretieren wir als die bedingte Wahrscheinlichkeit für die Ausgabe $b \in B$ unter der Voraussetzung, daß die Eingabe $a \in A$ erfolgt ist. Bitte nicht $P(a,b)$ schreiben, dieser Ausdruck ist in der Literatur schon für Verbundwahrscheinlichkeiten belegt, s.u.

(1) Eine Informationsquelle auf A kann selbst als Kanal aufgefaßt werden, nämlich als Kanal von der Singleton-Menge $1 = \{*\}$ nach A. Dies entspricht einer $(1 \times A)$-Matrix, d.h., einem Zeilenvektor, den wir mit kleinen griechischen Buchstaben bezeichnen, etwa mit π. Nun ist $\pi(a) = \pi(a|*)$ die bedingte Wahrscheinlichkeit für die Ausgabe a sofern die sichere Eingabe $*$ erfolgt ist.

1.3.02 Konvention. Ist $S = \langle A, \pi\rangle$ eine Informationsquelle und ist P ein Kanal von A nach B, so bezeichnen wir die Informationsquelle $\langle B, \pi \cdot P\rangle$ immer mit T, und die Wahrscheinlichkeitsverteilung $\pi \cdot P$ mit ρ.

Unser Ziel ist es zu verstehen, wie ein Kanal P von A nach B die natürlichen Entropien $h(S)$ und $h(T)$ in Beziehung setzt. Das folgende Flußdiagramm veranschaulicht die Idee:



Von der in den Kanal eingespeisten Entropie $h(S)$ geht ein gewisser Anteil $h(S|T)$ verloren, während ein anderer Anteil $h(T|S)$ durch "Rauschen" hinzukommt. Die Differenzen $h(S) - h(S|T)$ und $h(T) - h(T|S)$ stimmen überein und bilden den nutzbaren Anteil der Entropie. Um den Verlust an Entropie $h(S|T)$ bestimmen zu können, brauchen wir für eine gegebene Wahrscheinlichkeitsverteilung π auf A

- die Verbundwahrscheinlichkeiten für das Auftreten der Eingabe-Ausgabe-Paare $\langle a, b\rangle$, zweckmäßigerweise in Form einer $(A \times B)$-Matrix, die von π abhängt;
- die bedingten Wahrscheinlichkeiten für die Eingabe a unter der Voraussetzung, daß die Ausgabe b aufgetreten ist, zweckmäßigerweise in Form einer $(B \times A)$-Matrix $Q(\pi)$, die ebenfalls von π abhängt.

Bezeichnet $\Delta(\pi)$ die Matrix mit den Einträgen aus π in der Diagonale, so liefert uns die Matrix $\Delta(\pi) \cdot P$ die gesuchten Verbundwahrscheinlichkeiten. In der Literatur findet man hierfür die irritierende Schreibweise $P(a, b)$, welche die Abhängigkeit von der Wahrscheinlichkeitsverteilung π auf A völlig ignoriert. Es gilt

$$P(a, b) = \pi(a)P(b|a) = \pi(a)P_{a,b} \ ,$$

Offenbar hat die Wahrscheinlichkeitsverteilung $\rho = \pi \cdot P$ genau die Spaltensummen von $\Delta(\pi) \cdot P$ als Komponenten. Da die Wahrscheinlichkeit des Eingabe-Ausgabe-Paares $\langle a, b\rangle$ gleich der Wahrscheinlichkeit des Ausgabe-Eingabe-Paares $\langle b, a\rangle$ ist, lautet die Spezifikation für die gesuchte Matrix $Q(\pi)$

$$\left(\Delta(\pi) \cdot P\right)^{\top} = \Delta(\rho) \cdot Q(\pi)$$

Da π den Wert Null annehmen darf, ist dies auch für ρ möglich. Daher sind die entsprechenden Zeilen von $Q(\pi)$ nicht eindeutig bestimmt. Die übrigen Zeilen erhalten wir, indem wir die entsprechenden Spalten von $\Delta(\pi) \cdot P$ normieren, d.h., durch $\rho(b)$ dividieren, und dann transponieren. Um wieder eine stochastischen Matrix zu erhalten, setzen wir

$$Q(\pi)_{b,a} = \begin{cases} \frac{P(a,b)}{\rho(b)} = \frac{\pi(a)}{\rho(b)} P(b|a) = \frac{\pi(a)}{\rho(b)} P_{a,b} & \text{falls } \rho(b) \neq 0 \\ \pi(a) & \text{sonst} \end{cases}$$

Die Verwirrung steigert sich noch, wenn die Komponenten der Matrix $Q(\pi)$ mit $P(a|b)$ bezeichnet werden, was wiederum die Abhängigkeit von π außer Acht läßt. Wir hoffen, mit der Matrix $Q(\pi)$ dem Durcheinander von $P(b|a)$, $P(a, b) = P(b, a)$ und $P(a|b)$ Einhalt zu gebieten.

1.3.03 Beispiele. (0) Dem binären symmetrischen Kanal mit Fehlerwahrscheinlichkeit 0.1 entspricht die Kanalmatrix

$$P = \begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix}$$

Die Verteilung $\pi = (0.7, 0.3)$ liefert nun

$$\Delta(\pi) \cdot P = \begin{pmatrix} 0.7 & 0 \\ 0 & 0.3 \end{pmatrix} \begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix} = \begin{pmatrix} 0.63 & 0.07 \\ 0.03 & 0.27 \end{pmatrix}$$

woraus sich durch Spaltensummenbildung $q = (0.66, 0.34)$ ergibt. Spaltennormierung und Transposition führt zu

$$Q(\pi) = \begin{pmatrix} 21/22 & 1/22 \\ 7/34 & 27/34 \end{pmatrix}$$

(1) Der auslöschende Kanal von $A = \{0, 1\}$ nach $B = \{0, 1, ?\}$ mit Auslöschwahrscheinlichkeit 0.1 wird durch die Kanalmatrix

$$P = \begin{pmatrix} 0.9 & 0 & 0.1 \\ 0 & 0.9 & 0.1 \end{pmatrix}$$

beschrieben. Die Verteilung $\pi = (1/3, 2/3)$ auf A liefert

$$\Delta(\pi) \cdot P = \begin{pmatrix} 1/3 & 0 \\ 0 & 2/3 \end{pmatrix} \begin{pmatrix} 0.9 & 0 & 0.1 \\ 0 & 0.9 & 0.1 \end{pmatrix} = \begin{pmatrix} 0.3 & 0 & 0.0\bar{3} \\ 0 & 0.6 & 0.0\bar{6} \end{pmatrix}$$

woraus wir $q = (0.3, 0.6, 0.1)$ erhalten, sowie

$$Q(\pi) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1/3 & 2/3 \end{pmatrix}$$

Dagegen liefert die Eingangsverteilung $\pi = (0, 1)$

$$\Delta(\pi) \cdot P = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0.9 & 0 & 0.1 \\ 0 & 0.1 & 0.9 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0. \\ 0 & 0.9 & 0.1 \end{pmatrix}$$

woraus wir $q = (0, 0.9, 0.1)$ erhalten, sowie

$$Q(\pi) = \begin{pmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}$$

Zur Ermittlung der im Kanal verlorenen Entropie $h(S|T)$ bestimmen wir zunächst den Verlust an Information über das Eingabezeichen bei vorgegebener Ausgabe $b \in B$. Dabei wollen wir vereinbaren, undefinierte Summanden wegzulassen, vergl. Abschnitt 0.3.

$$h(S|b) = -\sum_{a \in A} P(a|b) \ln P(a|b) = -\sum_{a \in A} Q(\pi)_{b,a} \ln Q(\pi)_{b,a}$$

Der Mittelwert dieser Entropien über alle Ausgabewerte $b \in B$ ist der Informationsverlust im Kanal:

$$h(S|T) = \sum_{b \in B} \rho(b) h(S|b) = -\sum_{b \in B} \sum_{a \in A} \rho(b) P(a|b) \ln P(a|b) = -\sum_{b \in B} \sum_{a \in A} P(a, b) \ln P(a|b)$$

1.3.04 Definition. Für einen Kanal P von A nach B und eine Verteilung π auf A heißt die Differenz $i(S, T) := h(S) - h(S|T)$ *natürliche Transinformation.*

1.3.05 Proposition. $i(S, T) = \sum_{a \in A} \sum_{b \in B} P(a, b) \ln \frac{P(a,b)}{\pi(a)\rho(b)} = h(T) - h(T|S)$

Beweis:

$$\begin{aligned} i(S,T) = h(S) - h(S|T) &= -\sum_{a\in A} \pi(a)\ln \pi(a) + \sum_{a\in A}\sum_{b\in B} P(a,b)\ln P(a|b) \\ &= \sum_{a\in A}\sum_{b\in B} P(a,b)\left(-\ln \pi(a) + \ln \frac{P(a,b)}{\rho(b)}\right) \\ &= \sum_{a\in A}\sum_{b\in B} P(a,b)\ln \frac{P(a,b)}{\pi(a)\rho(b)} \end{aligned}$$

Der Rest folgt aus Symmetriegründen. □

1.3.06 Definition. Unter der *Kapazität* $\kappa(P)$ eines Kanals P von A nach B verstehen wir die mit $1/\ln|A|$ gewichtete maximale Transinformation, d.h.,

$$\kappa(P) = \frac{1}{\ln|A|}\max\{\, i(S,T) : \pi \in W(A)\,\}$$

Der Faktor $1/\ln|A|$ bedeutet, daß die in $i(S,T)$ vorkommenden natürlichen Logarithmen durch Logarithmen zur Basis $|A|$ zu ersetzen sind.

1.3.07 Beispiel. Wir setzen Beispiel 1.3.03(1) des auslöschenden Kanals fort. Allgemein ergibt sich für $\pi = (\pi_0, \pi_1)$

$$\Delta(\pi)\cdot P = \begin{pmatrix} 0.9\pi_0 & 0 & 0.1\pi_0 \\ 0 & 0.9\pi_1 & 0.1\pi_1 \end{pmatrix}$$

woraus wir $q = (0.9\pi_0, 0.1\pi_1, 0.1)$ erhalten, sowie

$$\begin{aligned} i(S,T) &= 0.9\pi_0 \ln\frac{0.9\pi_0}{\pi_0\cdot 0.9\pi_0} + 0.9\pi_1\ln\frac{0.9\pi_1}{\pi_1\cdot 0.9\pi_1} + 0.1\pi_0\ln\frac{0.1\pi_0}{\pi_0\cdot 0.1} + 0.1\pi_1\ln\frac{0.1\pi_1}{\pi_1\cdot 0.1} \\ &= 0.9\left(\pi_0\ln\frac{1}{\pi_0} + \pi_1\ln\frac{1}{\pi_1}\right) = 0.9h(S) \end{aligned}$$

Der Maximalwert von $H(S) = h(S)/\ln 2$ beträgt 1, daher ergibt sich $\kappa(P) = 0.9$. Dieses Ergebnis bestätigt die Begriffsbildungen dieses Abschnitts als sinnvoll.

1.3.08 Satz. *Der binäre symmetrische Kanal mit Fehlerwahrscheinlichkeit p hat die Kapazität $\kappa = 1 - H(p, 1-p)$.*

Beweis: Eine Wahrscheinlichkeitsverteilung $\pi = (\pi_0, \pi_1)$ auf $A = \{0,1\}$ liefert

$$\Delta(\pi)\cdot P = \begin{pmatrix} (1-p)\pi_0 & p\pi_0 \\ p\pi_1 & (1-p)\pi_1 \end{pmatrix}$$

und somit

$$\begin{aligned} H(T|S) &= -\sum_{a\in A}\sum_{b\in B} P(a,b)\log_2 P(b|a) \\ &= -p(\pi_0+\pi_1)\log_2 p - (1-p)(\pi_0+\pi_1)\log_2(1-p) = H(p,1-p) \end{aligned}$$

Dieser Ausdruck ist unabhängig von π, also folgt

$$\kappa = \kappa(P) = \max\{H(\langle B, \pi \cdot P\rangle) - H(p, 1-p) : \pi \in W(A)\}$$

Die (binäre) Entropie der Quelle $T = \langle B, \pi \cdot P\rangle$ ist sicher durch 1 beschränkt, sie nimmt diesen Wert für die Gleichverteilung $\pi = (0.5, 0.5)$ auch an, denn in diesem Fall gilt $q = \pi \cdot P = (0.5, 0.5)$. □

1.3.09 Beispiele. (0) Für $p = 0.001$ ergibt sich eine Kapazität $\kappa = 1 - H(0.001, 0.999) \cong 0.989$. Man vergleiche dies mit dem Beispiel aus Abschnitt 1.0!

(1) Für $p = 0.5$ ergibt sich eine Kapazität $\kappa = 0$, was den Ausschluß dieses Falles rechtfertigt.

1.3.10 Satz. (Shannons 2. Satz) *Für einen binären symmetrischen Kanal $A \xRightarrow{P} B$ der Kapazität κ gibt es eine Folge $\langle C_n\rangle_{n>0}$ von Blockcodierungen $A \xrightarrow{C_n} B^n$ mit*

$$\rho(C_n) \to \kappa^- \qquad \text{und} \qquad P_{\text{err}}(C_n) \to 0 \qquad \text{für} \qquad n \to \infty$$

Beweis: (Skizze) Betrachte $\epsilon > 0$ mit $\kappa - \epsilon$ rational, etwa $\kappa - \epsilon = u/v$ mit $u, v \in \mathbb{N}$. Wählt man für $m \in \mathbb{N}$ aus $\{0,1\}^{mv}$ beliebige $M = 2^{mv(\kappa-\epsilon)}$ Wörter aus, so erhält man eine Codierung K_m der Coderate $\kappa - \epsilon$. Wir fassen $P_{\text{err}}(K_m)$ als Zufallsvariable mit dem Erwartungswert $\tilde{P}_{\text{err}}(m)$ auf. Der technisch aufwendige Teil des Beweises besteht darin, $\lim_{m\to\infty} \tilde{P}_{\text{er}}(m) = 0$ zu zeigen. Sobald dies bewiesen ist, sind wir fertig: zu $\epsilon' > 0$ wählen wir $m' \in \mathbb{N}$ mit $\tilde{P}_{\text{err}}(m') < \epsilon'$. Ein Erwartungswert kleiner als ϵ' impliziert die Existenz einer konkreten Codierung $C_{m'}$ mit $P_{\text{err}}(C_{m'}) < \epsilon'$, deren Rate den Wert $\kappa - \epsilon$ hat. □

1.3.11 Bemerkungen. (0) Der Beweis ist nicht konstruktiv in Bezug auf die Codierung $C_{m'}$ mit $P_{\text{err}}(C_{m'}) < \epsilon'$.

(1) Die meisten praktischen Codierungen sind noch weit von der theoretischen Schranke entfernt.

(2) Wolfowitz (59) hat eine Umkehrung dieses Satzes bewiesen: für jede Folge von Codierungen $\langle C_n\rangle_{n>0}$ gilt: $\lim_{n\to\infty} \rho(C_n) > \kappa$ impliziert $\lim_{n\to\infty} P_{\text{err}}(C_n) = 1$.

2 Codierung und Algebra – Einführung

2.0 Algebraische Strukturen: von Halbgruppen zu Vektorräumen

Die linearen Codes des nächsten Abschnitts basieren auf dem Begriff des Vektorraums. Während Vektorräume über den Körpern der reellen oder der complexen Zahlen sicher allgemein bekannt sind, werden uns dort Vektorräume über anderen, speziell endlichen Körpern interessieren. Da bietet sich eine geballte Rekapitulation einiger Begriffe aus der Algebra geradezu an.

2.0.00 Definition. Unter einer *binären Operation* auf einer Menge X verstehen wir eine Abbildung $X \times X \xrightarrow{m} X$. Die zu m *duale Operation* m^{op} ist durch die Verknüpfung mit der twist-Abbildung gegeben, die die Komponenten eines geordneten Paares vertauscht

$$\begin{tikzcd} X \times X \arrow[rr, "\mathrm{twist}"] \arrow[dr, "m^{\mathrm{op}}"'] & & X \times X \arrow[dl, "m"] \\ & X & \end{tikzcd}$$

Wir nennen die Operation *kommutativ*, wenn $m = m^{\mathrm{op}}$ gilt.

Schwächt man den in Abschnitt 0.1 eingeführten Begriff des Monoids ab, indem man auf die Existenz des neutralen Element verzichtet, erhält man den Begriff der *Halbgruppe*. Monoide sind also genau die Halbgruppen, die ein neutrales Element besitzen. Entsprechend muß ein *Halbgruppenhomomorphismus* nur die binäre Operation bewahren. Nun wollen wir den Begriff des Monoids zum Begriff der Gruppe einengen indem wir zusätzlich die Existenz von Inversen verlangen.

2.0.01 Definition. Ein Element a eines Monoids $\langle M, m, e\rangle$ heißt *linksinvertierbar* (bzw. *rechtsinvertierbar*), falls ein $b \in M$ existiert mit $m(b,a) = e$ (bzw. $m(a,b) = e$). Das Element b heißt *Linksinverses* (bzw. *Rechtsinverses*) von a. Wir nennen a *invertierbar*, falls a linksinvertierbar und rechtsinvertierbar ist.

2.0.02 Proposition. *Gilt in einem Monoid* $m(b,a) = e = m(a,c)$, *so folgt bereits* $b = c$. □

2.0.03 Definition. Unter einer *Gruppe* verstehen wir einen Monoid, in dem jedes Element invertierbar ist. Im Allgemeinen benutzen wir die "multiplikative" Infixschreibweise $x \cdot y$ für die Operation m und schreiben x^{-1} für das eindeutig bestimmte Inverse. Es gelten die bekannten Regeln für Exponenten aus $\mathbb{Z}$. Eine Gruppe heißt *abelsch*, wenn ihre binäre Verknüpfung kommutativ ist. In diesem Fall benutzen wir häufig die additive Infixschreibweise $x + y$ für die Operation und bezeichnen das Inverse von x mit $-x$. Weiterhin ist dann $x - y$ eine Abkürzung für $x + (-y)$.

Wie Proposition 0.1.03 zeigt, kann es in einer Halbgruppe höchstens ein neutrales Element geben. Dennoch braucht ein Halbgruppenhomomorphismus zwischen Monoiden das neutrale Element nicht erhalten (HA 2). Für Gruppen ist dagegen die Welt wieder in Ordnung.

2.0.04 Proposition. *Ein Halbgruppenhomomorphismus zwischen Gruppen bewahrt Inverse und auch das neutrale Element.* $\square$

Dieses Ergebnis rechtfertigt, Gruppen nur durch Angabe der binären Operation zu spezifizieren: das neutrale Element und die Inversen sind eindeutig bestimmt und werden automatisch von solchen Funktionen bewahrt, welche die binäre Operation bewahren. Diese heißen in diesem Zusammenhang natürlich *Gruppenhomomorphismen*.

Außer den strukturerhaltenden Abbildungen sind in der Algebra auch *Unterstrukturen* sowie *Faktorstrukturen* von Bedeutung. Bei Unterstrukturen handelt es sich gemeinhin um Teilmengen, auf die sich die relevanten Operationen der Algebra einschränken lassen, oder anders ausgedrückt, die unter den relevanten Operationen abgeschlossen sind. So ist etwa eine *Unterhalbgruppe* eine bzgl. der Multiplikation $\cdot$ abgeschlossene Teilemenge, ein *Untermonoid* muß zusätzlich das neutrale Element enthalten (0-stellige Operation!), während eine *Untergruppe* darüberhinaus auch unter Inversenbildung abgeschloßen sein muß. In Einzelfällen mögen einige Abschlußeigenschafen redundant sein, aber grundsätzlich sind alle Operationen zu betrachten. Wichtig is noch, daß jede Teilmenge einer Algebra zu einer kleinsten sie umfassenden Unteralgebra erweitert werden kann: man bildet einfach iterativ den Abschluß under ten relevanten Operationen. Wir werden darauf verzichten, für alle nachfolgend definierten algebraischen Strukturen explizit Unterstrukturen anzugeben, sondern nur die für uns hier interessanten Eigenschaften auflisten.

Faktorstrukturen werden für uns sogar noch wichtiger sein. Sie sind zwar aus abstrakter Sicht nicht schwieriger zu verstehen, aber ihre Trägermengen sind komplizierter. Es handelt sich dabei um *Faktormengen*, d.h., Mengen von Äquivalenzklassen geeigneter Äquivalenzrelationen. Daher sind die Elemente von Faktoralgebren selbst wieder Mengen. Oft läßt sich aber diese Komplexität durch Auswahl geeigneter *Repräsentanten*(s.u.) wieder reduzieren. Nachfolgend werden wir die jeweiligen Faktorstrukturen detailliert angeben.

2.0.05 Proposition. (0) *Untergruppen sind unter beliebigen Durchschnitten abgeschlossen.*

(1) *Jede Teilmenge A einer Gruppe G ist in einer minimalen Untergruppe $< A >$ von G enthalten.*

(2) *Für jedes $a \in G$ gilt $< \{a\} >= \{ a^n : n \in \mathbb{Z} \}$.* $\square$

2.0.06 Definition. Ist für ein Element a einer Gruppe G die Untergruppe $< \{a\} >$ endlich, so bezeichnen wir die Zahl $\mathrm{ord}(a) = | < \{a\} > |$ als *Ordnung* von a.

2.0.07 Proposition. *Erzeugt $a \in G$ eine endliche Untergruppe, so gilt $a^{\mathrm{ord}(a)} = e$ und $a^k \neq e$ für $0 < k < \mathrm{ord}(a)$.* $\square$

2.0.08 Proposition. *Ist U eine Untergruppe einer abelschen Gruppe G, so wird durch*

$$x \sim y \;:\Longleftrightarrow\; y - x \in U$$

eine Äquivalenzrelation auf G definiert. Die Äquivalenzklasse von $x \in G$ ist gegeben durch

$$[x] = x + U := \{\, x + u : u \in U \,\}$$

und ist als Menge isomorph zu U. Auf der Menge G/U aller Äquivalenzklassen wird durch

$$(x + U) + (y + U) := (x + y) + U$$

eine Gruppenstruktur definiert, die sogenannte Faktorgruppe *von G nach U. Schließlich ist die sogenannte* kanonische Abbildung $x \to x + U$ *ein Gruppenhomomorphismus.* □

2.0.09 Beispiel. $\langle \mathbb{N}, +, 0 \rangle$ ist ein Monoid, aber außer 0 hat kein Element ein (additives) Inverses. Fügt man die fehlenden Inversen hinzu, erhält man die Gruppe $\langle \mathbb{Z}, + \rangle$ der *ganzen Zahlen.*

Für jedes $n \in \mathbb{N}$ ist die Menge $n \cdot \mathbb{Z} := \{\, n \cdot z : z \in \mathbb{Z} \,\}$ eine Untergruppe: $0 = n \cdot 0 \in n \cdot \mathbb{Z}$ und $n \cdot x + n \cdot y = n \cdot (x + y) \in n \cdot \mathbb{Z}$. Die Faktorgruppe $\mathbb{Z}/(n \cdot \mathbb{Z})$ wird abkürzend mit $\mathbb{Z}_n$ bezeichnet. Ihre Elemente sind eigentlich die Äquivalenzklassen der Relation

$$x \sim_n y \;:\Longleftrightarrow\; y - x \quad \text{ist ein Vielfaches von } n$$

Für $n = 0$ besteht die Äquivalenzklasse von x aus einem Element, $[x] = x + 0 \cdot \mathbb{Z} = \{x\}$, und es gibt unendlich viele dieser Klassen. Für $n \neq 0$ gibt es nur n Äquivalenzklassen, die dafür alle unendlich sind. Etwa für $n = 3$ haben wir $[x] = x + 3 \cdot \mathbb{Z} = \{\ldots, x-6, x-3, x, x+3, x+6, \ldots\}$. So entstehen die drei Klassen $[0] = [3] = \ldots$, $[1] = [4] = \ldots$ und $[2] = [5] = \ldots$.

Um nicht mit solch potentiell unhandlichen Mengen hantieren zu müssen, behilft man sich mit einem Trick: man wählt aus jeder Äquivalenzklasse einen sogenannten *Repräsentanten* aus und rechnet dann mit diesen Repräsentanten. Die Menge der Repräsentanten korrespondiert natürlich bijektiv zu der Menge der Äquivalenzklassen. Es stellt sich nur die Frage, welchen Repräsentanten man wählen soll. Bei einer beliebigen Äquivalenzrelation auf einer beliebigen Menge ist kein Element einer Äquivalenzklasse vor den anderen ausgezeichnet. Nicht so hier: immerhin gibt es auf der Menge $\mathbb{Z}$ eine natürliche Ordnung. Wir können also beispielsweise aus jeder Äquivalenzklasse das kleinste nicht-negative Element auswählen. In diesem Fall erhalten wir für $\mathbb{Z}_n$ die Menge $n = \{0, 1, \ldots, n-1\}$, also die Menge der möglichen Reste bei der Division durch n. Konkret gilt etwa

$$\mathbb{Z}_6 = \{0, 1, 2, 3, 4, 5\}$$

Die Addition auf $\mathbb{Z}_6$ ist durch die gewöhnliche Addition modulo 6 gegeben. Insbesondere gilt $-0 = 0$, $-1 = 5$, $-2 = 4$ und $-3 = 3$. Weiterhin erzeugt 0 die Untergruppe $\{0\}$ der Ordnung 1, 1 und 5 erzeugen die gesamte Gruppe $\mathbb{Z}_6$ der Ordnung 6, 2 und 4 erzeugen die Untergruppe $\{0, 2, 4\}$ der Ordnung 3, und 3 erzeugt die Untergruppe $\{0, 3\}$ der Ordung 2.

Als nächstes wollen wir uns mit algebraischen Strukturen beschäftigen, auf denen zwei kompatible binäre Operationen definiert sind.

2.0.10 Definition. Ein *Ring* $\langle R, +, \cdot, 1\rangle$ besteht aus einer Menge R, zwei binären Operationen $+$ und $\cdot$ auf R sowie einem Element 1, so daß gilt

(R0) $\langle R, +\rangle$ ist eine abelsche Gruppe;

(R1) $\langle R, \cdot, 1\rangle$ ist ein (nicht notwendig kommutatives) Monoid;

(R2) es gelten die *Distributivgesetze* $x \cdot (y + z) = x \cdot y + x \cdot z$ und $(x + y) \cdot z = x \cdot z + y \cdot z$ für alle $x, y, z \in R$.

Ein Ring $\langle R, +, \cdot, 1\rangle$ heißt kommutativ, wenn $\cdot$ diese Eigenschaft hat. Unter einem *Ringhomomorphismus* verstehen wir eine Abbildung, die bzgl. $+$ ein Gruppenhomomorphismus und bzgl. $\cdot$ und 1 ein Monoidhomomorphismus ist.

Die eigentliche Bedeutung der Distributivgesetze liegt darin, daß für jedes Element $x \in R$ die Abbildungen $x \cdot -$ und $- \cdot x$ Gruppenhomomorphismen auf $\langle R, +\rangle$ sind. Zwar handelt es sich bei der Operation $\cdot$ i.A. nicht um einen Gruppenhomomorphismus von $\langle R, +\rangle \times \langle R, +\rangle$ mit der komponentenweisen Struktur nach $\langle R, +\rangle$, aber zumindest Festhalten jeder der beiden Komponenten liefert einen solchen. Man nennt $\cdot$ daher auch *bi-additiv*. Ersetzt man das kartesische Produkt $\times$ zwischen abelschen Gruppen durch das sogenannte *Tensorprodukt* $\otimes$, so induziert $\cdot$ tatsächlich einen Gruppenhomomorphismus von $\langle R, +\rangle \otimes \langle R, +\rangle$ nach $\langle R, +\rangle$ und man kann einen Ring als ein Monoid in der Kategorie der abelschen Gruppen bzgl. des Tensorprodukts auffassen. Leider ist die konkrete Beschreibung des Tensorprodukts zweier abelscher Gruppen etwas kompliziert und führt über den Rahmen dieser Vorlesung hinaus.

Wie für Gruppen wollen wir Faktorringe betrachten. Leider lassen sich diese nicht mittels eines Unterrings (gleichzeitig Untergruppe und Untermonoid) eines Rings konstruieren. Wir brauchen eine etwas stärkere Unterstruktur.

2.0.11 Definition. Eine Teilmenge I eines Rings $\langle R, +, \cdot, 1\rangle$ heißt *Linksideal* (bzw. *Rechtsideal*), falls gilt

(I0) I ist eine Untergruppe von $\langle R, +\rangle$;

(I1) Die Einschränkung $I \times R \xrightarrow{\cdot} R$ (bzw. $R \times I \xrightarrow{\cdot} R$) faktorisiert durch I, d.h., für alle $i \in I$ und alle $r \in R$ gilt $i \cdot r \in I$ (bzw. $r \cdot i \in I$).

Unter einem *Ideal* verstehen wir ein Links- und Rechtsideal.

2.0.12 Proposition. *Ist I ein Ideal eines Rings $\langle R, +, \cdot, 1\rangle$, so definiert*

$$(x + I) \cdot (y + I) := (x \cdot y) + I$$

eine binäre Operation auf der Faktorgruppe R/I, die sie zu einem Ring $\langle R/I, +, \cdot, 1+I\rangle$ macht, dem sogenannten Faktorring *von $\langle R, +, \cdot, 1\rangle$ nach I. Weiterhin ist die kanonische Abbildung $x \to x + I$ ein Ringhomomorphismus.* □

Wie oben angedeutet, erzeugt jede Teilmenge A eines Rings ein kleinstes sie enthaltendes Ideal. Im Allgemeinen können diese Ideale recht kompliziert aussehen. Glücklicherweise brauchen wir im Folgenden nur sehr einfache spezielle Ideale.

2.0.13 Proposition. *Ist $\langle R, +, \cdot, 1\rangle$ ein kommutativer Ring, so ist für jedes Element $x \in R$ die Menge $x \cdot R := \{x \cdot r : r \in R\}$ ein Ideal, das sogenannte von x erzeugte* Hauptideal.

Beweis: Wegen $x \cdot r + x \cdot s = x \cdot (r + s) \in x \cdot R$ und $x \cdot r + x \cdot (-r) = x \cdot (r - r) = 0 \in x \cdot R$ ist $x \cdot R$ eine Untergruppe. Die Idealeigenschaft folgt ebenfalls sofort: $(x \cdot r) \cdot s = x \cdot (r \cdot s) \in x \cdot R$.
□

2.0.14 Beispiel. Die Mengen C der complexen, $\mathbb{R}$ der reellen, Q der rationalen und $\mathbb{Z}$ der ganzen Zahlen sind offenbar kommutative Ringe, mit der gewöhnlichen Addition und Multiplikation. Weiterhin sind die Untergruppen $n \cdot \mathbb{Z}$ von $\mathbb{Z}$ (vergl. Beispiel 2.0.09) natürlich Hauptideale, und damit sind die Faktorgruppen $\mathbb{Z}_n = \mathbb{Z}/n \cdot \mathbb{Z}$ sogar Faktorringe, mit der folgenden Multiplikation von Äquivalenzklassen

$$(x + n \cdot \mathbb{Z}) \cdot (y + n \cdot \mathbb{Z}) := (x \cdot y) + n \cdot \mathbb{Z}$$

Bezüglich der Standardrepräsentanten ergibt sich die Multiplikation modulo n.

2.0.15 Definition. Ein *Körper* ist ein kommutativer Ring $\langle F, +, \cdot, 1\rangle$ mit der Zusatzeigenschaft, daß $\langle F - \{0\}, \cdot\rangle$ eine Gruppe ist.

Insbesondere sind damit Körper per definitionem *nullteilerfrei*, d.h., aus $a \cdot b = 0$ folgt $a = 0$ oder $b = 0$.

2.0.16 Beispiel. Die Mengen $\mathbf{Q}$ und $\mathbb{R}$ der rationalen bzw. reellen Zahlen sind Körper bzgl. Addition und Multiplikation. Entsprechendes gilt für die Menge $\mathbf{C}$ der complexen Zahlen mit der komponentenweisen Addition und der Multiplikation $\langle a, b\rangle \cdot \langle c, d\rangle = \langle ac - bd, ad + bc\rangle$. Dagegen ist $\mathbb{Z}$ zwar ein kommutativer Ring, aber kein Körper: 2 hat kein multiplikatives Inverses.

Wie anfangs erwähnt, gilt unser besonderes Interesse endlichen Körpern. Zwar sind alle unsere Bespiele von Körpern bisher unendlich, aber die Faktorring-Konstruktion liefert uns eine Vielzahl endlicher Körper:

2.0.17 Proposition. *Der Faktorring $\mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$ ist genau dann ein Körper (mit p Elementen), wenn p eine Primzahl ist.*

Beweis: Wir müssen die Abgeschlossenheit von $\{1, 2, \ldots, p-1\}$ unter der Multiplikation (modulo p) und die Existenz multiplikativer Inverser nachweisen genau dann wenn p eine Primzahl ist.

($\Rightarrow$). Ist p keine Primzahl, etwa $r \cdot s = p$ in $\mathbb{N}$ mit $1 \le r, s \le p - 1$, dann gilt $r \cdot s = 0$ in $\mathbb{Z}_p$, und die Menge $\{1, 2, \ldots, p - 1\}$ ist nicht unter Multiplikation abgeschlossen.

($\Leftarrow$). p sei eine Primzahl. Ist für $1 \leq r, s \leq p-1$ das Produkt $r \cdot s$ in $\mathbb{Z}$ ein Vielfaches von p, so muß p entweder r oder s teilen, was nicht sein kann. Damit ist die Menge $\{1, 2, \ldots, p-1\}$ unter der Multiplikation in $\mathbb{Z}_p$ abgeschlossen. Die Existenz von Inversen zeigen wir mittels vollständiger Induktion. Offenbar gilt $1^{-1} = 1$. Nehmen wir nun an, i^{-1} existiert für alle i mit $1 \leq i < n \leq p-1$. Um n^{-1} zu konstruieren, dividieren wir zunächst p durch n in $\mathbb{Z}$ und erhalten

$$p = nq + r \quad \text{mit} \quad r < n \quad \text{und} \quad 0 \leq q < p \tag{2.0-00}$$

Da p eine Primzahl ist, folgt $r \neq 0$. Nach Induktionsvoraussetzung existiert $r^{-1} \in \{1, 2, \ldots, p-1\}$. In $\mathbb{Z}_p$ lautet Gleichung (2.0-00) wie folgt: $0 = n \cdot q + r$. Multiplikation von rechts mit r^{-1} liefert $0 = n \cdot q \cdot r^{-1} + 1$, woraus wir $n^{-1} = -q \cdot r^{-1}$ schließen. $\square$

In Kapitel 4 werden wir sehen, daß es sogar zu jeder Primzahlpotenz p^n einen Körper mit p^n Elementen gibt. Wir wenden uns jetzt den Vektorräumen zu, die wir im Rest dieses Kapitels zur Konstruktion fehlerkorrigierender Codierungen verwenden wollen.

2.0.18 Definition. Ein *Vektorraum* $\langle V, +, \mu \rangle$ über einem Körper $\langle F, +, \cdot, 1 \rangle$ besteht aus einer abelschen Gruppe $\langle V, + \rangle$ und einer sogenannten *Skalarmultiplikation* $K \times V \xrightarrow{\mu} V$ (geschrieben als Konkatenation), die folgenden Bedingungen genügt:

(V0) μ ist bi-additiv bzgl. $+$, d.h.,

- $(x+y)\vec{v} = x\vec{v} + y\vec{v}$ für alle $x, y \in F$ und alle $\vec{v} \in V$;
- $x(\vec{v} + \vec{w}) = x\vec{v} + x\vec{w}$ für $x \in F$ und alle $\vec{v}, \vec{w} \in V$.

(V1) gemischte Assoziativität, d.h.,

$$\begin{tikzcd} F \times F \times V \arrow[r, "F \times \mu"] \arrow[d, "\cdot \times V"'] & F \times V \arrow[d, "\mu"] \\ F \times V \arrow[r, "\mu"'] & V \end{tikzcd}$$

oder elementweise: $x(y\vec{v}) = (x \cdot y)\vec{v}$ für alle $x, y \in F$ und alle $\vec{v} \in V$.

(V2) $1\vec{v} = \vec{v}$ für alle $\vec{v} \in V$.

Eine *lineare Abbildung* zwischen Vektorräumen ist ein Gruppenhomomorphismus, der mit der Skalarmultiplikation verträglich ist.

2.0.19 Proposition. *Ist $V \xrightarrow{f} W$ eine lineare Abbildung von $\langle V, +, \mu \rangle$ nach $\langle W, +, \nu \rangle$, so ist sowohl das Bild* $\mathrm{img}(f) = \{ f(\vec{v}) : \vec{v} \in V \} \subseteq W$ *als auch der* Kern $\ker(f) = \{ \vec{v} \in V : f(\vec{v}) = \vec{0} \}$ *ein Unterraum von W bzw. von V.* $\square$

2.0.20 Proposition. *Ist U ein Unterraum von $\langle V, +, \mu \rangle$, so wird durch*

$$x(\vec{v} + U) := (x\vec{v}) + U$$

eine Skalarmultiplikation auf der Faktorgruppe V/U definiert, die selbige zu einem Vektorraum über F macht. Weiterhin ist die kanonische Abbildung $\vec{v} \to \vec{v} + U$ linear.

2.0.21 Definition. Wir betrachten einen Vektorraum $\langle V, +, \mu \rangle$ über einem Körper F.

(0) Eine endliche Summe der Form $\sum_{i<n} x_i\vec{v}_i$ mit $x_i \in F$ und $\vec{v}_i \in V$, $i < n$, heißt *Linearkombination* der Vektoren $\vec{v}_0, \ldots, \vec{v}_{n-1} \in V$.

(1) Eine Teilmenge $A \subseteq V$ heißt *linear unabhängig*, falls für jede Linearkombination von Elementen $v_i \in A$, $i < n \in \mathbb{N}$, gilt

$$\sum_{i<n} x_i\vec{v}_i = \vec{0} \iff \forall i < n.\, x_i = 0$$

(2) Eine maximale linear unabhängige Teilmenge von V wird als *Basis* bezeichnet, ihre Kardinalität als *Dimension* von V.

2.0.22 Satz. *Ist U ein k-dimensionaler Unterraum eines n-dimensionalen Vektorraums V, so gilt $dim(V/U) = n - k$.* □

2.0.23 Satz. (0) *Für jedes $n \in \mathbb{N}$ ist F^n ein n-dimensionaler Vektorraum über F.*

(1) *Jeder n-dimensionale Vektorraum über F ist isomorph zu F^n.* □

2.0.24 Satz. *Zu jedem k-dimensionalen Unterraum U des n-dimensionalen Vektorraums F^n existieren lineare Abbildungen $F^k \xrightarrow{g} F^n \xrightarrow{h} F^{n-k}$ mit $\operatorname{img}(g) = U = \ker(h)$.*

Beweis: Nach Satz 2.0.23(0) ist U zu F^k isomorph. Ist $F^k \xrightarrow{f} U$ ein Isomorphismus, so liefert die Verknüpfung mit der Inklusion $U \xrightarrow{i} F^n$ die gewünschte lineare Abbildung g. Andererseits hat nach Satz 2.0.22 der Faktorraum F^n/U die Dimension $n-k$, ist somit zu F^{n-k} isomorph. Die Verknüpfung der kanonischen Abbildung $\vec{v} \to [\vec{v}] = \vec{v} + U$ mit einem Isomorphismus $F^n/U \xrightarrow{j} F^{n-k}$ liefert die gewünschte lineare Abbildung h (beachte, daß U der Nullvektor von F^n/U ist). □

Die linearen Abbildungen g und h in diesem Satz sind nicht eindeutig bestimmt. Man kann beispielsweise g von links mit einem Isomorphismus auf F^k und h von rechts mir einem Isomorphismus auf F^{n-k} komponieren.

2.0.25 Bemerkung. Da im Körper $\mathbb{Z}_2$ gilt $1 + 1 = 0$, lassen sich Unterräume von Vektorräumen der Form $\mathbb{Z}_2^n$ besonders einfach charakterisieren: $U \subseteq \mathbb{Z}_2^n$ ist genau dann ein Unterraum wenn U nicht leer und unter der Addition abgeschlossen ist.

2.1 Lineare Codes

Im Folgenden bezeichnet F immer einen endlichen Körper. Anstelle beliebiger Blockcodierungen $A \xrightarrow{G} B^n$ wollen wir nun solche der Form $F^k \xrightarrow{G} F^n$ betrachten, die zudem linear sind. Von nun an wird auch der zugehörige Code, d.h., das Bild von F^k unter der Codierung, eine größere Rolle spielen als bisher.

2.1.00 Definition. Ein *linearer* (n, k)-*Code über* F ist ein k-dimensionaler Unterraum von F^n.

Nach Satz 2.0.24 gibt es zu einem linearen (n, k)-Code U über F lineare Abbildungen $F^k \xrightarrow{g} F^n \xrightarrow{h} F^{n-k}$ mit $\mathrm{img}(g) = U = \ker(h)$. Bezüglich der Standardbasen haben diese linearen Abbildungen Matrixdarstellungen, die wir mit G bzw. H bezeichnen wollen. Wir vereinbaren, weiterhin mit Zeilenvektoren zu rechnen; damit ist G eine $(k \times n)$-Matrix und H ist eine $(n \times (n - k))$-Matrix. Schließlich setzen wir $r := n - k$.

2.1.01 Definition. U sei ein linearer (n, k)-Code über F. Eine $(k \times n)$-Matrix G mit $\mathrm{img}(G) = U$ heißt *Generatormatrix* für U, und eine $(n \times r)$-Matrix H mit $U = \ker(H)$ heißt *Kontrollmatrix* für U.

Wir wenden uns zunächst den Kontrollmatrizen zu. Die Spalten einer (n, r)-Matrix H sind genau dann linear unabhängig, wenn der Lösungsraum des zugehörigen linearen Gleichungssystems (LGS) $\vec{v} \cdot H = \vec{0}$ k-dimensional ist.

2.1.02 Beispiele. (0) Das Hinzufügen eines Paritätsbits zu jedem Quellwort in F^k liefert eine $(k + 1, k)$-Codierung. Sie ist genau dann linear, wenn für die neue Komponente gilt $v_k = -\sum_{i<k} v_i$. Im Fall $F = \mathbb{Z}_2$ bedeutet dies, daß die Anzahl der Einsen unter den $k + 1$ Zeichen gerade zu sein hat. Als Kontrollmatrix können wir einen Spaltenvektor aus n Einsen verwenden.

(1) Für die Wiederholungscodierung mit $k = 1$ und $n = 5$ haben wir $|F| =: q$ Codewörter, von $(0, 0, 0, 0, 0)$ bis $(q - 1, q - 1, q - 1, q - 1, q - 1)$. Mögliche Kontrollmatrizen sind

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{und} \quad H' = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

dies entspricht den Gleichungen $x_0 + x_1 = 0$, $x_0 + x_2 = 0$, $x_0 + x_3 = 0$ sowie $x_0 + x_4 = 0$ bzw. $x_0 + x_1 = 0$, $x_1 + x_2 = 0$, $x_2 + x_3 = 0$ sowie $x_3 + x_4 = 0$. Die Lösungen müssen offenbar in beiden Fällen $x_0 = x_1 = x_2 = x_3 = x_4$ erfüllen, was sich natürlich auch durch andere lineare Gleichungen realisieren läßt.

(2) Das Bild der 2-aus-5-Codierung (vergl. Beispiel 0.1.09(2)) ist nicht linear, da 10 keine 2er-Potenz ist.

Was können wir allein aus der Kontrollmatrix H über den Code ablesen?

2.1.03 Definition. Das *Hamming-Gewicht* (oder kurz *Gewicht*) $||\vec{x}||$ eines Wortes $x \in F^n$ ist die Anzahl der von 0 verschiedenen Komponenten, d.h., $||\vec{x}|| := d_H(\vec{x}, \vec{0})$.

2.1.04 Proposition. *Der Minimalabstand eines linearen Codes ist das minimale Gewicht eines von* $\vec{0}$ *verschiedenen Codeworts.* □

2.1.05 Satz. *Ein linearer Code hat genau dann Minimalgewicht* d, *falls für jedes* $c < d$ *beliebige* c *Zeilen seiner Kontrollmatrix linear unabhängig sind und* d *linear abhängige Zeilen existieren.*

Beweis: ($\Rightarrow$). Wir betrachten c verschiedene Zeilen $\vec{h}_{i_t}$, $t < c$, der Kontrollmatrix H und eine Linearkombination $\sum_{t<c} \lambda_t \vec{h}_{i_t} = \vec{0}$. Der durch

$$x_j = \begin{cases} \lambda_t & \text{falls } j = i_t \text{ für ein } t < c \\ 0 & \text{sonst} \end{cases}$$

definierte Vektor $\vec{x}$ erfüllt nach Konstruktion $\vec{x} \cdot H = \vec{0}$, ist also ein Codewort. Wegen $||\vec{x}|| \leq c < d$ folgt nun $\vec{x} = \vec{0}$.

($\Leftarrow$). Wir nehmen an, daß jeweils $c < d'$ Zeilen von H linear unabhängig sind und d' linear abhängige Zeilen existieren. Diese induzieren ein von $\vec{0}$ verschiedenes Codewort mit Gewicht d', also gilt $d \leq d'$. Andererseits existiert ein Codewort $\vec{x}$ mit Gewicht d, wegen $\vec{x} \cdot H = \vec{0}$ werden dadurch d linear abhängige Zeilen bestimmt. Also folgt $d' \leq d$ und somit $d' = d$. □

Dieses Ergebnis setzt die einander widersprechenden Ziele von großem Minimalabstand und hoher Coderate in Beziehung. Um wenigstens einen Fehler korrigieren zu können, brauchen wir einen Minimalabstand von $d \geq 3$. In der folgenden Definition wird für $d = 3$ (SEC-Codes) die Coderate k/n maximiert.

2.1.06 Definition. Ein linearer (n,k)-Code über F heißt *Hamming-Code*, sofern n die Maximalzahl paarweise linear unabhängiger Vektoren in F^{n-k} ist.

2.1.07 Beispiele. (0) $F = \mathbb{Z}_2$: Zwei Vektoren in einem Vektorraum über $\mathbb{Z}_2$ sind genau dann linear unabhängig, wenn sie verschieden und beide von $\vec{0}$ verschieden sind. Für jedes $r = n - k \geq 1$ gibt es $n = 2^r - 1$ Vektoren, die paarweise diesen Bedingungen genügen: das sind gerade diejenigen, die den Binärdarstellungen der Zahlen von 1 bis $2^r - 1$ entsprechen. Damit ergibt sich k zu $k = n - r = 2^r - r - 1$. Speziell für $r = 3$ erhalten wir $n = 7$ und $k = 4$. Als Kontrollmatrizen eignen sich beispielsweise

$$H = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \qquad \text{oder} \qquad H' = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Wir werden später die zweite Variante bevorzugen. Offenbar ist im ersten Fall 1110000 und im zweiten Fall 0001011 ein Codewort, also gilt tatsächlich $d = 3$.

(1) $F = \mathbb{Z}_3$: Jetzt müssen wir auf skalare Vielfache achten. Wir können dieses Problem vermeiden, indem wir nur solche Zeilen betrachten, deren erste von 0 verschiedene Komponente eine 1 ist. Wiederum für $r = 3$ erhalten wir etwa

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

2.1.08 Definition. Ein Blockcode der Länge n heißt *perfekt für t Fehler*, falls die abgeschlossenen t-Kugeln um die Codewörter die Menge F^n partitionieren, d.h., wenn jedes $\vec{x} \in F^n$ zu genau einem Codewort einen Hammingabstand $\leq t$ hat.

Die binären Hamming-Codes werden sich als perfekt für 1 Fehler herausstellen. Ansonsten gibt es bis auf Wiederholungscodes ungerader Länge nur noch einen linearen binären perfekten Code: der sogenannte *Golay-Code* der Länge 23 ist perfekt für 3 Fehler.

Welche praktischen Vorteile bietet eine Kontrollmatrix der Form $\binom{A}{E_r}$, wobei A eine $(k \times r)$-Matrix und E_r eine $(r \times r)$-Einheitsmatrix ist (wie etwa H' in Beispiel 2.1.07(0) und

H in Beispiel 2.1.07(1))? Offenbar gilt für ein Codewort $\vec{x}$

$$\vec{x} \cdot \begin{pmatrix} A \\ E_r \end{pmatrix} \iff -(x_0, x_1, \ldots, x_{k-1}) \cdot A = (x_k, x_{k+1}, \ldots, x_{n-1})$$
$$\iff (x_0, x_1, \ldots, x_{k-1}) \cdot (E_k, -A) = \vec{x}$$

Also können wir $G = (E_k, -A)$ als Generatormatrix benutzen. Die Codierung erfolgt somit durch Anfügen von $r = n - k$ Kontrollbits an das Originalwort aus F^k.

2.1.09 Beispiele. (0) In Beispiel 2.1.07(0) und 2.1.07(1) können wir Generatorenmatrizen direkt von H' bzw. H ablesen.

(1) Für den Wiederholungscode mit $n = 5$ aus Beispiel 2.1.02(1) erhalten wir die Generatormatrix $G = (1, 1, 1, 1, 1)$.

(2) Für den Paritätskontrollcode aus Beispiel 2.1.02(0) ergibt sich die Generatormatrix

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

Man beachte die Ähnlichkeiten!

2.1.10 Definition. (0) Ein linearer Code $U \leq F^n$ heißt *systematisch*, falls U eine Generatormatrix der Form (E_k, B) hat.

(1) Zwei lineare Codes $U, U' \leq F^n$ heißen *äquivalent*, falls Generatormatrizen G und G' existieren, die sich nur in der Reihenfolge der Spalten unterscheiden.

2.1.11 Bemerkung. (0) Entsteht die Matrix G' aus einer Generatormatrix G für $U \leq F^n$ durch elementare Zeilenoperationen, so ist G' ebenfalls eine Generatormatrix für U (HA). Also entsprechen elementare Zeilenoperationen Isomorphismen auf F^k mit $k := \dim(U)$.

(1) Permutiert man die Spalten einer Generatormatrix G, so entspricht das einem Isomorphismus auf F^n, der Hamming-Abstände bewahrt, auch *Isometrie* genannt. Sind alle Isometrien von dieser Form?

2.1.12 Proposition. *Jeder lineare Code $U \leq F^n$ ist zu einem systematischen Code äquivalent.*

Beweis: Eine Generatormatrix G für U hat $k := \dim(U)$ linear unabhängige Spalten. Haben die ersten k Spalten diese Eigenschaft, so können wir mittels elementarer Zeilenoperationen eine systematische Generatormatrix für U erzeugen. Sind die ersten k Spalten linear abhängig, läßt sich erst nach einer Permutation der Spalten mittels elementarer Zeilenoperationen eine systematische Matrix herstellen. Die inverse Spaltenpermutation liefert dann eine Generatormatrix für U. □

2.1.13 Corollar. *Jeder lineare Code $U \leq F^n$ ist äquivalent zu einem Code mit einer Kontrollmatrix der Form $\binom{A}{E_r}$.* □

Damit haben wir auch ein Verfahren, eine Generatormatrix G zu einer beliebigen vorgegebenen Kontrollmatrix H zu finden: mittels einer geeigneten Permutation befördern wir r linear unabhängige Zeilen von H nach unten, und erzeugen dann mittels elementaren Spaltenoperationen eine neue Kontrollmatrix $H' = \binom{A}{E_r}$ desselben Codes. Auf die Spalten der zugehörigen Generatormatrix $G' = (E_k, -A)$ wenden wir nun die inverse Permutation an und erhalten die gesuchte Generatormatrix G.

Man beachte, daß in den vorangehenden Ergebnissen die Unterscheidung zwischen Codes und Codierungen wichtig ist.

2.1.14 Beispiel. Über $F = \mathbb{Z}_3$ betrachten wir die Kontrollmatrix

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

Vertauschen der Zeilen 3 und 5 liefert

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

mit zugehöriger Generatormatrix

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

In dieser sind nun die Spalten 3 und 5 zu vertauschen:

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

Aus vorgegebenen linearen Codes kann man auf verschiedene Weise neue erzeugen.

2.1.15 Definition. Für einen k-dimensionalen linearen Code $U \leq F^n$ definieren wir

(0) den zu U *dualen Code* $U^{\perp} := \{ \vec{x} \in F^n : \forall \vec{u} \in U.\, \vec{u} \cdot \vec{x}^t = 0 \} \leq F^n$;

(1) die *Erweiterung* $\bar{U} := \{ \langle \vec{x}, -\sum_{i<n} x_i \rangle : \vec{x} \in U \} \leq F^{n+1}$;

(2) die *Punktierung* $\dot{U} := \{ \vec{y} \in F^{n-1} : \exists x \in F.\, \langle \vec{y}, x \rangle \in U \} \leq F^{n-1}$ falls $n \geq 1$;

(3) die *Augmentierung* $U^{\text{aug}} :=< U \cup \{\langle 1, 1, \ldots, 1 \rangle\} >\leq F^n$.

Falls $F = \mathbb{Z}_2$ gilt, betrachten wir gelegentlich auch

(4) die *Ausdünnung* $U^{\text{even}} := \{ \vec{x} \in U : \|\vec{x}\| \text{ gerade} \}$.

2.1.16 Proposition. *Hat ein k-dimensionaler linearer Code $U \leq F^n$ eine Generatormatrix G und eine Kontrollmatrix H, so sind*

(0) H^t und G^t Genrator- bzw. Kontrollmatrizen für $U^{\perp}$;

(1)

$$\bar{G} = \left(\begin{array}{c|c} G & \begin{matrix} -\sum_{i<n} g_{0,i} \\ \vdots \\ -\sum_{i<n} g_{k-1,i} \end{matrix} \end{array} \right) \quad \text{und} \quad \bar{H} = \left(\begin{array}{c|c} \begin{matrix} \\ H \\ \\ \end{matrix} & \begin{matrix} 1 \\ \vdots \\ \vdots \end{matrix} \\ \hline 0 \;\ldots\; 0 & 1 \end{array} \right)$$

Genrator- bzw. Kontrollmatrizen für $\bar{U}$. □

2.1.17 Beispiele. (0) Paritätskontrollcodes und Wiederholungscodes sind zueinander dual, wie wir in den Beispielen 2.1.02 und 2.1.09 gesehen haben.

(1) Für den (7,4) Hamming-Code aus Beispiel 2.1.07(0) hat der duale Code die Generatormatrix

$$G^{\perp} = H'^{\mathrm{t}} = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

und die Kontrollmatrix

$$H^{\perp} = G'^{\mathrm{t}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

Aus Symmetriegründen ist der Nachweis der linearen Unabhängigkeit je dreier Zeilen dieser Matrix noch handhabbar. Weiterhin gibt es 4 linear abhängige Zeilen, etwa die ersten 4. Also hat dieser Code Minimalabstand 4.

(2) Erweitern wir den binären (7,4) Hamming-Code zu einem (8,4) Code mit $d = 4$, so ergeben sich die folgenden Matrizen:

$$\bar{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} \quad \text{sowie} \quad \bar{H} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Addiert man die ersten drei Spalten zur letzten, so ergibt sich die alternative Kontrollmatrix

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

die mit $\bar{G}^{t}$ übereinstimmt. Der zugehörige Code ist also *selbstdual.*

2.1.18 Proposition. *Für den Körper $F = \mathbb{Z}_2$ und einen linearen Code $U \leq F^n$ mit Minimalabstand d ergibt sich der Minimalabstand $\bar{d}$ der Erweiterung $\bar{U}$ zu*

$$\bar{d} = \begin{cases} d & \text{falls } d \text{ gerade} \\ d+1 & \text{falls } d \text{ ungerade} \end{cases}$$

□

Damit ist die Erweiterung binärer linearer Codes nur dann sinnvoll, wenn ein ungerader Minimalabstand vorliegt und das Verfahren kann nur einmal angewandt werden.

Für Körper F mit mehr als zwei Elementen sind zur Bestimmung des Minimalabstands der Erweiterung die Codewörter mit minimalem Gewicht zu untersuchen. Existiert eins, dessen Komponentensumme den Wert 0 hat, vergrößert sich der Minimalabstand beim Erweitern nicht.

2.1.19 Proposition. *Die Ausdünnung eines linearen binären Codes mit ungeradem Minimalabstand liefert einen linearen Code mit größerem Minimalabstand.* □

2.2 Standarddecodierung

Bisher hatten wir zur Decodierung eines gestörten Worts $\vec{z}$ die maximum likelihood Methode angewendet: man wählt unter den Codewörtern mit minimalem Hamming-Abstand zu $\vec{z}$ zufällig eins aus. Bei linearen Codes läßt sich der Auswahlschritt vorverlegen, so daß die Decodierung mechanisch und somit schneller erfolgen kann.

Wir betrachten einen k-dimensionalen Untervektorraum $U \leq F^n$. Der Faktorraum F^n/U ist nichts anderes als die entsprechende Faktorgruppe mit der induzierten Skalarmultiplikation. Also sind die Elemente von F^n/U gerade die Äquivalenzklassen der Relation $\vec{x} \sim_U \vec{y} \iff \vec{y} - \vec{x} \in U$. Es gibt $|F|^{n-k}$ solche Klassen, die alle $|F|^k$ viele Elemente enthalten; insbesondere ist F^n/U zu F^{n-k} isomorph. Genauer: jede Kontrollmatrix H des Codes U kann als surjektive lineare Abbildung $F^n \xrightarrow{H} F^{n-k}$ aufgefaßt werden, die mittels eines geeigneten Isomorphismus $F^n/U \xrightarrow{i} F^{n-k}$ durch die kanonische Abbildung $F^n \longrightarrow F^n/U$, $\vec{x} \mapsto [\vec{x}]_U = \vec{x} + U$ faktorisiert. H und i bestimmen einander eindeutig.

2.2.00 Proposition. *Für jedes Codewort $\vec{u} \in U$ und jedes $\vec{z} \in F^n$ gehören $\vec{z}$ und das potentielle Fehlerwort $\vec{e} := \vec{z} - \vec{u}$ zur selben Äquivalenzklasse und folglich gilt $\vec{z} \cdot H = \vec{e} \cdot H$.*

Beweis: $\vec{e} - \vec{z} = \vec{u} \in U$ □

Betrachtet man die Zeilen der Kontrollmatrix H als *Symptome* für Fehler an der entsprechenden Stelle des empfangenen Wortes, motiviert das die folgende Begriffsbildung:

2.2.01 Definition. Für $\vec{z} \in F^n$ wird der Vektor $\vec{z} \cdot H \in F^{n-k}$ als das *Syndrom* (= Summe aller Symptome) von $\vec{z}$ bzgl. H bezeichnet.

Die Vorverlegung der Auswahl in der MLD geschieht nun dadurch, daß in jeder Äquivalenzklasse vorab ein Element minimalen Gewichts als wahrscheinliches Fehlerwort ausgewählt wird. Existieren in einer Klasse mehrere Wörter gleichen minimalen Gewichts, so wird beim Auftreten der nicht gewählten Wörter als tatsächliche Fehler immer falsch decodiert. Aber dies ist im Durchschnitt nicht schlechter (aber auch nicht besser) als wenn bei jedem empfangenen Wort unter den möglichen minimalen Fehlern gewählt werden muß. Der Vorteil ist allein technischer Natur: gewählt wird in jeder Äquivalenzklasse einmal, alles Übrige kann mechanisch erfolgen.

In der sogenannten "Standardtabelle" eines Codes $U \leq F^n$ sind zeilenweise die Äquivalenzklassen aufgelistet, beginnend mit U. Jede Zeile beginnt mit einem Wort minimalen Gewichts, dem sogenannten "Anführer" oder "coset leaders", die übrigen Einträge ergeben sich durch Addition des ersten Wortes der Zeile zum ersten Wort (einem Codewort) der entsprechenden Spalte.

2.2.02 Beispiel. Wir betrachten den $(4,2)$-Code über $\mathbb{Z}_2$ mit folgender Generator- und Kontrollmatrix

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

Das liefert beispielsweise die folgende Standardtabelle

Nachricht	00	10	01	11	Syndrom
Codewörter	0000	1011	0101	1110	00
Äquivalenzklasse	1000	0011	1101	0110	11
Äquivalenzklasse	0100	1111	0001	1010	01
Äquivalenzklasse	0010	1001	0111	1100	10

Wird z.B. das Wort $\vec{z} = 1111$ empfangen, so bestimmt diese Tabelle dazu das Fehlerwort $\vec{e} = 0100$, was zum korrigierten Wort $\vec{u} = \vec{z} - \vec{e} = 1011$ führt. Eine andere Tabelle ist möglich, da wir als Anfang der dritten Zeile auch das Wort 0001 hätten wählen können. Dies hätte eine Korrektur der letzten Stelle des empfangenen Wortes zur Folge gehabt.

2.2.03 Proposition. *Die Standard-Decodierung ist genau dann korrekt, wenn der tatsächliche Fehler ein Äquivalenzklassenanführer ist.*

Beweis: Wird das Codewort $\vec{u} \in U$ gesendet, aber $\vec{z} \in F^n$ empfangen, so gilt für den Anführer $\vec{y}$ der Klasse $[\vec{z}]_U = \vec{z} + U$

$$\vec{e} = \vec{z} - \vec{u} = \vec{y} + (\vec{z} - \vec{y}) - \vec{u} \in \vec{y} + U$$

woraus die Äquivalenz von $\vec{y} = \vec{e}$ und $\vec{z} - \vec{y} = \vec{u}$ folgt. □

2.2.04 Beispiel. K_5 sei der binäre lineare Code der Länge 5, in dem das vorletzte Bit die Parität der ersten beiden Bits prüft und das letzte Bit eine Gesamtparitätskontrolle durchführt. Die Codewörter erfüllen daher das folgende LGS

$$x_3 = x_0 + x_1$$
$$x_4 = x_0 + x_1 + x_2 + x_3$$

Dies liefert die folgende Kontrollmatrizen

$$H = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 1 \\ 1 & 1 \\ 0 & 1 \end{pmatrix} \qquad \text{bzw. nach Addition der ersten zur letzten Spalte} \qquad H' = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

woraus sich die Generatormatrix

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

ergibt. Eine mögliche Standardtabelle ist die folgende:

000	001	010	011	100	101	110	111	
00000	00101	01010	01111	10010	10111	11000	11101	00
10000	10101	11010	11111	00010	00111	01000	01101	10
00100	00001	01110	01011	10110	10011	11100	11001	01
10001	10100	11011	11110	00011	00110	01001	01100	11

Obwohl wir aus der Kontrollmatrix den Minimalabstand $d = 2$ ablesen, der Code also nur Einzelfehler erkennt aber nicht korrigiert, werden sogar Doppelfehler in der ersten und letzten Stelle erfolgreich korrigiert.

In der Praxis benötigen wir also eine Liste von Syndromen und den aus der entsprechenden Äquivalenzklasse ausgewählten Anführer. Dies erfordert relativ wenig Platz.

Für den $(2^r - 1, 2^r - 1 - k)$-Hamming Code liefert die Kontrollmatrix, deren Zeilen aufsteigend die Binärdarstellungen der Zahlen von 1 bis $2^r - 1$ sind Syndrome, die direkt das zu korrigierende bit angeben. Der für soviel Bequemlichkeit zu zahlende Preis ist allerdings, daß für $n > 3$ der Code nicht mehr systematisch ist.

2.2.05 Proposition. *Ein linearer Code ist genau dann perfekt für t Fehler, wenn genau die Wörter mit Hamminggewicht $\leq t$ als Äquivalenzklassenanführer vorkommen.*

Beweis: ($\Rightarrow$). Wenn die abgeschlossenen t-Kugeln um die Codewörter F^n partitionieren, dann gehört jeder Anführer, etwa $\vec{y}$, in einer dieser Kugeln, etwa um $\vec{u}$. Aber die Differenz $\vec{y} - \vec{u}$ gehört zur selben Äquivalenzklasse, liegt in der t-Kugel um $\vec{0}$ und hat somit Gewicht $\leq t$. Daraus folgt $\vec{u} = \vec{0}$. Also liegen alle Anführer in der t-Kugel um $\vec{0}$. Andererseits ist für zwei verschiedene Elemente $\vec{x}, \vec{y}$ der t-Kugel um $\vec{0}$ das Hamminggewicht von $\vec{y} - \vec{x}$ kleiner als $2t$ und somit kein Codewort. Also sind $\vec{x}$ und $\vec{y}$ nicht äquivalent.

($\Leftarrow$). Die Anführer konstituieren die t-Kugel um $\vec{0}$. Entsprechend stehen in den Spalten der Standardtabelle genau die Elemente der t-Kugeln um die Codewörter in der ersten Zeile. Da die Standardtabelle alle Elemente von F^n genau einmal enthält, folgt die Behauptung. □

2.2.06 Beispiele. (0) Binäre Hamming-Codes sind perfekt für $t = 1$: Ein (n, k)-Code mit $n = 2^r - 1$ und $k = 2^r - 1 - r$ hat $2^n - k = 2^r = n + 1$ Äquivalenzklassen. Die Standardeinheitsvektoren liegen in verschiedenen Klassen, denn sie erzeugen verschiedene Syndrome (= Zeilen der Kontrollmatrix). Zusammen mit $\vec{0}$ haben wir damit alle Wörter vom Gewicht ≤ 1 als Anführer.

(1) Binäre Wiederholungscodes der Länge $2t+1$ sind perfekt für t Fehler: Wegen $U = \{\vec{0}, \vec{1}\}$ bestehen alle Äquivalenzklassen aus zwei Elementen: $[\vec{x}]_U = \vec{x} + U = \{\vec{x}, \vec{x} + \vec{1}\}$. Diese haben wegen der ungeraden Länge verschiedene Hamminggewichte, die sich aber zu $2t+1$ summieren. Daher können wir alle Wörter mit Gewicht $\leq t$ als Anführer wählen.

Es gibt einen linearen binären $(23, 12)$-Code, der perfekt für $t = 3$ ist, den nach seinem Erfinder benannten *Golay-Code*.

2.2.07 Satz. (Tietvävainen, VanLint, 1974) *Jeder perfekte binäre lineare Code ist äquivalent zu einem Hamming Code, oder einem Wiederholungscode ungerader Länge oder dem Golay-Code.* □

2.3 Reed-Muller Codes

In diesem Kapitel betrachten wir nur den Körper $F = \mathbb{Z}_2$. Für $n = 2^m$ suchen wir "interessante" Untervektorräume von $(\mathbb{Z}_2)^n$, die eine Decodierung ohne Verwendung von Syndromen erlauben.

Identifiziert man die *Zahl* 2 mit der *Menge* $\{0,1\}$, und erinnert man sich daran daß die Menge aller Funktionen von A nach B auch mit B^A bezeichnet wird, so kann man die Elemente von $(\mathbb{Z}_2)^{2^m}$ als Funktionen von $\{0,1\}^m$ nach $\{0,1\}$ auffassen, als sogenannte "m-stellige boolesche Funktionen". Da auf den m-Tupeln in $\{0,1\}^m$ eine natürliche Ordnung definiert ist, man interpretiert sie als Binärdarstellungen der Zahlen von 0 bis $2^m - 1$, kann man jede Funktion $\{0,1\}^m \xrightarrow{f} \{0,1\}$ mit dem 2^m-Tupel ihrer Funktionswerte

$$\langle f(0\dots00), f(0,\dots01), \dots, f(1\dots11)\rangle$$

identifizieren.

Von besonderem Interesse sind die konstanten Funktionen $\boldsymbol{0}$ und $\boldsymbol{1}$, sowie die Projektionen x_j, $j < m$, die j-te Koordinate.

2.3.00 Beispiel. Für $m = 4$ erhalten wir folgende Tupeldarstellungen der vier Projektionen:

$$\begin{aligned} x_0 &: 0101010101010101 \\ x_1 &: 0011001100110011 \\ x_2 &: 0000111100001111 \\ x_3 &: 0000000011111111 \end{aligned}$$

Man erkennt ein einfaches Muster: x_j, als Tupel aufgefaßt, besteht aus alternierenden Blöcken von 2^j Nullen und 2^j Einsen. Die Spalten reproduzieren die Argumente, nämlich die Zahlen von 0 bis $2^4 - 1 = 15$ in Binärdarstellung.

Auf den Funktionen von $\{0,1\}^m$ nach $\{0,1\}$ sind komponentenweise Addition und Multiplikation erklärt. Da wir über dem $\mathbb{Z}_2$ arbeiten ist die Addition selbst-invers, d.h., $f + f = \boldsymbol{0}$, und die Multiplikation ist idempotent, d.h., $ff = f$. Es stellt sich die Frage, welche m-stelligen booleschen Funktionen mittels dieser Operationen aus den Konstanten und den Projektionen konstruiert werden können. Erfreulicherweise heißt die Antwort "Alle". Um dies einzusehen, benötigen wir den folgenden Satz:

2.3.01 Satz. *Ist ≥ 2, so gilt für jede m-stellige boolesche Funktion f*

$$\begin{aligned} f &= (f \circ (x_0, x_1, \dots, x_{m-1}))\,\boldsymbol{1} \\ &= (f \circ (x_0, x_1, \dots, x_{m-2}, \boldsymbol{0}))\,\boldsymbol{1} \\ &\quad + (f \circ (x_0, x_1, \dots, x_{m-2}, \boldsymbol{0}) + f \circ (x_0, x_1, \dots, x_{m-2}, \boldsymbol{1}))\,x_{m-1} \end{aligned}$$

Beweis: Anwendung beider Funktionen auf ein m-Tupel $\langle z_0, z_1, \ldots, z_{m-1}\rangle$ liefert

$$\begin{aligned} f(z_0, z_1, \ldots, z_{m-1}) &= f(z_0, z_1, \ldots, z_{m-2}, 0) \\ &\quad + \big(f(z_0, z_1, \ldots, z_{m-2}, 0) + f(z_0, z_1, \ldots, z_{m-2}, 1)\big) z_{m-1} \end{aligned}$$

Für die beiden Fälle $z_{m-1} = 0$ und $z_{m-1} = 1$ ist dies trivialerweise korrekt. □

Um diesen Satz anwenden zu können, bedienen wir uns eines Tricks: die Tupeldarstellungen der m-stelligen Funktionen $f \circ (\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{m-2}, \mathit{0})$ und $f \circ (\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{m-2}, \mathit{1})$ sind schon durch die ersten bzw. letzten 2^{m-1} Komponenten vollständig bestimmt. Verwendet man Exponenten zur Konkatenation von Tupeln, so ergibt sich

$$\begin{aligned} f &: \; f_0 f_1 \ldots f_{2^m-1} \\ f \circ (\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{m-2}, \mathit{0}) &: \; (f_0 f_1 \ldots f_{2^{m-1}-1})^2 \\ f \circ (\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{m-2}, \mathit{1}) &: \; (f_{2^{m-1}} f_{2^{m-1}+1} \ldots f_{2^m-1})^2 \end{aligned}$$

Weiterhin gilt $0^{2^m} = \mathit{0}$ und $1^{2^m} = \mathit{1}$.

2.3.02 Beispiel. Wie kann man $f = 10001100$ mittels der Projektionen und der Konstanten $\mathit{1}$ konstruieren? Nach dem oben beschriebenen Rezept erhalten wir

$$\begin{aligned} 10001100 &= 1000^2\,\mathit{1} + [1000^2 + 1100^2]\boldsymbol{x}_2 = 1000^2\,\mathit{1} + 0100^2\,\boldsymbol{x}_2 \\ &= 10^4\,\mathit{1} + [10^4 + 00^4]\boldsymbol{x}_1 + (01^4\,\mathit{1} + [01^4 + 00^4]\boldsymbol{x}_1)\boldsymbol{x}_2 \\ &= 10^4\,\mathit{1} + 10^4\boldsymbol{x}_1 + 01^4\boldsymbol{x}_2 + 01^4\boldsymbol{x}_1\boldsymbol{x}_2 \\ &= \mathit{1} + [\mathit{1} + \mathit{0}]\boldsymbol{x}_0 + (\mathit{1} + [\mathit{1} + \mathit{0}]\boldsymbol{x}_0)\boldsymbol{x}_1 + (\mathit{0} + [\mathit{0} + \mathit{1}]\boldsymbol{x}_0)\boldsymbol{x}_2 + (\mathit{0} + [\mathit{0} + \mathit{1}]\boldsymbol{x}_0)\boldsymbol{x}_1\boldsymbol{x}_2 \\ &= \mathit{1} + \boldsymbol{x}_0 + \boldsymbol{x}_1 + \boldsymbol{x}_0\boldsymbol{x}_1 + \boldsymbol{x}_0\boldsymbol{x}_2 + \boldsymbol{x}_0\boldsymbol{x}_1\boldsymbol{x}_2 \end{aligned}$$

2.3.03 Satz. *Die (komponentenweisen) Produkte der Projektionen $\boldsymbol{x}_j$, $j < m$ einschließlich des leeren Produkts $\mathit{1}$ bilden einen Basis für den Vektorraum $(\mathbb{Z}_2)^{2^m}$.*

Beweis: Daß diese Produkte den Raum aufspannen, haben wir gerade gesehen. Um ihre lineare Unabhängigkeit zu zeigen genügt der Nachweis, daß es $n = 2^m$ verschiedene solche Produkte gibt. Aber die Anzahl der Produkte mit $k \leq m$ verschiedenen Faktoren ist $\binom{n}{k}$ und Summieren über k liefert 2^m. □

Man könnte diesen Satz auch anders formulieren: "Alle booleschen Funktionen sind schon boolesche Polynome." Da wir aber im nächsten Kapitel den Begriff des Polynoms in einem anderen Zusammenhang gebrauchen werden, wollen wir ihn hier nicht weiter benutzen.

2.3.04 Definition. Der *Grad* einer von $\mathit{0}$ verschiedenen booleschen Funktion f ist die Maximalzahl $\operatorname{grd} f$ verschiedener Faktoren $\boldsymbol{x}_j$, $j < m$, die in einem Basisvektor der oben beschriebenen Darstellung von f vorkommen. Weiter setzen wir $\operatorname{grd} 0 = -1$. Unter dem *Reed-Muller Code* der Länge $n = 2^m$ verstehen wir den Untervektorraum $\mathcal{R}(r, m) \leq \{0,1\}^{2^m}$ aller booleschen Funktionen mit Grad $\leq r$.

2.3.05 Beispiele. (0) $\mathcal{R}(0,m)=\{\vec{0},\vec{1}\}$ ist der Wiederholungscode der Länge 2^m.

(1) $\mathcal{R}(1,m)$ besteht aus allen Linearkombinationen der Projektionen x_j, $j<m$, und der Konstanten $\mathit{1}$. Also gibt es 2^{m+1} Codewörter und $\mathcal{R}(1,m)$ ist ein $(2^m, m+1)$-Code. Im Jahre 1969 wurde der $(32,6)$-Code $\mathcal{R}(1,5)$ bei der Transmission von Mondfotos durch die Raumsonde Mariner verwendet. Dieser Code kann 7 Fehler korrigieren.

Kennen wir eine Basis eines linearen Codes, so erhalten wir eine Generatormatrix indem wir die Basisvektoren als Zeilen verwenden. Im Falle des Codes $\mathcal{R}(1,m)$ wollen wir der Reihe nach die Projektionen x_j , $j<m$ und dann die $\mathit{1}$ als Zeilen benutzen.

2.3.06 Beispiel. Eine Generatormatrix für $\mathcal{R}(1,3)$ ist gegeben durch

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

Addiert man die ersten drei Zeilen zur letzten, erhält man

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

Bis auf die Vertauschung zweier Spalten ist dies die Generatormatrix des erweiterten Hammingcodes aus Beispiel 2.1.17(2).

Wie decodiert man Reed-Muller Codes? Es gibt zwei Möglichkeiten, auch ohne Syndrome zu decodieren, davon stellen wir die einfachere vor. Empfangen wir nach Codierung der Nachricht $\vec{a}\in(\mathbb{Z}_2)^{m+1}$ das Wort $\vec{y}\in(\mathbb{Z}_2)^{2^m}$, so wollen wir es mit den Mustern vergleichen, die potentielle Summanden aufweisen. Ist kein Fehler aufgetreten, muß der Koeffizient a_0 wegen $\vec{y}=\sum_{j<m}a_jx_j+a_m\mathit{1}$ die Folgenden 2^{m-1} Gleichungen erfüllen:

$$a_0=y_0+y_1=y_2+y_3=\ldots=y_{n-2}+y_{n-1}$$

denn die zu addierenden Komponenten der Projektion x_0 sind immer verschieden, und in den übrigen Basisvektoren sind sie immer gleich. Ist $\vec{y}$ gestört, so treffen wir eine Majoritätsentscheidung. Es dürfen also maximal $2^{m-2}-1$ Fehler auftreten, bevor diese Verfahren versagt. Entsprechende Gleichungen lassen sich auch für die übrigen Projektionen aufstellen, etwa

$$a_1=y_0+y_2=y_1+y_3=y_4+y_6=y_5+y_7=\ldots=y_{n-3}+y_{n-1}$$
$$a_2=y_0+y_4=y_1+y_5=y_2+y_6=y_3+y_7=\ldots=y_{n-5}+y_{n-1}$$

Das Aufstellen einer allgemeinen Formel ist etwas mühsam, aber das Prinzip dürfte klar sein. Die Summanden, die sich zu a_i addieren, müssen 2^i Positionen voneinander entfernt sein, und jede Komponente kommt genau einmal vor. [Formel?] Schließlich ist a_m aus der Majorität der Symbole in

$$\vec{y} - \sum_{j<m} a_i x_j$$

zu bestimmen.

2.3.07 Beispiel. Für $m = 3$ betrachten wir $\vec{y} = 10101101$.

$$\begin{array}{llll} y_0 + y_1 = y_2 + y_3 = y_6 + y_7 = 1 & \text{und} \quad y_4 + y_5 = 0 & \text{impliziert} & a_0 = 1 \\ y_0 + y_2 = y_1 + y_3 = y_5 + y_7 = 0 & \text{und} \quad y_4 + y_6 = 1 & \text{impliziert} & a_1 = 0 \\ y_1 + y_5 = y_2 + y_6 = y_3 + y_7 = 1 & \text{und} \quad y_0 + y_4 = 0 & \text{impliziert} & a_2 = 1 \end{array}$$

Weiterhin gilt $\vec{y} - x_0 - x_2 = 11110111$, woraus $a_3 = 1$ folgt. Das ursprünglich codierte Wort war also $\vec{a} = 1011$.

2.3.08 Satz. *$\mathcal{R}(1, m)$ korrigiert maximal $2^{m-2} - 1$ Fehler.*

Beweis: Wie das Verfahren zeigt, werden $2^{m-2} - 1$ Fehler korrigiert. Verfälscht man alle 2^{m-2} Komponenten mit durch 4 teilbarem Index, so entsteht bei der Bestimmung von a_0 eine Patt-Situation: jeweils die Hälfte der Summen haben den Wert 0 bzw. 1 . Wir müssen also einen dieser Werte zufällig wählen, was zu einem Fehler führen kann. □

2.3.09 Corollar. *Der Minimalabstand von $\mathcal{R}(1, m)$ beträgt $d = 2^{m-1}$*

Beweis: Da maximal $2^{m-2} - 1$ Fehler korrigiert werden, gilt nach Satz 1.2.06(2)

$$2\left(2^{m-2} - 1\right) = 2^{m-1} - 2 < d \leq 2^{m-1}$$

Da aber alle Projektionen x_j , $j < m$, sowie $\mathbf{1}$ gerades Hamming-Gewicht haben, gilt dies auch für alle Linearkombinationen. Also folgt $d = 2^{m-1}$. □

3 Zyklische Codes

3.0 Zyklische Codes und Polynome

Zyklische Codes stellen eine Unterklasse der linearen Codes dar, die besonders angenehme Eigenschaften haben. Wir werden einige der schon bekannten Codes als zyklisch wiedererkennen.

3.0.00 Definition. Ein linearer Code $U \leq F^n$ heißt *zyklisch*, falls mit jedem Codewort $\vec{u} = u_0 u_1 \ldots u_{n-1}$ auch die zyklische Vertauschung $u_{n-1} u_0 u_1 \ldots u_{n-2}$ zu U gehört.

Um die neue Operation auf Codeworten, die der zyklischen Vertauschung der Komponenten entspricht, besser in den Griff zu bekommen, wählen wir eine andere *Darstellung* der Elemente von F^n : statt als Vektoren oder Tupel $\vec{u} = u_0 u_1 \ldots u_{n-1}$ schreiben wir sie als "formale Polynome" $u(x) = \sum_{i<n} u_i x^i$. Der Einfachheit halber lassen wir in Zukunft den Zusatz "formal" weg, vergl. Bemerkung 3.0.01(2).

3.0.01 Bemerkungen. (0) In der Polynomschreibweise ist x ein neues Symbol, dessen "Potenzen" nur dazu dienen, als Marken die verschiedenen Komponenten von $\vec{u}$ auseinanderzuhalten. Ein ähnliches Phänomen kennen wir von den complexen Zahlen. Man kann sie entweder als Paare $\langle a, b \rangle \in \mathbb{R} \times \mathbb{R}$, d.h., als Punkte der euklidischen Ebene auffassen, oder in der Form $a + bi$ schreiben, wobei der Faktor $i^0 = 1$ unsichtbar bleibt.

(1) Bezüglich der Standardeinheitsvektoren $\vec{e}_i$, $i < n$, die genau an der Position i eine Eins und sonst Nullen enthalten, können wir $\vec{u}$ folgendermaßen darstellen: $\vec{u} = \sum_{i<n} u_i \vec{e}_i$. Somit lassen sich die Symbole x^i , $i < n$, als neue *Namen* für die Standardeinheitsvektoren interpretieren. Während ein Ausdruck der Form $\sum_{i<n} u_i x^i$ unzweideutig zu interpretieren ist, führt die übliche Praxis, bei ausgeschriebenen Summen Summanden mit dem Koeffizienten 0 fortzulassen, zu Mehrdeutigkeiten: $3 + x^3 - 2x^5$ ergibt für jedes $n > 5$ im F^n Sinn. Demzufolge ist der Wert von n entweder explizit anzugeben, oder er muß aus dem Zusammenhang ersichtlich sein. Die Marke x^0 wird übrigens häufig weggelassen.

(2) Über endlichen Körpern dürfen wir (formale) Polynome nicht mit Funktionen von F nach F identifizieren, wie wir das von den reellen Zahlen her gewohnt sind. Beispielsweise sind die Wörter 110 und 101 über $\mathbb{Z}_2$ verschieden, aber die von $1+x$ und $1+x^2$ induzierten Funktionen $\mathbb{Z}_2 \longrightarrow \mathbb{Z}_2$ stimmen überein: beide vertauschen die Elemente 0 und 1 .

(3) Von den reellen Zahlen her sind wir es gewohnt, Polynome miteinander zu multiplizieren, und auch hier wird diese Operation nützlich sein: die Multiplikation mit x läßt die Komponenten "nach rechts" wandern. Technische Probleme ergeben sich allerdings, falls die Komponente u_{n-1} von Null verschieden ist. Soll sie nach der Multiplikation mit x verschwinden, oder als Koeffizient von x^n wieder auftreten?

Die ersten angedeuteten technischen Probleme lassen sich mit einem typisch mathematischen Trick in den Griff bekommen. Statt vieler verschiedener endlich-dimensionaler Vektorräume F^n , $n \in \mathbb{N}$, führen wir einen unendlich-dimensionalen Vektorraum ein, der alle endlich-dimensionalen Räume umfaßt und die unbeschränkte Multiplikation von Polynomen erlaubt. Dazu überlegt man sich leicht, daß für *jede* Menge X die Menge F^X aller Funktionen $X \longrightarrow F$ unter den komponentenweisen Operationen der Addition und Skalarmultiplikation zu einem Vektorraum wird – falls X eine endliche Menge ist, etwa $X = n = \{0, 1, \ldots, n-1\}$, sind Funktionen von n nach F nichts anderes als n-Tupel.

3.0.02 Definition. (0) Mit $F[x]$ bezeichnen wir den Untervektorraum von $F^{\mathbb{N}}$, dessen Elemente nur an endlich vielen Stellen einen von Null verschiedenen Wert annehmen. Die Elemente von $F[x]$ heißen *(formale) Polynome* über F .

(1) Unter dem *Leitkoeffizienten* eines von 0 verschiedenen Polynoms verstehen wir die letzte von Null verschiedene Komponente; der *Grad* eines solchen Polynoms ist die Position des Leitkoeffizienten. Hat der Leitkoeffizient den Wert 1, so heißt das Polynom *normiert.* Dem Nullpolynom wird der Grad $-\infty$ zugeordnet.

(2) Auf $F[x]$ ist eine Shift-Operation definiert durch $v_0v_1v_2 \ldots \mapsto 0v_0v_1v_2 \ldots$.

In der Polynomschreibweise entspricht die Shift-Operation offenbar der Multiplikation mit x , und ihre i-fache Wiederholung der Multiplikation mit x^i . Das Produkt zweier Polynome $u(x) = \sum_{i \in \mathbb{N}} u_i x^i$ und $v(x)$ kann auch als die Linearkombination $\sum_{i \in \mathbb{N}} u_i x^i v(x)$ der Verschiebungen aufgefaßt werden. Offenbar ist dieses Produkt kommutativ.

3.0.03 Satz. *Die Menge $F[x]$ mit der Vektorraumaddition und der Polynommultiplikation ist ein kommutativer Ring mit $\vec{e}_0 = 100\ldots0 = x^0$ als Einselement.* □

Natürlich könne wir alle Vektorräume F^n , $n \in \mathbb{N}$, als Untervektorräume von $F[x]$ wiederfinden: es gilt

$$F^n = \{\, f(x) \in F[x] : \operatorname{grd} f(x) < n \,\}$$

Genau besehen ist die *Standardübersetzung* $\vec{u} = u_0u_1 \ldots u_{n-1} \mapsto u(x) = \sum_{i<n} u_i x^i$ allerdings nur ein Isomorphismus von F^n auf den Untervektorraum der Polynome vom Grad $< n$. Es gibt auch noch andere Isomorphismen, eine Tatsache, die wir später ausnutzen werden. Als Untervektorräume von $F[x]$ betrachtet sind die Räume F^n , $n \in \mathbb{N}$ unter der Polynommultiplikation nicht abgeschlossen, der Grad der Produkte kann zu groß werden. Andererseits sind wir aber aus Kapitel 2.0 mit der Tatsache vertraut, daß man jedes Element s eines kommutativen Rings R ein sogenanntes Hauptideal der Form $s \cdot R = \{\, s \cdot r : r \in R \,\}$ erzeugt, nach dem wir den ursprünglichen Ring faktorisieren können. Auf diese Weise hatten wir die Ringe $\mathbb{Z}_n = \mathbb{Z}/n \cdot \mathbb{Z}$ erhalten, deren Elemente mit den möglichen Resten bei der Division durch n zu identifizieren waren. Ganz analog erzeugt jedes Polynom $f(x) \in F[x]$ einen Faktorring

$F[x]_{f(x)} := F[x]/f(x)\cdot F[x]$, dessen Elemente wir mit den möglichen Resten bei der Polynomdivision (euklidischer Algorithmus!) durch $f(x)$ identifizieren wollen. Hat $f(x)$ den Grad n , so sind die möglichen Reste genau alle Polynome vom Grad $< n$, d.h., als *Menge* stimmt $F[x]_{f(x)}$ mit dem Untervektorraum $F^n < F[x]$ überein! Jedes Polynom vom Grad n induziert eine solche Faktorring-Struktur auf F^n . Wie wir später sehen werden, müssen die von verschiedenen Polynomen gleichen Grades induzierten Ring-Strukturen nicht übereinstimmen. In der Addition können sie sich nicht unterscheiden, dafür aber in der Multiplikation.

Welche Ring-Struktur auf F^n erlaubt uns eine einfache Beschreibung der zyklischen Vertauschung der Komponenten eines n-Tupels? Dazu vergleichen wir die von den Vektoren $v_{n-1}v_0v_1\ldots v_{n-2}$ und von $v_0v_1\ldots v_{n-2}v_{n-1}$ induzierten Polynome $w(x)$ und $v(x)$:

$$w(x) = xv(x) - v_{n-1}x^n + v_{n-1} = xv(x) - v_{n-1}(x^n - 1)$$

Folglich entspricht in $F[x]_{x^n-1}$ die Multiplikation mit x genau der gewünschten zyklischen Vertauschung der Komponenten. Wir wollen also die Ring-Struktur betrachten, deren Multiplikation durch die Multiplikation modulo $x^n - 1$ in $F[x]$ gegeben ist.

3.0.04 Lemma. *Jeder zyklische Code $U \leq F^n$ ist ein Ideal von $F[x]_{x^n-1}$.*

Beweis: Für $\vec{u} \in U$ und $f(x) = \sum_{i<n} f_i x^i$ gilt $f(x)u(x) = \sum_{i<n} f_i x^i u(x)$ in $F[x]_{x^n-1}$. Da nach Voraussetzung jeder der Summanden ein Codewort ist, gilt dies auch für die Summe. □

3.0.05 Satz. *Zu jedem zyklischen (n,k)-Code $U \leq F^n$ gibt es ein Polynom $g(x) \in U$ mit $\operatorname{grd} g(x) = n - k$ und $U = g(x)F[x]_{x^n-1}$. Insbesondere ist U damit ein Hauptideal von $F[x]_{x^n-1}$.*

Beweis: Wir wählen für $g(x)$ eine Codewort mit kleinstem Grad, etwa $\operatorname{grd} g(x) = s$. Nun gilt es nachzuweisen, daß $v(x) \in F^n$ genau dann zu U gehört, wenn $v(x)$ ein Vielfaches von $g(x)$ ist. Eine Richtung folgt aus Lemma 3.0.04. Für die Umkehrung dividieren wir $v(x) \in U$ nach dem euklidischen Algorithmus durch $g(x)$ und erhalten $v(x) = q(x)g(x) + r(x)$ mit $\operatorname{grd} r(x) < s$. Da sowohl $v(x)$ als auch $q(x)g(x)$ zu U gehören, folgt dies auch für $r(x)$. Die Minimalität des Grades von $g(x)$ impliziert also $r(x) = 0$. Folglich gilt $U = \{\, q(x)g(x) : \operatorname{grd} q(x) < n - s \,\}$. Damit stimmt die Dimension k von U mit der Anzahl $n - s$ der frei wählbaren Koeffizienten der Multiplikatoren $q(x)$ überein, und es gilt $s = n - k$. □

3.0.06 Definition. Ist $U \leq F^n$ ein zyklischer Code, so heißt ein von 0 verschiedenes Polynom $g(x) \in U$ minimalen Grades ein *Generatorpolynom* von U .

3.0.07 Lemma. *Jeder von $\{0\}$ verschiedene zyklische Code hat genau ein normiertes Generatorpolynom.*

Beweis: Da zwei Generatorpolynome einander gegenseitig teilen, können sie sich nur durch einen konstanten Faktor unterscheiden. Division durch den Leitkoeffizienten liefert also dasselbe normierte Polynom. □

3.0.08 Beispiele. (0) Da die Polynome $g(x)$, $xg(x)$, …, $x^{n-k}g(x)$ eine Basis von U bilden, erhalten wir eine Generatormatrix der folgenden charakteristischen Form einer *Bandmatrix*:

$$G = \begin{pmatrix} g(x) \\ xg(x) \\ \vdots \\ x^{n-k}g(x) \end{pmatrix} = \begin{pmatrix} g_0 & \dots & g_{n-k} & 0 & \dots & \dots & 0 \\ 0 & g_0 & \dots & g_{n-k} & 0 & \dots & 0 \\ \vdots & & & & & & \vdots \\ 0 & \dots & \dots & 0 & g_0 & \dots & g_{n-k} \end{pmatrix}$$

(1) Paritätskontrolle: Über jedem Körper ist $x-1$ das gradniedrigste normierte Polynom gerader Parität. Also erhalten wir etwa für $n=7$ die Generatormatrix

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

(1) Wiederholungscode für $n=7$: Über jedem Körper liefert das Generatorpolynom $g(x) = \sum_{i<7} x^i$ die Matrix $(1,1,1,1,1,1,1)$.

(2) Wählen wir ein beliebiges Polynom über $\mathbb{Z}_2$ vom Grad <7, etwa $1+x+x^3$, so erhalten wir die Generatormatrix

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

Aber ist der entsprechende Code wirklich zyklisch? Verschieben der letzten Zeile liefert 1000110 , ist dies wirklich ein Codewort? In diesem Fall ja, es ist die Summe der ersten drei Zeilen.

Das letzte Beispiel wirft die Frage auf, welche Polynome als Generatorpolynome eines zyklischen Codes in Frage kommen. Der folgende Satz liefert die Antwort.

3.0.09 Satz. *Die (normierten) Generatorpolynome zyklischer Codes $U \leq F^n$ sind genau die (normierten) Teiler von $x^n - 1$.*

Beweis: ($\Rightarrow$). Wir betrachten ein normiertes Generatorpolynom $g(x)$ vom Grad $n-k=r$. Die Polynome $g(x)$, $xg(x)$, ..., $x^{k-1}g(x)$ sind Codewörter, wie auch $w(x) = x^k g(x) - x^n + 1$. Weiterhin gelte $x^n - 1 = q(x)g(x) + r(x)$ mit $\operatorname{grd} r(x) < \operatorname{grd} g(x)$. Auflösen nach $r(x)$ liefert

$$\begin{aligned} r(x) &= -q(x)g(x) + x^n - 1 \\ &= -q(x)g(x) - w(x) + x^k g(x) \\ &= \left(x^k - q(x)\right) g(x) - w(x) \end{aligned}$$

Andererseits ist mit $g(x)$ auch $q(x)$ normiert und hat Grad k, daher folgt $\operatorname{grd}\left(x^k - q(x)\right) < k$. Somit ist $\left(x^k - q(x)\right) g(x)$ ein Codewort. Als Summe zweier Codewörter gehört auch $r(x)$ zu U, und wegen $\operatorname{grd} r(x) < \operatorname{grd} g(x)$ folgt $r(x) = 0$.

($\Leftarrow$). $g(x)$ sei ein echter normierter Teiler von $x^n - 1$ vom Grad $n-k$, etwa $x^n - 1 = q(x)g(x)$. Wir setzen $U = \{\, p(x)g(x) : \operatorname{grd} p(x) < k \,\}$. Die Linearität von U ist offensichtlich und ebenso, daß $g(x)$ unter den Elementen von U minimalen Grad hat. Nachzuweisen bleibt, daß U zyklisch ist. Da die Wörter $g(x)$, $xg(x)$, ..., $x^{k-1}g(x)$ eine Basis von U bilden, ist die Zyklizität von U äquivalent dazu daß die zyklische Verschiebung $w(x) = x^k g(x) - x^n + 1$ wieder zu U gehört, denn die zyklische Verschiebung eines Codeworts ist eine Linearkombination der zyklischen Verschiebungen der Basiselemente. Nach Voraussetzung gilt $w(x) = g(x)\left(x^k - q(x)\right)$ und $\operatorname{grd}\left(x^k - q(x)\right) < k$, woraus $w(x) \in U$ folgt. Für nicht normierte echte Teiler von $x^n - 1$ argumentiert man analog. □

3.0.10 Definition. Ist $g(x)$ ein Generatorpolynom eines zyklischen Codes $U \leq F^n$, so heißt $h(x) := (x^n - 1)/g(x)$ das zugehörige *Kontrollpolynom*.

3.0.11 Beispiel. Wir bestimmen alle zyklischen binären Codes der Länge 7 indem wir $x^7 - 1$ vollständig faktorisieren:

$$x^7 - 1 = (x-1)\left(x^3 + x + 1\right)\left(x^3 + x^2 + 1\right)$$

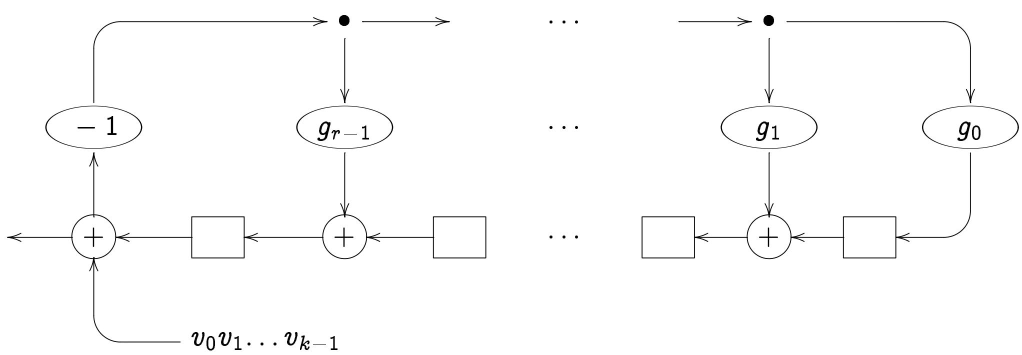
In Abschnitt 3.2 werden wir sehen, daß diese Faktoren *irreduzibel*, d.h., nicht weiter zerlegbar sind.

$g(x)$	U
1	$\mathbb{Z}_2^7$
$x-1$	Paritätskontrollcode
$x^6 + x^5 + x^4 + x^3 + x^2 + x^1 + 1$	Wiederholungscode
$x^3 + x + 1$	$(7,4)$-Hammingcode
$x^3 + x^2 + 1$	$(7,4)$-Hammingcode
$(x-1)(x^3 + x + 1)$	ausgedünnter Hammingcode
$(x-1)(x^3 + x^2 + 1)$	ausgedünnter Hammingcode

Man beachte, daß Codes mit komplementären Faktoren von $x^7 - 1$ als Generatorpolynom zueinander dual sind. Außerdem liefert das Produkt zweier Generatorpolynome das Generatorpolynom des Durchschnitts der Codes. Speziell die Multiplikation mit $x - 1$ entspricht der Ausdünnung, d.h., dem Durchschnitt mit dem Code der Wörter gerader Parität.

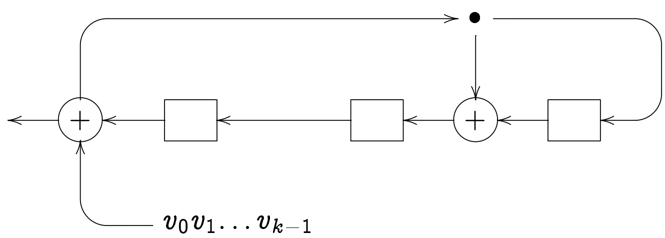
Wie codiert man geschickt? Wie die aus dem Generatorpolynom gewonnene Generatormatrix zeigt, ist die direkte Codierung durch Multiplikation vom links mit G meist nicht systematisch. Wir behalfen uns daher mit einem Trick: anstelle der Standardübersetzung $\vec{u} = u_0u_1 \ldots U - n - 1 \mapsto \sum_{i<n} u_i x^i$ ordnen wir den Komponenten des Tupels $\vec{u}$ *fallende* Potenzen von x zu, die mit x^{n-1} beginnen. Genauer: wir setzen $\tilde{u}(x) = \sum_{i<n} u_i x^{n-i-1}$. Dies ist ebenfalls ein Isomorphismus zwischen F^n und dem Untervektorraum der Polynome vom Grad $< n$ von $f[x]$. Division des Polynoms $\tilde{u}(x)$ durch $g(x)$ liefert $\tilde{u}(x) = q(x)g(x) + \tilde{r}(x)$ mit $\operatorname{grd} \tilde{r}(x) < n - k$. Damit ist $\tilde{u}(x) - \tilde{r}(x) = q(x)g(x)$ ein Codewort, und die Informationsbits in $\tilde{u}(x)$ sowie die Kontrollbits in $\tilde{r}(x)$ erscheinen nacheinander, d.h., wir haben eine systematische Codierung. Achtung: die $-\tilde{r}(x)$ entsprechenden Kontrollbits sind ebenfalls nach *fallenden* Potenzen von x zu interpretieren und an das zu codierende Wort anzuhängen.

Die eben beschriebene Codierung, wie auch die Decodierung im nächsten Abschnitt, basiert auf der Polynomdivision mit Rest. Wandelt man die übliche manuelle Methode so ab, daß in jedem Schritt nur ein Term des Dividenden benötigt wird, läßt sich dafür eine relativ einfache Schieberegistermaschine konstruieren:



Nach Abarbeitung der Eingabesymbole v_0, v_1, …, v_{k-1} stehen in den Schieberegistern genau die Koeffizienten des Restes $\tilde{r}(x)$ der Division von $\tilde{v}(x)$ durch $g(x)$.

3.0.12 Beispiel. Für das Generatorpolynom $g(x) = x^3 + x + 1$ über $\mathbb{Z}_2$ erhalten wir die folgende Maschine



Für $g_j = 0$ können wir die Multiplikatoren weglassen und wegen $-1 = 1$ in $\mathbb{Z}_2$ entfällt das Negieren. Um etwa das Wort 1101 zu codieren, ist das Polynom $x^6 + x^5 + x^3$ durch $g(x)$ zu dividieren und der Rest zu bestimmen:

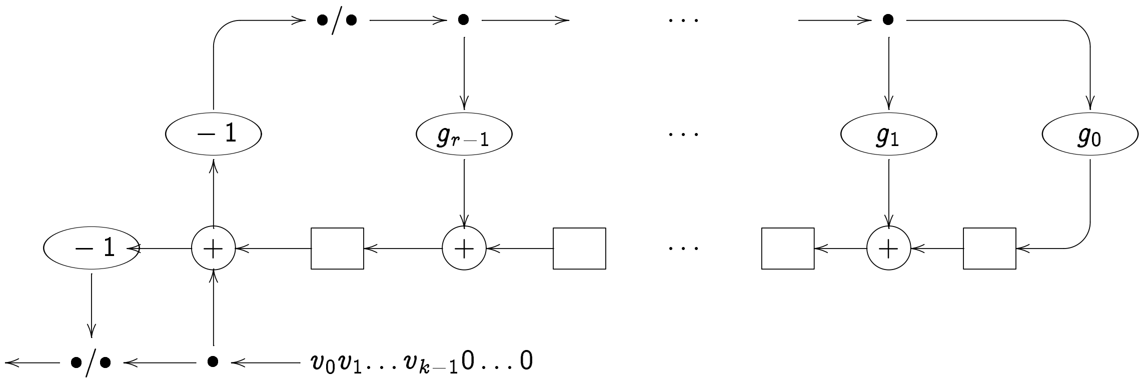
$$
\begin{array}{ccccccccccccccc}
 & 1x^6 & + & 1x^5 & + & 0x^4 & + & 1x^3 & & & & & & & \\
\hline
 & 0x^6 & + & 0x^5 & + & 0x^4 & & & & & & & & & \\
-(& 1x^6 & + & 0x^5 & + & 1x^4 & + & 1x^3 &) & & & & & & \\
\hline
 & & & 0x^5 & + & 1x^4 & + & 1x^3 & & & & & & & \\
 & & -(& 1x^5 & + & 0x^4 & + & 1x^3 & + & 1x^2 &) & & & & \\
\hline
 & & & & & 1x^4 & + & 1x^3 & + & 0x^2 & & & & & \\
 & & & & -(& 1x^4 & + & 0x^3 & + & 1x^2 & + & 1x^1 &) & & \\
\hline
 & & & & & & & 1x^3 & + & 1x^2 & + & 1x^1 & & & \\
 & & & & & & -(& 0x^3 & + & 0x^2 & + & 0x^1 & + & 0x^0 &) \\
\hline
 & & & & & & & & & 1x^2 & + & 1x^1 & + & 0x^0 &
\end{array}
$$

Die oberste Zeile listet $\tilde{v}(x)$ auf. Wir initialisieren die Schieberegister mit Nullen, was dem Polynom $r_6(x) = 0x^6 + 0x^5 + 0x^4$ entspricht. Dann addieren wir v_0 zum Inhalt des ersten Schieberegisters und subtrahieren das entsprechende Vielfache von $x^3g(x)$ von $v_0x^6 + r_6(x)$, was als neuen Inhalt der Schieberegister die Koeffizienten von $r_5(x) = 0x^5 + 1x^4 + 1x^3$ liefert. Nun addieren wir v_1 zum Inhalt des ersten Schieberegisters und subtrahieren das entsprechende Vielfache von $x^2g(x)$ von $v_1x^5 + r_5(x)$, was die Koeffizienten von $r_4(x) = x^4 + x^3 + 0x^2$ liefert. Da $\tilde{v}(x)$ keine Summanden vom Grad < 3 hat, ist $r_2(x) = x^2 + x + 0$ der Rest der Division. Andernfalls sind die Terme des Dividenden vom Grad < 3 zu $r_2(x)$ zu addieren.

Offenbar ist das Mitführen der Potenzen von x in dieser Rechnung redundant. Die entsprechend vereinfachte Rechnung sieht folgendermaßen aus:

$$
\begin{array}{ccccccccc}
 & 1 & 1 & 0 & 1 & & & & \\
\hline
 & 0 & 0 & 0 & & & & & \\
-(& 1 & 0 & 1 & 1 &) & & & \\
\hline
 & & 0 & 1 & 1 & & & & \\
 & -(& 1 & 0 & 1 & 1 &) & & \\
\hline
 & & & 1 & 1 & 0 & & & \\
 & & -(& 1 & 0 & 1 & 1 &) & \\
\hline
 & & & & 1 & 1 & 1 & & \\
 & & & -(& 0 & 0 & 0 & 0 &) \\
\hline
 & & & & & 1 & 1 & 0 &
\end{array}
$$

Die systematische Codierung hängt die Koeffizienten des Restpolynoms an das Wort $\vec{v}$ an. Durch Einbau zweier Schalter in die Schieberegistermaschine läßt sich dieses Verhalten erreichen:



Der obere Schalter ist während der ersten k Takte geschlossen und während der letzten $r = n-k$ Takte offen. Der untere Dreiwegeschalter läßt während der ersten k Takte die Koeffizienten von $\tilde{v}(x)$ passieren, und während der letzten r Takte den Inhalt der Schieberegister abfließen. Nach den k Koeffizienten von $\tilde{v}(x)$ werden r Nullen eingespeist.

3.1 Decodierung Zyklischer Codes

Der Begriff des Syndroms hat sich bei der Decodierung linearer Codes als nützlich erwiesen. Wie läßt er sich auf die vorliegende Situation übertragen?

3.1.00 Definition. Ist $g(x)$ ein echter Faktor von $x^n - 1$, so verstehen wir unter dem *Syndrompolynom* $s(x)$ von $w(x) \in F[x]_{x^n-1}$ bzüglich $g(x)$ den Rest, der bei der Division von $w(x)$ durch $g(x)$ auftritt.

Offenbar kann man ein beliebiges Vielfaches von $g(x)$ in $F[x]_{x^n-1}$ zu $w(x)$ addieren, ohne das Syndrompolynom zu verändern. Man beachte, daß Syndrome hier für Polynome und nicht für Tupel definiert worden sind, da es verschiedene Übersetzungen zwischen Tupeln und Polynomen gibt. Für $\vec{w} \in F^n$ unterscheiden wir

- das *direkte Syndrom*, d.h., die Rückübersetzung des Syndrompolynoms von $w(x)$ nach F^n, geordnet nach steigenden Potenzen von x, sowie
- das *systematische Syndrom*, d.h., die Rückübersetzung des Syndroms von $\tilde{w}(x)$ nach F^n, geordnet nach fallenden Potenzen von x.

Trotzdem gilt natürlich in beiden Fällen, daß das empfangene Wort $\vec{w}$ und der aufgetretene Fehler $\vec{e}$ dasselbe Syndrom aufweisen.

Die Standarddecodierung linearer Codes, die mit einer Liste aus Äquivalenzklassenanführern und zugehörigen Syndromen auskommt, ist auch hier anwendbar. Allerdings sollte die Zyklizität eines Codes weitere Vereinfachungen zulassen. Wir werden zwei vereinfachte Methoden vorstellen, den "Meggit"-Decoder und die Decodierung durch "Einfangen der Fehler". Zunächst wollen wir uns aber überlegen, wie sich das Syndrom eines Wortes bei der zyklischen Verschiebung des Wortes verändert.

3.1.01 Proposition. *Ist $g(x)$ ein echter Faktor von $x^n - 1$ und hat $w(x) \in F[x]_{x^n-1}$ das Syndrompolynom $s(x)$, so stimmen die Syndrompolynome von $xw(x) \in F[x]_{x^n-1}$ und von $xs(x) \in F[x]_{x^n-1}$ überein.*

Beweis: Wir wollen in $F[x]$ rechnen. Dort hat das Produkt $xw(x) \in F[x]_{x^n-1}$ die Darstellung $xw(x) - w_0x^n + w_0$, während $xs(x) \in F[x]_{x^n-1}$ wegen $\operatorname{grd} s(x) < \operatorname{grd} g(x) < n$ auch in $F[x]$ die Form $xs(x)$ hat. Mit $w(x) = q(x)g(x) + s(x)$ und $x^n - 1 = g(x)h(x)$ erhalten wir

$$\begin{aligned} xw(x) - w_0x^n + w_0 &= xw(x) - w_0\,(x^n - 1) \\ &= xw(x) - w_0 g(x)h(x) \\ &= x\big(s(x) - q(x)g(x)\big) - w_0 g(x)h(x) \\ &= xs(x) + g(x)\big(xq(x) - w_0 h(x)\big) \end{aligned}$$

Damit haben $xw(x) - w_0x^n + w_0$ und $xs(x)$ bei der Division durch $g(x)$ denselben Rest. □

Achtung: bei systematischer Codierung entspricht das Polynom $x\tilde{w}(x) \in F[x]_{x^n-1}$ der zyklischen Vertauschung des Ausgangsvektors $\vec{w}$ um eine Position nach links!

3.1.02 Beispiel. Für die Variante des $(7,4)$-Hamming-Codes mit Generatorpolynom $g(x) = x^3 + x + 1$ erhalten wir die folgende Standardtabelle:

Fehlermuster	Syndrompolynom
0	0
1	1
x	x
x^2	x^2
x^3	$x+1$
x^4	x^2+x
x^5	x^2+x+1
x^6	$x^2 \quad +1$

Empfangen wir etwa das systematisch codierte Wort $\vec{w} = 1001101$, so gilt $\tilde{w}(x) = x^6 + x^3 + x^2 + 1$. Unser Divisionsalgorithmus liefert

$$
\begin{array}{ccccccccc}
\cline{2-8}
 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & \\
\cline{2-8}
 & 0 & 0 & 0 & & & & & \\
-(& 1 & 0 & 1 & 1 &) & & & \\
\cline{2-5}
 & & 0 & 1 & 1 & & & & \\
-(& 0 & 0 & 0 & 0 &) & & & \\
\cline{3-6}
 & & & 1 & 1 & 0 & & & \\
 & & -(& 1 & 0 & 1 & 1 &) & \\
\cline{4-7}
 & & & & 1 & 1 & 1 & & \\
 & & & -(& 0 & 0 & 0 & 0 &) \\
\cline{5-8}
 & & & & & 1 & 1 & 0 &
\end{array}
$$

woraus wir wegen $101 + 110 = 011$ den Rest $x + 1$ ablesen. Also ist der Koeffizient von x^3 zu korrigieren, und das abgesandte Wort war $\vec{v} = 1000$.

3.1.03 Algorithmus. Der *Meggitt* Decoder: Anstelle der vollständigen Liste aller Syndrome und dazu gehöriger Fehlermuster minimalen Hamming-Gewichts benötigen wir nur die reduzierte Liste, die die korrigierbaren Fehlermuster maximalen Grades enthält. In Beispiel 3.1.02 ist das nur die letzte Zeile mit dem Syndrompolynom x^2+1. Der Algorithmus besteht aus 4 Schritten:

(0) Die ersten k Komponenten des empfangenen Worts $\vec{w} \in F^n$ werden in k Takten in die Schieberegistermaschine zur Polynomdivision durch $g(x)$ eingegeben; aus der Summe der Schieberegister mit den letzten r Komponenten von $\vec{w}$ ergibt sich das systematische Syndrom $s(x)$.

(1) Wir lassen die Maschine weitere $n-1$ Takte laufen, unter Eingabe der letzten r Komponenten von $\vec{w}$ und weiterer $k-1$ Nullen. Dabei erhalten wir die Syndrome der $n-1$ Linksverschiebungen von $\vec{w}$, indem wir jeweils die Schieberegister zu den entsprechenden verbliebenen Komponenten der Eingabe addieren.

(2) Sobald eines der aufgelisteten Syndrome erscheint, korrigieren wir die aktuellen Verschiebung von $\vec{w}$, indem wir linksbündig das entsprechende Fehlermuster subtrahieren, und fahren mit dem so korrigierten Wort fort.

(3) Eine letzte Linksverschiebung zum Schluß liefert das gesuchte Codewort.

3.1.04 Beispiel. Fortsetzung von Beispiel 3.1.02:

$$
\begin{array}{ccccccccccccccc}
 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
 & 0 & 0 & 0 \\
-(& 1 & 0 & 1 & 1 &) \\
 & & 0 & 1 & 1 \\
 & -(& 0 & 0 & 0 & 0 &) \\
 & & & 1 & 1 & 0 \\
 & & -(& 1 & 0 & 1 & 1 &) \\
 & & & & 1 & 1 & 1 \\
 & & & -(& 0 & 0 & 0 & 0 &) \\
 & & & & & 1 & 1 & 0 & \rlap{$\Longrightarrow s^{(0)}(x)=x+1$} \\
 & & & & -(& 0 & 0 & 0 & 0 &) \\
 & & & & & & 1 & 0 & 0 & \rlap{$\Longrightarrow s^{(1)}(x)=x^2+x$} \\
 & & & & & -(& 1 & 0 & 1 & 1 &) \\
 & & & & & & & 0 & 1 & 1 & \rlap{$\Longrightarrow s^{(2)}(x)=x^2+x+1$} \\
 & & & & & & -(& 1 & 0 & 1 & 1 &) \\
 & & & & & & & & 1 & 0 & 1 & \rlap{$\Longrightarrow s^{(3)}(x)=x^2+1, \text{ Korrektur!}$} \\
 & & & & & & & -(& 1 & 0 & 1 & 1 &) \\
 & & & & & & & & & 0 & 0 & 1 & \rlap{$\Longrightarrow s^{(4)}(x)=1$} \\
 & & & & & & & & -(& 0 & 0 & 0 & 0 &) \\
 & & & & & & & & & & 0 & 1 & 0 & \rlap{$\Longrightarrow s^{(5)}(x)=x$} \\
 & & & & & & & & & -(& 0 & 0 & 0 & 0 &) \\
 & & & & & & & & & & & 1 & 0 & 0 & \rlap{$\Longrightarrow s^{(6)}(x)=x^2$}
\end{array}
$$

Wir haben hier die Syndrompolynome so nummeriert, daß die zu korrigierende Komponente direkt ablesbar ist. In diesem Fall handelt es sich um w_3, und wir decodieren $\vec{w}=1001101$ zu $\vec{v}=1000$. Wir können die Korrektur auch direkt in die Rechnung einfließen lassen, indem wir bei der Subtraktion von der Zeile, in der eine Korrektur notwendig wird, das am weitesten rechts stehende Schieberegister modifizieren (die korrigierte signifikanteste Komponente der aktuellen Verschiebung von $\vec{w}$ wird zyklisch an den rechten Rand getauscht). In unserem konkreten

Beispiel treten dann in den Schieberegistern nur noch Nullen auf. Somit wissen wir daß keine weiteren Korrekturen mehr folgen können.

Die zweite Decodiermethode, die wir vorstellen wollen, kommt aus dem Bereich der Korrektur von *Burst-Fehlern*. Darunter verstehen wir Fehler, die lokalisiert in Gruppen auftreten, im Gegensatz zu statistisch gleichmäßig verteilten Fehlern ("weißes Rauschen"). Einzelfehler sind trivialerweise lokalisiert, aber die Methode eignet sich auch für manche Codes, die Doppelfehler korrigieren können.

3.1.05 Definition. Ein *zyklisches Fehlerbündel* der Länge l in F^n ist ein Wort $\vec{e} \in F^n$ mit Hamminggewicht $\leq l$, dessen von 0 verschiedene Komponenten innerhalb von l zyklisch aufeinanderfolgenden Positionen auftreten.

Einzelfehler sind immer Bündel der Länge 1.

3.1.06 Proposition. *Der zyklische (n,k)-Code $U \leq F^n$ möge maximal t Fehler korrigieren. Falls im empfangenen Wort $\vec{w}$ höchstens t Fehler aufgetreten sind, so gilt:*

(0) *hat das systematische Syndrom Hamminggewicht $\leq t$, so stimmt es mit dem Fehlerpolynom überein;*

(1) *ist der Fehler ein Bündel der Länge $n-k$, so gibt es eine zyklische Vertauschung von $\vec{w}$, deren systematisches Syndrom ein Hamminggewicht $\leq t$ hat (d.h., der Fehler läßt sich "einfangen").*

Beweis: (0) Das Fehlerpolynom $\tilde{e}(x) = \tilde{w}(x) - (\tilde{v}(x) - \tilde{r}(x))$ mit Hamminggewicht $\leq t$ hat dasselbe systematische Syndrompolynom $s(x)$ wie $\tilde{w}(x)$, also gilt $\tilde{e}(x) = q(x)g(x) + s(x)$ für ein geeignetes $q(x)$. Hat $s(x)$ Hamminggewicht $\leq t$, so hat das Codewort $\tilde{e}(x) - s(x)$ ein Hamminggewicht von maximal $2t < d$, muß also das Nullwort sein.

(1) $\tilde{e}(x)$ sei ein Fehlerbündel der Länge $n-k$. Dann gibt es eine zyklische Verschiebung $\vec{y}$ von $\vec{w}$, so daß alle Fehler in den letzten $n-k$ Komponenten auftreten. Das entsprechende Fehlerpolynom $\tilde{f}(x)$ hat dann einen Grad $< n-k$. Da $\tilde{y}(x)$ und $\tilde{f}(x)$ dasselbe systematische Syndrom haben, und dies nach (0) mit dem Fehler $\tilde{f}(x)$ übereinstimmt, ist das Hamminggewicht dieses Polynoms nach oben durch t beschränkt. □

3.1.07 Algorithmus. Decodierung durch Fehlereinfangen: Ähnlich wie beim Meggitt-Decoder werden die systematischen Syndrome $s^{(i)}(x)$, $i < n$, des empfangenen Worts $\tilde{w}(x)$ und seiner zyklischen Linksvertauschungen berechnet. Sobald eines dieser Polynome ein Hamminggewicht $\leq t$ hat, addieren wir es rechtsbündig zur aktuellen Vertauschung von $\vec{w}$ und haben damit alle Fehler auf einmal korrigiert. Beachte: wenn mehrere systematische Syndrome Hamminggewicht $\leq t$ haben, ist es egal, an welcher Stelle wir die Korrektur vornehmen. Immer werden alle Fehler korrigiert.

3.1.08 Beispiel. Fortsetzung von Beispiel 3.1.04: Im obigen Beispiel taucht nach 11 Takten erstmals ein systematisches Syndrom mit einem Hamminggewicht ≤ 1 auf, nämlich 001 . Die aktuelle Vertauschung von $\vec{w}$ beginnt mit w_4 und endet mit w_3 . Damit wird durch rechtsbündige Addition von 001 zu 1011001 die Komponente w_3 korrigiert, wie zuvor auch.

3.1.09 Beispiel. Der schon mehrfach erwähnte Golay-Code ist ein zyklischer, perfekter, 3 Fehler korrigierender $(23, 12)$-Binärcode mit dem Generatorpolynom

$$g(x) = x^{11} + x^{10} + x^6 + x^5 + x^4 + x^2 + 1$$

Falls zwei Fehler auftreten, bilden sie ein Bündel der Länge $11 = 23 - 12$, also funktioniert die Decodierung durch Fehlereinfangen. Falls drei Fehler in einem Bündel der Länge > 12 auftreten, schlägt diese Decodierungsmethode fehl (warum?). Also verändern wir w_0 und versuchen es wieder. Nach spätestens elf Versuchen muß die Decodierung durch Fehlereinfangen funktionieren.

3.2 Algebraische Körpererweiterungen

In Abschnitt 3.0 haben wir gesehen, daß zyklische Codes der Länge n über einem endlichen Körper F genau die Ideale des Faktorraums $F[x]_{x^n-1}$ sind und durch die echten Faktoren von x^n-1 bestimmt werden. Wie findet man diese Faktoren?

3.2.00 Definition. Ein Polynom $p(x) \in F[x]$ heißt *irreduzibel*, falls es kein Produkt von Polynomen kleineren Grades ist.

3.2.01 Beispiele. (0) Jedes Polynom mit einem Grad ≤ 1, d.h., der Form $ax^0 = a$ oder $ax+b$ ist automatisch irreduzibel.

(1) x^2+1 ist irreduzibel über $\mathbb{Z}_3$ und über $\mathbb{R}$, aber nicht über $\mathbb{Z}_2$: wegen $1+1=0$ gilt dort $x^2+1=(x+1)^2$. Dies zeigt, daß der Begriff der Irreduzibilität nur relativ zu einem spezifischen Körper sinnvoll ist.

(2) Jedes Polynom $p(x)$ mit Grad ≥ 3 über $\mathbb{R}$ ist reduzibel: hat $p(x)$ eine reelle Wurzel oder Nullstelle a, ist bekanntermaßen $p(x)$ durch $x-a$ teilbar. Hat $p(x)$ die konjugiert complexen Wurzeln $a+ib$ und $a-ib$, so ist $p(x)$ durch

$$(x-(a+ib))\,(x-(a-ib)) = ((x-a)-ib)\,((x-a)+ib) = (x-a)^2+b^2$$

teilbar.

Irreduzible Polynome verhalten sich im Polynomring weitgehend so wie wir es von Primzahlen in $\mathbb{Z}$ kennen.

3.2.02 Satz. *Jedes Polynom $p(x)$ in $F[x]$ ist ein Produkt*

$$p(x) = k \prod_{i<m} p_i(x)$$

mit $k \in F$ und $p_i(x)$ irreduzibel und normiert für alle $i<m$. Für $p(x) \neq 0$ ist diese Faktorisierung bis auf die Reihenfolge der Faktoren eindeutig.

Beweis: Induktion über den Grad von $p(x)$: Für $\operatorname{grd} p(x) \leq 1$ ist die Behauptung klar. Wir nehmen nun an, sie gilt auch für Polynome vom Grad $\leq n$ und betrachten $p(x)$ mit $\operatorname{grd} p(x) = n+1$. Ist $p(x)$ irreduzibel, so spalten wir nur den Leitkoeffizienten k ab. Andernfalls zerlegen wir $(1/k)p(x)$ in zwei normierte Faktoren vom Grad $< n$, auf die die Annahme anwendbar ist. Dies zeigt die Existenz einer Faktorisierung $p(x) = k\prod_{i<m} p_i(x)$. Zum Nachweis der Eindeutigkeit betrachten wir eine weitere Faktorisierung $p(x) = l\prod_{j<l} q_j(x)$. OBdA können wir annehmen, daß $\operatorname{grd} p_0(x) \leq \operatorname{grd} q_0(x)$ gilt, also finden wir Polynome $q(x)$ und $r(x)$ mit $q_0(x) = q(x)p_0(x) + r(x)$ und $\operatorname{grd} r(x) < \operatorname{grd} p_0(x)$. Folglich gilt

$$p_0(x)\left(\prod_{1\leq i<m} p_i(x) - q(x)\prod_{1\leq j<l} q_j(x)\right) = r(x)\prod_{1\leq j<l} q_j(x)$$

Aber diese beiden Polynome haben einen kleineren Grad als $p(x)$, also folgt nach Induktionsvoraussetzung die Existenz eines j_0 mit $1 \leq j_0 < l$ und $p_0(x) = q_{j_0}(x)$. $\square$

3.2.03 Beispiel. Das Nullpolynom ist über jedem Körper weder normiert noch irreduzibel. Für "kleine" Körper F und niedrige Grade lassen sich die normierten irreduziblen Polynome relativ leicht bestimmen. Das einzige solche Polynom vom Grad 0 ist natürlich $1 = x^0$, während es $|F|$ verschiedene derartige Polynome vom Grad 1 gibt, und zwar $x - a$, $a \in F$.

(0) Über $\mathbb{Z}_2$ sind x^2, $(x+1)^2$ und $x(x+1)$ reduzibel, und alle übrigen Polynome vom Grad 2 irreduzibel. Das einzige solche ist $x^2 + x + 1$. Mit Grad 3 sind nur die Polynome $x^3 + x^1$ sowie $x^3 + x^2 + 1$ irreduzibel und normiert, die beiden Polynome die $(7,4)$-Hamming-Codes erzeugen.

(1) $x^2 + x + 1$ hat über $\mathbb{Z}_3$ die Faktorisierung $(x+2)^2$, aber $x^2 + x + 2$ ist irreduzibel.

(2) Über $\mathbb{R}$ ist $x^2 + 1$ irreduzibel, $(x^2 - 1 = (x-1)(x+1)$ aber nicht.

Leider geht der Beweis des folgenden Satzes über den Rahmen dieser Vorlesung hinaus.

3.2.04 Satz. *Für jede Primzahl p existieren über $\mathbb{Z}_p$ irreduzible Polynome jeden Grades.* $\square$

Wir hatten bereits gesehen, daß im Gegensatz zu $\mathbb{R}$ Polynome über einem endlichen Körper F nicht mit Funktionen von F nach F identifiziert werden können, aber natürlich induziert jedes Polynom eine derartige Funktion.

3.2.05 Definition. Die von einem Polynom $p(x) = \sum_{i<m} p_i x^i$ in $F[x]$ induzierte Funktion, die $a \in F$ auf $\sum_{i<m} p_i a^i$ abbildet, bezeichnen wir mit $\widehat{p(x)}$. Abkürzend schreiben wir $p(a)$ statt $\widehat{p(x)}(a)$ für $a \in F$. Unter der *Wurzel* eines Polynoms verstehen wir eine Nullstelle der induzierten Funktion.

3.2.06 Lemma. *Ein Polynom $p(x) \in F[x]$ hat genau dann $a \in F$ als Wurzel, wenn $p(x)$ durch $x - a$ teilbar ist.*

Beweis: Wegen $p(x) = q(x)(x-a) + r(x)$ mit $\operatorname{grd} r(x) < \operatorname{grd} p(x)$ führt eine Induktion über den Grad von $p(x)$ leicht zum Ziel. $\square$

3.2.07 Corollar. *Irreduzible Polynome haben keine Wurzeln.* $\square$

Wir stellen uns nun die Frage, ob das Verfahren, das zu $\mathbb{R}$ eine formale Nullstelle i des irreduziblen Polynoms $x^2 + 1$ hinzufügt, und so den Körper C als Menge der Linearkombinationen von 1 und i induziert, auch auf endliche Körper und irreduzible Polynome anwendbar ist.

3.2.08 Beispiel. Ist α eine (formale) Wurzel des irreduziblen Polynoms x^2+x+1 über $\mathbb{Z}_2$, so ergeben sich nur vier Linearkombinationen: 0, 1, α und $\alpha+1$. Wegen $1+1=0$ erhalten wir die folgende Additionstabelle:

$+$	0	1	α	$\alpha+1$
0	0	1	α	$\alpha+1$
1	1	0	$\alpha+1$	α
α	α	$\alpha+1$	0	1
$\alpha+1$	$\alpha+1$	α	1	0

In Aufgabe 15 haben wir diese vierelementige Gruppe bereits kennengelernt. Sie ist auch unter dem Namen *Klein'sche Vierergruppe* bekannt. Was die Multiplikation angeht, so haben wir nach Konstruktion $\alpha^2=\alpha+1$. Daher erhalten wir die folgende Tabelle:

$\cdot$	0	1	α	$\alpha+1$
0	0	0	0	0
1	0	1	α	$\alpha+1$
α	0	α	$\alpha+1$	1
$\alpha+1$	0	$\alpha+1$	1	α

Man weist nun leicht die Körperaxiome nach.

Um weitere Körper zu produzieren, können wir aber auch analog zu Proposition 2.0.17 verfahren. Der Beweis läßt sich direkt übertragen:

3.2.09 Satz. *Für jedes Polynom $p(x)\in F[x]$ gilt: $F[x]_p(x)$ ist genau dann ein Körper, wenn $p(x)$ irreduzibel ist.* □

Dieser Körper enthält alle möglichen Reste bei Division durch $p(x)$ als Elemente, von denen es $|F|^{\operatorname{grd} p(x)}$ gibt. Wir benötigen einen weiteren Satz aus der Algebra, dessen Beweis über den Rahmen dieses Skripts hinausgeht.

3.2.10 Satz. *Zwei endliche Körper mit der gleichen Anzahl von Elementen sind isomorph.* □

Insbesondere stimmt der oben konstruierte Körper auf der Menge $\{0,1,\alpha,\alpha+1\}$ mit $(\mathbb{Z}_2)_{x^2+x+1}$ überein.

3.2.11 Definition. Für eine Primzahl p bezeichnen wir die Erweiterung von $\mathbb{Z}_p$ durch ein irreduzibles Polynom vom Grad n als den *Galois-Körper* $GF(p^n)$.

3.2.12 Beispiele. (0) Die ersten endlichen Körper sind gegeben durch

$$\begin{aligned} GF(2) &= \mathbb{Z}_2 \\ GF(3) &= \mathbb{Z}_3 \\ GF(4) &= \mathbb{Z}_2[x]_{x^2+x^1} \\ GF(5) &= \mathbb{Z}_5 \\ GF(7) &= \mathbb{Z}_7 \\ GF(8) &= \mathbb{Z}_2[x]_{x^3+x+1} = \mathbb{Z}_2[x]_{x^3+x^2+1} \\ GF(9) &= \mathbb{Z}_3[x]_{x^2+x+2} \end{aligned}$$

(1) Die innere Struktur von $GF(8) = \mathbb{Z}_2[x]_{x^3+x+1}$ wird aus folgender Tabelle deutlich. Die letzte Zeile ist eine Übersetzung der Polynome in α in Tupelschreibweise, geordnet nach fallenden Potenzen von α.

0	1	α	$\alpha+1$	α^2	α^2+1	$\alpha^2+\alpha$	$\alpha^2+\alpha+1$
–	α^7	α^1	α^3	α^2	α^6	α^4	α^5
000	001	010	011	100	101	110	111

Wählt man stattdessen eine formale Wurzel β für x^3+x^2+1, so lassen sich die Elemente von $GF(8)$ etwas anders organisieren:

0	1	β	$\beta+1$	β^2	β^2+1	$\beta^2+\beta$	$\beta^2+\beta+1$
–	β^7	β^1	β^5	β^2	β^3	β^6	β^4
000	001	010	011	100	101	110	111

Da beide Körper isomorph sind, müssen wir β in der ersten Menge und α in der zweiten Menge wiederfinden können. Aus der Potenzdarstellung ergibt sich $\beta = \alpha^{-1} = \alpha^6$ und $\alpha = \beta^{-1} = \beta^6$.

(2) Auf ähnliche Weise erhalten wir eine Tabellendarstellung von $GF(9)$, wobei wir die formale Wurzel von x^2+x+2 wieder mit α bezeichnen:

0	1	2	α	$\alpha+2$	$\alpha+1$	2α	$2\alpha+1$	$2\alpha+2$
–	α^8	α^4	α^1	α^6	α^7	α^5	α^2	α^3
00	01	02	10	11	12	20	21	22

3.3 Zyklische Codes mittels Wurzeln

Wie wir gesehen haben, sind die Generatorpolynome zyklischer Codes $U \leq F^n$ durch die echten Faktoren von $x^n - 1$ in $F[x]$ bestimmt. Es wäre nun sehr bequem, wenn wir $x^n - 1$ in Linearfaktoren zerlegen könnten, etwa $x^n - 1 = \prod_{i<n}(x - a_i)$. Aber zumeist existieren solche a_i nicht im Ausgangskörper F . Andererseits lassen sich Körper um formale Wurzeln irreduzibler Polynome erweitern, also werden wir versuchen, die a_i in einem solchen Erweiterungskörper zu finden. Es stellt sich dann heraus, daß alle Generatorpolynome zyklischer Codes in F^n durch Mengen dieser Wurzeln bestimmt werden, allerdings ist die entsprechende Funktion nicht injektiv. Die Wurzelmengen können unter gewissen Potenzen abgeschlossen werden, ohne daß sich das Generatorpolynom ändert. Der Zusammenhang wird durch den Begriff des Minimalpolynoms hergestellt.

Für Gruppen hatten wir den Begriff der Ordnung schon eingeführt, vergl. Definition 2.0.06 sowie Proposition 2.0.07.

3.3.00 Definition. Unter der *Ordnung* $\mathrm{ord}(a)$ eines von 0 verschiedenen Elements a eines Körpers F verstehen wir seine Ordnung bezüglich der multiplikativen Gruppe $\langle F \setminus \{0\}, \cdot, 1\rangle$. a heißt *primitiv*, falls a die Ordnung $|F| - 1$ hat.

Konkret bedeutet dies $\mathrm{ord}(a) = \min\{\, n > 0 : a^n = 1\,\}$.

3.3.01 Lemma. *Hat $a \in F \setminus \{0\}$ die Ordnung n , so gilt $x^n - 1 = \prod_{i<n}(x - a^i)$.*

Beweis: Wegen $(a^i)^n = (a^n)^i = 1$ ist jede Potenz a^i , $i < n$, eine Wurzel von $x^n - 1$. Da all diese Wurzeln verschieden sind (vergl. Proposition 2.0.07), teilt das Produkt der Linearfaktoren $x^n - 1$. Andererseits haben beide Polynome den Grad n und sind normiert, also folgt ihre Gleichheit. □

3.3.02 Corollar. *Ist F' eine algebraische Körpererweiterung von F mit einem Element $a \in F' \setminus 0$ der Ordnung n , so wird das normierte Generatorpolynom jedes zyklischen Codes $U \leq F^n$ schon durch seine einfachen Wurzeln der Form a^i , $i < n$, in F' bestimmt.*

Beweis: Das Generatorpolynom $g(x)$ eines zyklischen Codes $U \leq F^n$ dividiert $x^n - 1 = \prod_{i<n}(x - a^i)$ über $F'[x]$, also gibt es nach Satz 3.2.02 eine Auswahl von $m < n$ verschiedenen Exponenten i_j , $j < m$, so daß gilt $g(x) = \prod_{j<m}(x - a^{i_j})$. Damit besteht U aus genau den Polynomen $w(x) \in F[x]$, deren Grad kleiner als n ist und die $w(a^{i_j}) = 0$, $j < n - k$, erfüllen. □

3.3.03 Beispiel. In $GF(8) = \mathbb{Z}_2[x]_{x^3+x+1}$ haben sowohl α wie auch α^3 die Ordnung 7.

(0) Der Code $U = \{\, u(x) \in \mathbb{Z}_2[x] : \mathrm{grd}\, u(x) < n \wedge u(\alpha) = 0 \,\}$ enthält das Polynom $g(x) = x^3 + x + 1$ und nach Konstruktion kein Polynom kleineren Grades. Über $GF(8)[x]$

faktorisiert $g(x)$ nun folgendermaßen:

$$g(x) = (x - \alpha)(x - \alpha^2)(x - \alpha^4)$$

(1) Der Code $U' = \{ u(x) \in \mathbb{Z}_2[x] : \operatorname{grd} u(x) < n \wedge u(\alpha^3) = 0 \}$ muß ein Generatorpolynom in $\mathbb{Z}_2[x]$ haben, das über $GF(8)[x]$ durch $x - \alpha^3$ teilbar ist. Aber welche anderen Linearfaktoren der Form $x - \alpha^i$ sind nötig, damit das Produkt tatsächlich in $\mathbb{Z}_2[x]$ liegt?

Die in Beispiel 3.3.03(1) aufgetretenen Frage können wir mit Korollar 3.3.02 noch nicht beantworten. Zunächst wollen wir daher unsere Rechenfertigkeiten in endlichen Körpern verbessern:

3.3.04 Satz. *Wir betrachten eine Primzahl p*

(0) *Für jedes $n \in \mathbb{N}$ und alle $a, b \in \mathbb{Z}_p$ gilt $(a+b)^{p^n} = a^{p^n} + b^{p^n}$.*

(1) *Hat $w(x) \in \mathbb{Z}_p[x]$ eine Wurzel a in einer algebraischen Erweiterung von $\mathbb{Z}_p$, so ist für jedes $n \in \mathbb{N}$ auch a^{p^n} eine Wurzel von $w(x)$.*

Beweis: (0) Induktion über n: Für $n = 0$ ist die Behauptung trivial. Nehmen wir also an, für ein $n > 0$ sei die Behauptung erfüllt, und betrachten

$$(a+b)^{p^{n+1}} = \left((a+b)^{p^n}\right)^p = \left(a^{p^n} + b^{p^n}\right)^p$$

Damit reduziert sich der Induktionsschritt darauf, die Behauptung für $n = 1$ explizit nachzuweisen. Nach der binomischen Formel gilt für $A, B \in \mathbb{Z}_p$

$$(a+b)^p = \sum_{k \le p} \binom{p}{k} A^{p-k} B^k = A^p + \sum_{1 \le k < p} \binom{p}{k} A^{n-k} B^k + B^p$$

Wegen $p! = \binom{p}{k} k!(p-k)!$ muß für jedes k mit $1 \le k < p$ der Binominalkoeffizient $\binom{n}{k}$ durch p teilbar sein, denn $p!$ ist durch p teilbar, aber weder $k!$ noch $(p-k)!$ sind durch p teilbar. Da in $\mathbb{Z}_p$ gilt $p = 0$, entfallen alle mittleren Terme. □

(1) Da die Ordnung jedes von 0 verschiedenen Elements eines endlichen Körpers F die Zahl $|F| - 1$ teilt (vergl. HA 31), erhalten wir für $w(x) = \sum_{i<n} w_i x^i$ wegen (0)

$$(w(x))^p = \sum_{i<n} w_i^p x^{ip} = \sum_{i<n} w_i x^{ip} = w(x^p)$$

Damit ist jede p-fache Potenz einer Wurzel wieder eine Wurzel. □

3.3.05 Beispiel. Wir können nun Beispiel 3.3.03(1) fortsetzen: außer α^3 müssen auch $(\alpha^3)^2 = \alpha^6$, $(\alpha^3)^4 = \alpha^{12} = \alpha^5$ usw. Wurzeln des gesuchten Generatorpolynoms sein. Da höhere Potenzen von α^3 keine neuen Wurzeln liefern, steht zu vermuten, daß $(x-\alpha^3)(x-\alpha^6)(x-\alpha^5)$ das Generatorpolynom sein könnte. Ausmultiplizieren dieser Linearfaktoren über $GF(8) = \mathbb{Z}_2[x]_{x^3+x+1}$ liefert $x^3 + x^2 + 1$, also tatsächlich einen alten Bekannten aus $\mathbb{Z}_2[x]$.

Wir müssen uns noch davon überzeugen, daß dieses Produkt von Linearfaktoren in $GF(8)[x]$ nicht zufällig ein Polynom in $\mathbb{Z}_2[x]$ liefert.

3.3.06 Definition. Unter einem *Minimalpolynom* eines Elementes a einer algebraischen Körpererweiterung F' eines endlichen Körpers F verstehen wir ein normiertes Polynom minimalen Grades in $F[x]$ mit Wurzel a.

3.3.07 Lemma. *Minimalpolynome existieren zu jedem Element einer algebraischen Körpererweiterung. Sie sind eindeutig bestimmt und irreduzibel.*

Beweis: Zu F' existiert ein irreduzibles Polynom $f(x) \in F[x]$, so daß gilt $F' = F[x]_{f(x)}$. Die zugehörige Polynomfunktion auf F' bildet $a \in F'$ auf $f(a) \in F'$ ab. Die Ordnung dieses Elements teilt $|F'| - 1$, insbesondere gilt $(f(a))^{\operatorname{ord} f(a)} = 1$. Folglich hat das Polynom $(f(x))^{\operatorname{ord} f(a)} - 1$ die Wurzel a. Da ein Polynom mit Wurzel a existiert, gibt es auch ein normiertes Polynom minimalen Grades mit dieser Wurzel.

Nehmen wir an, $p(x)$ und $p'(x)$ sind Minimalpolynome von $a \in F'$. Da beide denselben Grad haben, gilt $p(x) = p'(x) + r(x)$ mit $\operatorname{grd} r(x) < \operatorname{grd} p'(x)$. Wegen $p(a) = p'(a) = 0$ folgt daraus $r(a) = 0$. Also ist $r(x)$ das Nullpolynom.

Die Irreduzibilität folgt schließlich aus der Nullteilerfreiheit des Körpers F'. □

3.3.08 Definition. Wir bezeichnen das eindeutig bestimmte Minimalpolynom eines Elementes a einer algebraischen Körpererweiterung F' eines endlichen Körpers F mit $M_a^F(x)$. Sofern keine Mißverständnisse zu befürchten sind, schreiben wir auch kurz $M_a(x)$.

3.3.09 Satz. *Ist p eine Primzahl und a Element einer algebraischen Körpererweiterung F' von $\mathbb{Z}_p$, so gilt für die kleinste Zahl $j \in \mathbb{N}$ mit $a = a^{p^j}$*

$$M_a^{\mathbb{Z}_p}(x) = M_a(x) = \prod_{i<j} \left(x - a^{p^i}\right)$$

Beweis: Wir setzen $m(x) := \prod_{i<j} \left(x - a^{p^i}\right)$ und zeigen zunächst, daß $m(x)$ zu $\mathbb{Z}_p[x]$ gehört. Wegen

$$(m(x))^p = \prod_{i<j} \left(x - a^{p^i}\right)^p = \prod_{i<j} \left(x^p - a^{p^i}\right) = m(x^p)$$

folgt aus $m(x) = \sum_{i<m} d_i x^i \in F'[x]$ sofort

$$\sum_{i<m} d_i^p x^{ip} = \sum_{i<m} d_i x^{jp}$$

also $d_i^p = d_i$ für alle $i < m$. Damit sind die Koeffizienten $d_i \in F'$, $i < m$, auch Wurzeln von $x^p - x$. Aber dieses Polynom hat maximal p Wurzeln, und alle Elemente von $\mathbb{Z}_p$ sind solche, also gehören die Koeffizienten schon zu $\mathbb{Z}_p$. Nach Satz 3.3.04(1) teilt $m(x)$ jedes Polynom in $\mathbb{Z}_p[x]$ mit Wurzel a, also insbesondere das Minimalpolynom $M_a(x)$. Wegen der Minimalität desselben, stimmen beide Polynome überein. □

3.3.10 Corollar. *Ist a ein Element der Ordnung n in einer algebraischen Erweiterung F' eines endlichen Körpers F, dann ist für jede Auswahl von $m < n$ verschiedenen Exponenten i_j, $j < m$, das Generatorpolynom des zyklischen Codes*

$$\{ w(x) \in F[x] : \operatorname{grd} w(x) < n \wedge w(a^{i_j}) = 0, j < m \} \leq F^n$$

durch das Produkt aller verschiedenen Minimalpolynome $M_{a^{i_j}}(x)$, $j < m$, gegeben.

3.3.11 Beispiel. In $GF(8) = \mathbb{Z}_2[x]_{x^3+x+1}$ wählen wir die Potenzen α, α^2, α^3 und α^4 des Elements α der Ordnung 7 aus. Die entsprechenden Minimalpolynome in $\mathbb{Z}_2[x]$ sind gegeben durch

$$\begin{aligned} M_\alpha(x) &= x^3 + x + 1 = M_{\alpha^2}(x) = M_{\alpha^4}(x) \\ M_{\alpha^3}(x) &= x^3 + x^2 + 1 \end{aligned}$$

Damit erhalten wir das Generatorpolynom

$$g(x) = (x^3 + x + 1)(x^3 + x^2 + 1) = x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$$

Der zugehörige Code ist dann der Wiederholungscode, vergl. Beispiel 3.0.11.

In welchen Körpererweiterungen lohnt es sich, nach Elementen einer vorgegebenen Ordnung zu suchen? Die Antwort liefert die Zahlentheorie.

3.3.12 Definition. (0) Zwei natürliche Zahlen heißen *relativ prim*, falls 1 ihr größter gemeinsamer Teiler ist.

(1) Die *Euler-Funktion* $\mathbb{N} \xrightarrow{\varphi} \mathbb{N}$ bildet $n \in \mathbb{N}$ auf die Anzahl der echt kleineren Zahlen ab, die relativ prim zu n sind.

3.3.13 Lemma. (0) *Für jede Primzahl p und jedes $k \geq 1$ gilt $\varphi(p^k) = p^k - p^{k-1}$.*

(1) *Für verschiedene Primzahlen p und q gilt $\varphi(pq) = (p-1)(q-1)$.* □

3.3.14 Satz. (Satz von Euler-Fermat) *Für relativ prime natürliche Zahlen q und n ist $q^{\varphi(n)} - 1$ durch n teilbar.* □

3.3.15 Corollar. *Ist p eine Primzahl die $n \in \mathbb{N}$ nicht teilt, so enthält der Erweiterungskörper $GF(p^{\varphi(n)})$ ein Element der Ordnung n.*

Beweis: Da $GF(p^{\varphi(n)})$ ein Element a der Ordnung $p^{\varphi(n)} - 1 = qn$ enthält, ist a^q ein Element der Ordnung n. □

4 BCH-Codes

4.0 Zyklische Binärcodes, die 1 bzw. 2 Fehler korrigieren

Ist $a \in GF(2^r)$ ein primitives Element a, d.h., $\operatorname{ord} a = 2^r - 1$, so wollen wir den Code

$$U = \{\, u(x) \in \mathbb{Z}_2[x] : \operatorname{grd} u(x) < n \wedge u(a) = 0 \,\}$$

näher betrachten. Im letzten Abschnitt hatten wir gesehen, daß sich im Fall $r = 3$ dabei für α und α^3 Einzelfehler korrigierende $(7,4)$-Hammingcodes ergeben. Gilt dies auch allgemeiner?

Die Bedingung $u(a) = 0$ können wir in eine Bedingung an ein zu $u(x)$ gehöriges Tupel umwandeln. Dabei gibt es zwei Möglichkeiten, je nachdem, ob

- direkt codiert werden soll, d.h., durch Multiplikation mit dem Generatorpolynom oder der entsprechenden Generatormatrix, oder
- systematisch codiert werden soll, mittels Polynomdivision.

$$\sum_{i<n} u_i a^i = 0 \iff \vec{u} \cdot \begin{pmatrix} 1 \\ a \\ a^2 \\ \vdots \\ a^{n-2} \\ a^{n-1} \end{pmatrix} = 0 \quad \text{bzw.} \quad \sum_{i<n} u_i a^{n-1-i} = 0 \iff \vec{u} \cdot \begin{pmatrix} a^{n-1} \\ a^{n-2} \\ \vdots \\ a^2 \\ a \\ 1 \end{pmatrix} = 0$$

Der durch die Potenzen von a beschriebene Spaltenvektor legt den Gedanken an eine binäre Kontrollmatrix nahe. Wir erhalten tatsächlich eine solche, wenn wir die Potenzen von a durch die entsprechenden Tupel in $\mathbb{Z}_2^r$ ersetzen. Dabei ist es irrelevant, ob diese binären Tupel nach steigenden oder fallenden Potenzen von A geordnet sind, denn elementare Spaltenoperationen überführen eine Kontrollmatrix eines Codes U in eine Kontrollmatrix desselben Codes. Wir werden hier immer nach fallenden Potenzen von a ordnen. Die Primitivität von a garantiert $2^r - 1$ verschiedene Zeilen, also liegt tatsächlich ein binärer Hamming-Code vor.

4.0.00 Beispiel. Die primitiven Elemente α und α^3 in $GF(8) = \mathbb{Z}_2[x]_{x^3+x+1}$ liefern bei systematische Codierung die folgenden Matrizen (vergl. Beispiel 3.2.12(1)):

$$\begin{pmatrix} \alpha^6 \\ \alpha^5 \\ \alpha^4 \\ \alpha^3 \\ \alpha^2 \\ \alpha \\ 1 \end{pmatrix} \mapsto H^{\text{syst}} = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{und} \quad \begin{pmatrix} \alpha^4 \\ \alpha \\ \alpha^5 \\ \alpha^2 \\ \alpha^6 \\ \alpha^3 \\ 1 \end{pmatrix} \mapsto H^{\text{syst}} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

Die bei direkter Codierung relevanten Matrizen entstehen durch Zeilenvertauschung.

Weist das empfangene Wort $\vec{w}$ in Position $i < n$ einen Fehler auf, also $\tilde{w}(x) = \tilde{u}(x) + x^{n-1-i}$, wobei $\tilde{u}(x)$ das dem gesendeten Codewort $\vec{u}$ entsprechende Polynom ist, so liefert die Auswertung von $\tilde{w}(x)$ an der Stelle a den Wert $\tilde{w}(a) = \tilde{u}(a) + a^{n-1-i} = a^{n-1-i}$, das Syndrom. Daraus läßt sich unmittelbar die zu korrigierende Position ablesen.

Man beachte: dieses Verfahren bestimmt nur die Position des Fehlers. Erst die Tatsache, daß wir über $\mathbb{Z}_2$ arbeiten, ermöglicht uns dann auch die Korrektur. Über einem anderen Körper wäre nicht klar, wie groß die Korrektur zu sein hat.

4.0.01 Beispiel. Wird im obigen Beispiel das Wort 1001000 empfangen, so entspricht das bei systematischer Codierung dem Polynom $x^6 + x^3$. Dessen Wert an der Stelle α ist $\alpha^6 + \alpha^3 = \alpha^2 + \alpha = \alpha^4$. Also ist die Position Nummer $2 = 6 - 4$ zu korrigieren: das gesuchte Codewort ist 1011000, und die ursprüngliche Nachricht ist 1011.

Bezüglich der Codierung mittels α^3 ist das Polynom $x^6 + x^3$ an der Stelle α^3 auszuwerten, was $\alpha^{18} + \alpha^9 = \alpha^4 + \alpha^2 = \alpha = (\alpha^3)^5$ liefert. Folglich ist die Position Nummer $1 = 6 - 5$ zu korrigieren und wir erhalten das Codewort 1101000, sowie die Ausgangsnachricht 1101.

Wählen wir andererseits die Darstellung $\mathbb{Z}_2[x]_{x^3+x^2+1}$ für $GF(8)$ mit formaler Wurzel β der Ordnung 7, so ist das Polynom $x^6 + x^3$ zunächst an der Stelle β auszuwerten. Dies liefert $\beta^6 + \beta^3 = \beta + 1 = \beta^5$, was die Korrektur an Position Nummer $1 = 6 - 5$ erfordert, genau wie bei dem durch α^3 induzierten Code. Dies ist kein Zufall, denn wie wir gesehen haben, gilt $\beta = \alpha^6$, und jedes der Elemente α^3, α^6 und $\alpha^{12} = \alpha^5$ erzeugt denselben Code. Folglich sollte bei dem durch β^3 induzierten Code Position Nummer 2 korrigiert werden. Die Rechnung bestätigt das: $\beta^{18} + \beta^9 = \beta^4 + \beta^2 = \beta^5 = (\beta^3)^4$

Bei direkter Codierung entspricht das empfangene Wort dem Polynom $1 + x^3$. An der Stelle α hat dies den Wert α, daher ergibt sich in diesem Fall das Codewort 1101000. Dagegen führt α^3 auf den Funktionswert $1 + \alpha^2 = \alpha^6 = (\alpha^3)^2$ und damit auf das Codewort 1011000.

Es sollte nicht verwundern, daß bei verschiedenen Decodierungsmethoden immer dieselben beiden Codewörter auftauchen: es gibt eben nur zwei zyklische Hammingcodes im $\mathbb{Z}_2^7$ und jeder ist das Spiegelbild des anderen. Die beiden möglichen Interpretationen eines Vektors als Polynom in steigenden oder fallenden Potenzen entspricht dem Übergang von dem einen Code zu seinem Spiegelbild.

Lassen sich auf ähnliche Weise auch Codes erzeugen, die mehr als einen Fehler korrigieren können? Ist a wieder ein primitives Element einer Erweiterung von $\mathbb{Z}_2$, so können wir für die Codepolynome statt nur einer Wurzel an der Stelle a noch weitere Wurzeln verlangen. Da mit a auch a^{2^n} eine Wurzel ist, brauchen wir nur solche Exponenten von a zu betrachten, die keine 2er Potenzen sind. Der naheliegenste Kandidat ist also a^3.

In Beispiel 3.3.11 hatten wir über $GF(8)$ in der Darstellung $\mathbb{Z}_2[x]_{x^3+x+1}$ mit formaler Wurzel α genau diese Forderung gestellt und als Ergebnis den Wiederholungscode der Länge 7 erhalten, der maximal 3 Fehler korrigieren kann. Das Korrekturverfahren, das wir vorstellen werden, kann aber nur maximal zwei Fehler korrigieren, nutzt also die Fähigkeiten des Codes nicht voll aus. Anders ausgedrückt: der Code ist besser als für die Aufgabe, zwei Fehler korrigieren zu können, nötig wäre.

4.0.02 Definition. Ist a ein Element der Ordnung n in einer Erweiterung F' von $\mathbb{Z}_2$, so ist der *Bose-Chaudhuri-Hocquenghem* oder kurz *BCH-Code* der Länge n für Doppelfehler gegeben durch

$$U = \{\, u(x) \in F[x] : \operatorname{grd} u(x) < n \wedge u(a) = u(a^3) = 0 \,\}$$

Analog zur Situation für Hamming-Codes gilt $u(a) = u(a^3) = 0$ genau dann wenn

$$\vec{u} \cdot \begin{pmatrix} a^{n-1} & a^{3(n-1)} \\ a^{n-2} & a^{3(n-2)} \\ \vdots & \vdots \\ a^2 & a^6 \\ a & a^3 \\ 1 & 1 \end{pmatrix} = (0\ 0) \quad \text{bzw.} \quad \vec{u} \cdot \begin{pmatrix} 1 & 1 \\ a & a^3 \\ a^2 & a^6 \\ \vdots & \vdots \\ a^{n-2} & a^{3(n-2)} \\ a^{n-1} & a^{3(n-1)} \end{pmatrix} = (0\ 0)$$

Das *erweiterte Syndrom* eines empfangenen Worts $\vec{w}$ ist bei systematischer Codierung gegeben durch $(\tilde{w}(a)\ \tilde{w}(a^3)) \in (F')^2$ und bei direkter Codierung durch $(w(a), w(a^3)) \in (F')^2$.

4.0.03 Beispiel. Wir geben zunächst $GF(2^4) = \mathbb{Z}_2[x]_{x^4+x+1}$ mit der formalen Wurzel α für $x^4 + x + 1$ tabellarisch an, allerdings aus Platzgründen mit einer gegenüber Beispiel 3.2.12 transponierten Tabelle. Die Tupel sind nach fallenden Potenzen von α geordnet.

Polynom in α	Potenz von α	4-Tupel	Minimalpolynom
0	0	0000	x
1	1	0001	$x+1$
α	α	0010	x^4+x+1
α^2	α^2	0100	x^4+x+1
α^3	α^3	1000	$x^4+x^3+x^2+x+1$
$\alpha+1$	α^4	0011	x^4+x+1
$\alpha^2+\alpha$	α^5	0110	x^2+x+1
$\alpha^3+\alpha^2$	α^6	1100	$x^4+x^3+x^2+x+1$
$\alpha^3+\alpha+1$	α^7	1011	x^4+x^3+1
α^2+1	α^8	0101	x^4+x+1
$\alpha^3+\alpha$	α^9	1010	$x^4+x^3+x^2+x+1$
$\alpha^2+\alpha+1$	α^{10}	0111	x^2+x+1
$\alpha^3+\alpha^2+\alpha$	α^{11}	1110	x^4+x^3+1
$\alpha^3+\alpha^2+\alpha+1$	α^{12}	1111	$x^4+x^3+x^2+x+1$
$\alpha^3+\alpha^2+1$	α^{13}	1101	x^4+x^3+1
α^3+1	α^{14}	1001	x^4+x^3+1

Die zugehörige Kontrollmatrix hat die folgende Form:

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

In der rechten Hälfte wiederholen sich fünf 4-Tupel je dreimal, da das Element α^3 in $GF(16)$ die Ordnung 5 hat (im Gegensatz zu $GF(8)$).

Das Generatorpolynom diese zyklischen Codes ergibt sich nach Korollar 3.3.10 als

$$g(x) = M_\alpha(x)M_{\alpha^3}(x) = (x^4+x+1)(x^4+x^3+x^2+x+1) = x^8+x^7+x^6+x^4+1$$

Wie decodieren wir einen solchen BCH-Code? Wir betrachten zunächst den Fall daß zwei Fehler auftreten. Wird das Polynom $\tilde{w}(x) = \tilde{u}(x) + x^i + x^j$ empfangen, mit $i \neq j < n$, so erhalten wir als Komponenten des Syndroms

$$s_0 = \tilde{w}(a) = a^i + a^j \qquad \text{und} \qquad s_1 = \tilde{w}(a^3) = a^{3i} + a^{3j}$$

woraus folgt

$$(s_0)^3 = a^{3i} + 3a^{2i}a^j + 3a^ia^{2j} + a^{3j} = s_1 + a^ia^js_0$$

oder äquivalent

$$a^ia^j = (s_0)^2 + s_1(s_0)^{-1}$$

Aber damit sind a^i und a^j Lösungen der quadratischen Gleichung

$$\lambda^2 + s_0\lambda + (s_0)^2 + s_1(s_0)^{-1} = 0 \tag{4.0-00}$$

Zu korrigieren sind dann die Positionen $n-1-i$ und $n-1-j$.

Falls nur ein Fehler auftritt und wir $\tilde{w}(x) = \tilde{u}(x) + x^i$ empfangen, folgt aus $s_0 = a^i$ und $s_1 = a^{3i}$, daß $(s_0)^2 + s_1(s_0)^{-1} = 0$ erfüllt ist. Damit hat die quadratische Gleichung (4.0-00) die Lösungen a^i und 0 . Wir korrigieren die Position $n-1-i$.

Läßt sich die quadratische Gleichung (4.0-00) im Erweiterungskörper nicht lösen, oder ergibt sich $s_0 = 0 \neq s_1$, so sind mehr als zwei Fehler aufgetreten.

Man beachte, daß die quadratische Formel wegen des Faktors 2 im Nenner in Erweiterungen von $\mathbb{Z}_2$ nicht benutzt werden kann. Wir werden aber im übernächsten Abschnitt ein Verfahren kennenlernen, das die Lösung dieser Gleichung überflüssig macht.

Es bleibt noch zu überprüfen, ob die Wahl des Erweiterungskörpers Einfluß auf den BCH-Code hat. Erfreulicherweise ist dies nicht der Fall, aber der Beweis geht über den Rahmen des Skripts hinaus.

4.0.04 Satz. *Sind a und $\bar{a}$ zwei Elemente der Ordnung n in zwei Erweiterungskörpern von $\mathbb{Z}_2$, so sind die durch a und a^3 bzw. $\bar{a}$ und $\bar{a}^3$ definieren BCH-Codes äquivalent.* □

4.1 BCH-Codes allgemein

Die Überlegungen aus dem vorigen Abschnitt legen es nahe, auch über beliebigen endlichen Körpern Codes mit vorgegebenen Fehlerkorrekturfähigkeiten durch hinreichend viele Wurzeln zu definieren.

4.1.00 Definition. Ist $\beta \in F'$ ein Element der Ordnung n in einer Erweiterung von F, so ist der *BCH-Code* der Länge n über F für t Fehler ist gegeben durch

$$U = \{\, u(x) \in F[x] : \operatorname{grd} u(x) < n \;\wedge\; u(\beta^i) = 0 \text{ für } i < 2t \,\}$$

4.1.01 Beispiele. (0) Für $F = \mathbb{Z}_2$ und $t = 1$ werden β und β^2 als Wurzeln verlangt. Aber β^2 als Wurzel zu verlangen ist nach Satz 3.3.04(1) redundant. Also erhalten wir auf diese Weise alle binären Hammingcodes. Für $t = 2$ folgt mit einem entsprechenden Argument, daß die Bedingungen $u(\beta^2) = u(\beta^4) = 0$ automatisch erfüllt sind. Also erhalten wir die in Definition 4.0.02 spezifizierten Codes.

(1) Für $F = \mathbb{Z}_2$ und $n = 15 = 2^4 - 1$ betrachten wir das primitive Element $\alpha \in GF(16)$, das formale Wurzel des irreduziblen Polynoms $x^4 + x^2 + 1$ ist. Aus der Tabelle in Beispiel 4.0.03 können wir für verschiedene Werte von t die entsprechenden Generatorpolynome bestimmen:

t	Generatorpolynom	Zahl der Infobits
1	$x^4 + x^2 + 1$	11
2	$(x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)$	7
3	$(x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)(x^2 + x + 1)$	5

(2) Für $F = \mathbb{Z}_3$ und $t = 1$ sind die Forderungen $u(\beta) = 0$ und $u(\beta^2) = 0$ unabhängig. Allerdings handelt es sich bei diesen BCH-Codes nicht um Hamming-Codes. Im Fall $n = 8$ etwa hat die zugehörige ternäre Kontrollmatrix die Dimensionen 8×4. Nach HA18 ist aber die Maximalzahl paarweise linear unabhängiger Vektoren im $\mathbb{Z}_3^4$ durch $1 + 3 + 9 + 27 = 40$ gegeben. Es stellt sich heraus, daß dieser BCH-Codes einen Minimalabstand von $d = 4$ hat.

Allgemein ist das Generatorpolynom eines BCH-Codes für t Fehler von der Form

$$g(x) = \prod_{i<2t} M_{\beta^i + 1}(x)$$

Die entsprechende F'-wertige Kontrollmatrix ist bei direkter Codierung gegeben durch

$$H^{\text{dir}} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ \beta & \beta^2 & \dots & \beta^{2t} \\ \beta^2 & (\beta^2)^2 & \dots & (\beta^2)^{2t} \\ \vdots & \vdots & \ddots & \vdots \\ \beta^{n-1} & (\beta^{n-1})^2 & \dots & (\beta^{n-1})^{2t} \end{pmatrix}$$

4.1.02 Satz. *Ein BCH-Code für t Fehler kann mindestens t Fehler korrigieren.*

Beweis: Nachzuweisen ist, daß ein von $\vec{0}$ verschiedenes Codewort mindestens ein Gewicht von $2t+1$ hat, oder äquivalent, daß jedes Codewort $\vec{u}$ mit $\|\vec{u}\| \le 2t$ der Nullvektor sein muß. Wir wählen beliebig $2t$ verschiedene der n möglichen Positionen aus, $i_j < n$ für $j < 2t$, und verlangen, daß alle übrigen Komponenten von $\vec{u}$ den Wert 0 haben. Die Codewortbedingung $\vec{u} \cdot H^{\mathrm{dir}} = \vec{0} \in (F')^{2t}$ ist nun äquivalent zu

$$(u_{i_0}, u_{i_1}, \ldots, u_{i_{2t-1}}) \cdot \begin{pmatrix} \beta^{i_0} & \left(\beta^{i_0}\right)^2 & \ldots & \left(\beta^{i_0}\right)^{2t} \\ \beta^{i_1} & \left(\beta^{i_1}\right)^2 & \ldots & \left(\beta^{i_1}\right)^{2t} \\ \vdots & \vdots & \ddots & \vdots \\ \beta^{i_{2t-1}} & \left(\beta^{i_{2t-1}}\right)^2 & \ldots & \left(\beta^{i_{2t-1}}\right)^{2t} \end{pmatrix} = \vec{0}$$

Um $\vec{u} = \vec{0}$ nachzuweisen genügt es nun zu zeigen, daß diese reduzierte quadratische Matrix eine von 0 verschiedene Determinante hat. Zur Vereinfachung der Schreibweise setzen wir $b_j := \beta^{i_j}$, $j < 2t$. Nun gilt

$$\det \begin{vmatrix} b_0 & (b_0)^2 & \ldots & (b_0)^{2t} \\ b_1 & (b_1)^2 & \ldots & (b_1)^{2t} \\ \vdots & \vdots & \ddots & \vdots \\ b_{2t-1} & (b_{2t-1})^2 & \ldots & (b_{2t-1})^{2t} \end{vmatrix} = \prod_{j<2t} b_j \det \begin{vmatrix} 1 & b_0 & \ldots & (b_0)^{2t-1} \\ 1 & b_1 & \ldots & (b_1)^{2t-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & b_{2t-1} & \ldots & (b_{2t-1})^{2t-1} \end{vmatrix}$$

Die letzte Determinante ist als *Vandermonde-Determinante* bekannt und hat den Wert

$$\prod_{J<i<2t} (b_i - b_j)$$

Wegen $\operatorname{ord}\beta = n$ sind alle Faktoren von 0 verschieden, also gilt dies auch für die Determinante. Da die Wahl der $2t$ Positionen beliebig war, beendet dies den Beweis. □

4.2 Reed-Solomon Codes

Die folgende spezielle Klasse von BCH-Codes kann für die Korrektur von Fehlerbündeln nützlich sein. F ist wieder ein beliebiger endlicher Körper.

4.2.00 Definition. Ein BCH-Code der Länge $|F|-1$ über F heißt *Reed-Solomon Code.*

In diesem Fall gehört das Element β der Ordnung $n = |F|-1$ bereits zu F statt zu einer echten Erweiterung, und ist folglich dort primitiv. Dies stellt natürlich eine gewisse Einschränkung an die möglichen Codelängen dar: nur die Vorgänger von Primzahlpotenzen können auftreten, etwa $1, 2, 3, 4, 6, 7, 9, 10, 12, 15, \ldots$.

4.2.01 Beispiel. Um einen Reed-Solomon Code der Länge 6 konstruieren zu können, brauchen wir ein primitives Element von $\mathbb{Z}_7$. Dafür kommen 3 oder 5 in Frage, die in $\mathbb{Z}_7$ multiplikative Inverse sind. (Weiterhin gilt $\operatorname{ord} 2 = \operatorname{ord} 4 = 3$ und $\operatorname{ord} 6 = 2$.)

Für $t = 1$ erhalten wir den Code

$$\{\, u(x) \in F[x] : \operatorname{grd} u(x) < 6 \wedge u(3) = u(2) = 0 \,\}$$

mit dem Generatorpolynom

$$g(x) = M_3(x)M_2(x) = (x-3)(x-2) = x^2 + 2x + 6$$

Dies ist ein $(6, 4)$-Code. Wir könne sogar verlangen, daß Doppelfehler korrigiert werden: $t = 2$ liefert den $(6, 2)$-Code

$$\{\, u(x) \in F[x] : \operatorname{grd} u(x) < 6 \wedge u(3) = u(2) = u(6) = u(4) = 0 \,\}$$

mit dem Generatorpolynom

$$g(x) = M_3(x)M_2(x)M_6(x)M_4(x) = (x-3)(x-2)(x-6)(x-4) = x^4 + 6x^3 + 3x^2 + 2x + 4$$

4.2.02 Lemma. *Der Reed-Solomon Code der Länge n für t Fehler über F ist ein $(n, n-2t)$-Code mit Generatorpolynom $\prod_{i<2t}(x - \beta^{i+1})$ und Minimalabstand $d = 2t + 1$.*

Beweis: Jedes Element a eines Körpers F hat das Minimalpolynom $M_a(x) = (x - a)$. Damit ergibt sich das gewünschte Generatorpolynom mit dem Grad $2t$ und einem Gewicht $\leq 2t + 1$. Andererseits erfüllt der Minimalabstand $d \geq 2t+1$, woraus $d = 2t+1$ folgt. Da Syndrome einen echt keineren Grad haben als das Generatorpolynom, stehen $2t$ Komponenten als Kontrollbits zur Verfügung. Damit bleiben $n - 2t$ Komponenten für die Informationsbits übrig. $\square$

Um binäre Fehlerbündel korrigieren zu können, betrachten wir Reed-Solomon Codes über Erweiterungskörpern der Form $GF(2^m)$, d.h., der Länge $n = 2^m - 1$. Offenbar induziert solch ein Reed-Solomon Code einen unterliegenden Binärcode der Länge $n \cdot m$: jedes Zeichen aus $GF(2^m)$ kann als binäres m-Tupel aufgefaßt werden.

4.2.03 Satz. *Der einem Reed-Solomon Code über $GF(2^m)$ entsprechende Binärcode korrigiert l-fache Fehlerbündel deren Länge den Wert $(\lfloor t/l \rfloor - 1)m + 1$ nicht überschreitet.*

Beweis: Ein Fehlerbündel der Länge $(t-1)m+1$ kann höchstens t benachbarte Komponenten aus $GF(2^m)$ beeinflussen. Entsprechend können l Fehlerbündel der angegebenen Länge höchstens l Gruppen von jeweils $\lfloor t/l \rfloor$ benachbarten Komponenten aus $GF(2^m)$ in Mitleidenschaft ziehen, insgesamt also höchstens t solcher Komponenten. □

4.2.04 Beispiel. Der BCH-Code der Länge 7 über $GF(8)$ für Doppelfehler kann etwa durch das primitive Element $\alpha \in GF(8)$, das formale Wurzel von $x^3 + x + 1 \in \mathbb{Z}_2[x]$ ist, erzeugt werden. Das Generatorpolynom in $GF(8)[x]$ ist dann

$$g(x) = (x-\alpha)(x-\alpha^2)(x-\alpha^3)(x-\alpha^4) = x^4 + \alpha^3 x^3 + x^2 + \alpha x + \alpha^3 \in GF(8)[x]$$

Es handelt sich um einen $(7,3)$-Code, dem ein binärer $(21,9)$-Code zugrunde liegt. Letzterer kann einzelne Fehlerbündel der Länge $(2-1)\cdot 3+1 = 4$ korrigieren. Sollen doppelte Fehlerbündel korrigiert werden, so dürfen sie nur die Länge $(1-1)\cdot 3+1 = 1$ haben, d.h., es sind gewöhnliche Doppelfehler.

4.3 Schnelle direkte Decodierung von BCH-Codes

Der binäre BCH-Code der Länge 15 für 2 Fehler hat das Generatorpolynom

$$g(x) = M_\alpha(x)M_{\alpha^3}(x) = x^8 + x^7 + x^6 + x^4 + 1 \ ,$$

ist also ein $(15,7)$-Code. Daher ist Decodierung mittels Fehlereinfangen möglich (je 2 Fehler sind höchstens 8 Positionen voneinander entfernt, zyklisch gerechnet). Wir können auch mittels der quadratischen Gleichung aus Kapitel 4.1 decodieren. Der Meggitt-Decoder ist immer anwendbar, aber noch langsamer. Daher stellt sich die Frage: wie decodiert man BCH-Codes direkt und effizient?

Wir betrachten ein Verfahren, das auf dem Euklidischen Algorithmus zur Bestimmung des größten gemeinsamen Teilers (GGT) sowie formalen Ableitungen beruht. Andere, kompliziertere Verfahren existieren, wie z.B. das von Berlekamp, welches bei Längen $\geq 10^6$ schneller ist.

Zur Erinnerung: in $\mathbb{Z}$ bzw. $\mathbb{N}$ finden wir den GGT von $a_0 > a_1 > 0$ mit der Rekursion

$$\begin{aligned} a_0 &= q_1 a_1 + a_2 \qquad \text{mit} \qquad a_2 < a_1 \\ a_k &= q_{k+1} a_{k+1} + a_{k+2} \qquad \text{mit} \qquad a_{k+2} < a_{k+1} \end{aligned}$$

die nach endlich vielen Schritten abbricht, nämlich sobald erstmals $a_{k+2} = 0$ gilt.

4.3.00 Satz. *Aus* $a_{K+2} = 0$ *folgt* $a_{K+1} = GGT(a_0, a_1)$.

Beweis: Wir zeigen zunächst, daß jeder gemeinsame Teiler von a_0 und a_1 auch a_k teilt, für $k \leq K+1$. Dazu definieren wir rekursiv zwei Zahlenfolgen $(m_i)_{i<K+3}$ und $(n_i)_{i<K+3}$ mittels

$$\begin{aligned} m_0 &= 1 & & & n_0 &= 0 \\ m_1 &= 0 & &\text{bzw.} & n_1 &= 1 \\ m_{k+2} &= m_k - q_k m_{k+1} & & & n_{k+2} &= n_k - q_k n_{k+1} \end{aligned}$$

Ausgehend von $a_0 = m_0 a_0 - n_0 a_1$ sowie $a_1 = -(m_1 a_0 + n_1 a_1)$ erhalten wir mit vollständiger Induktion

$$a_{k+2} = (-1)^{k+2}(m_{k+2}a_0 - n_{k+2}a_1)$$

Nun gilt es, a_{K+1} als Teiler von a_0 und a_1 zu identifizieren. Wegen $a_{K+2} = 0$ gilt offenbar $a_K = q_{K+1}a_{K+1}$, d.h., a_{K+1} teilt a_K . Zusammen mit $a_{K-1} = q_K a_K + a_{K+1}$ folgt nun a_{K+1} teilt a_{K-1} . Nach endlich vielen Schritten erhalten wir das gewünscht Ergebnis. □

Dasselbe Verfahren läßt sich fast direkt auf den Polynomring $F[x]$ übertragen, einzig die Bedingung $a_{k+2} < a_{k+1}$ ist durch $\operatorname{grd} a_{k+2} < \operatorname{grd} a_{k+1}$ zu ersetzen.

4.3.01 Definition. Für jedes von 0 verschiedene Polynom $f(x) = \sum_{i<n} f_i x^i \in F[x]$ definieren wir die *formale Ableitung*

$$f'(x) := \sum_{i<n} (i \cdot f_i) x^{i-1} \ ,$$

wobei $i \cdot f_i$ als i-fache Summe von f_i in F aufzufassen ist. In Tupelschreibweise folgt aus $\vec{f} = (f_0, f_1, f_2, \ldots)$ also

$$\vec{f}' = (f_1, f_2 + f_2, f_3 + f_3 + f_3, \ldots)$$

4.3.02 Proposition. *Für alle von* 0 *verschiedenen Polynome* $f(x)$ *und* $g(x)$ *und jedes Körperelement* $\lambda \in F$ *gilt:*

(0) $(f(x) + g(x))' = f'(x) + g'(x)$;

(1) $(\lambda f(x))' = \lambda f'(x)$;

(2) $(f(x)g(x))' = f'(x)g(x) + f(x)g'(x)$.

Beweis: (0) und (1) sind trivial; sie stellen die Linearität der Ableitungsfunktion fest.

(2) Wegen (0) und (1) können wir uns auf Potenzen von x beschränken.

$$\begin{aligned}(x^m x^n)' &= \left(x^{m+n}\right)' = (m+n)x^{(m+n)-1} \\ &= mx^{m-1+n} + nx^{n-1+m} = \left(mx^{m-1}\right)x^n + x^m\left(nx^{n-1}\right)\end{aligned}$$

Der Rest ist nun trivial. □

Unser Ziel ist die Decodierung eines durch ein Element $\beta \in F'$ der Ordnung n induzierten BCH-Codes für t Fehler. Das empfangene Wort $\vec{w}$ mit dem Fehler $\vec{e}$ liefert bei direkter Codierung das Syndrom

$$s(x) = \sum_{i<2t} w(\beta^{i+1})x^i = \sum_{I<2t} e(\beta^{i+1})x^i = \vec{w} \cdot \begin{pmatrix} 1 & 1 & \ldots & 1 \\ \beta & \beta^2 & \ldots & \beta^{2t} \\ \beta^2 & (\beta^2)^2 & \ldots & (\beta^2)^{2t} \\ \vdots & \vdots & \ddots & \vdots \\ \beta^{n-1} & (\beta^{n-1})^2 & \ldots & (\beta^{n-1})^{2t} \end{pmatrix} \in F'[x]$$

Wir nehmen an, daß $p \leq t$ Fehler an den Stellen i_j , $j < p$, auftreten. Damit hat das Fehlerpolynom die Form $e(x) = \sum_{j<p} e_j x^{i_j}$. Es sind $2p$ Zahlen zu bestimmen:

- die p Fehlerpositionen i_j , $j < p$;
- die p Fehlerwerte e_j , $j < p$.

Wir fassen diese unbekannten Zahlen wie folgt in zwei Polynomen zusammen:

4.3.03 Definition. Das *Fehlerstellenpolynom* $\sigma(x) \in F'[x]$ vom Grad p und das *Fehlerwertepolynom* $\omega(x) \in F'[x]$ vom Grad $p-1$ sind gegeben durch

$$\sigma(x) := \prod_{j<p}(1 - \beta^{i_j}x) \qquad \text{bzw.} \qquad \omega(x) := \sum_{k<p} e_k \beta^{i_k} \frac{\sigma(x)}{1 - \beta^{i_k}x}$$

Offenbar ist β^l genau dann eine Wurzel von $\sigma(x)$, wenn ein Fehler an der Stelle $k < p$ auftritt mit $\beta^k = \beta^{-l}$. Aber warum erlaubt uns das Polynom $\omega(x)$, die Fehlerwerte zu bestimmen? Dazu betrachten wir die Ableitung von $\sigma(x)$:

$$\sigma'(x) = \sum_{k<p}(-\beta^{i_k}) \prod_{k\neq j<p} (1-\beta^{i_j}) = -\sum_{k<p} \beta^{i_k} \frac{\sigma(x)}{1-\beta^{i_k}x}$$

Daraus ergibt sich für $j < p$

$$\omega(\beta^{i_k}) = -e_k\sigma'(\beta^{i_k}) \qquad \text{oder, äquivalent} \qquad e_k = -\frac{\omega(\beta^{i_k})}{\sigma'(\beta^{i_k})}$$

Nun wenden wir uns der Aufgabe zu, das Fehlerstellenpolynom und das Fehlerwertepolynom zu bestimmen.

4.3.04 Satz. *Die Polynome $\sigma(x)$ und $\omega(x)$ sind durch folgende Bedingungen eindeutig bestimmt:*

(0) *Die Polynome $\omega(x)$ und $\sigma(x)s(x)$ aus $F'[x]$ sind äquivalent modulo x^{2t}, d.h., die Koeffizienten aller Potenzen x^i mit $i < 2t$ stimmen überein;*

(1) $\mathrm{grd}(\sigma(x)) \leq t$;

(2) $\mathrm{grd}(\omega(x)) < t$;

(3) $\sigma(0) = 1$.

Beweis: Umschreiben des Syndroms liefert

$$\begin{aligned} s(x) &= \sum_{j<2t} w(\beta^{j+1})x^j = \sum_{j<2t} e(\beta^{j+1})x^j \\ &= \sum_{j<2t} (\sum_{k<p} e_k(\beta^{j+1})^{i_k})x^j = \sum_{j<2t}\sum_{k<p} e_k\beta^{i_k}(\beta^{i_k}x)^j = \sum_{k<p} e_k\beta^{i_k} \sum_{j<2t} (\beta^{i_k}x)^j \end{aligned}$$

Damit ist $s(x)$ äquivalent modulo x^{2t} zu der Potenzreihe (dies ist kein Polynom!)

$$\sum_{k<p} e_k\beta^{i_k} \sum_{j<infty} (\beta^{i_k}x)^j = \sum_{k<p} e_k\beta^{i_k} \frac{1}{1-\beta^{i_k}x} \tag{4.3-00}$$

Multiplikation mit $\sigma(x)$ liefert die Äquivalenz (0). Die Gradbedingungen folgen wegen $p \leq t$, während die Forderung $\sigma(0) = 1$ die Eindeutigkeit sichert. □

Bedingung (0) dieses Satzes liefert nun den Aufhänger zur Bestimmung des Fehlerwertepolynoms und des Fehlerstellenpolynoms mittels des Euklidischen Algorithmus. Und zwar wenden wir diesen an auf $a_0(x) := x^{2t}$ und $a_1(x) = s(x)$. Insbesondere sind für jedes k vor Beendigung des Algorithmus die Polynome

$$a_k(x) = (-1)^{k+1}(m_k(x)x^{2t} - n_k(x)s(x)) \qquad \text{und} \qquad (-1)^{k+1}n_k(x)s(x)$$

äquivalent modulo x^{2t}. Setzen wir nun $k_0 := \min\{\, k : \operatorname{grd} a_k < t \,\}$, so gilt für ein geeignetes Skalar $d \in F'$

$$\omega(x) = d a_{k_0}(x) \qquad \text{und} \qquad \sigma(x) = d(-1)^{k_0+1} n_{k_0}(x) \tag{4.3-01}$$

Der Wert von d ist wegen $\sigma(0) = 1$ gegeben durch $d = (-1)^{k_0+1}/n_{k_0}(0)$.

Somit erhalten wir ein Verfahren zur schnellen direkten Decodierung eines BCH-Codes:

4.3.05 Algorithmus. Der *BCH-Algorithmus* benötigt vier Schritte:

(0) Berechnung der Zahlen $s_i := w(\beta^{i+1})$, $i < 2t$. Dies liefert das (direkte) Syndrompolynom $s(x) := \sum_{i<2t} s_i x^i$;

(1) Anwendung des Euklidischen Algorithmus auf $a_0(x) = x^{2t}$ und $a_1(x) = s(x)$; Abbruch sobald $\operatorname{grd} a_{k_0}(x) < t$ gilt;

(2) rekursive Berechnung von $n_{k_0}(x)$; Bestimmung von $d = (-1)^{k_0+1}/n_{k_0}(0)$; damit erhalten wir $\omega(x)$ und $\sigma(x)$ gemäß der Formeln 4.3-01;

(3) Bestimmung aller Wurzeln von $\sigma(x)$; Korrektur von w_i genau dann, wenn β^{1-i} eine Wurzel ist, und zwar auf den Wert $w_i + \omega(\beta^{1-i})/\sigma'(\beta^{1-i})$. □

Wir beschließen dieses Kapitel mit einem Beispiel für die Anwendung dieses Algorithmus.

4.3.06 Beispiel. Für $F = \mathbb{Z}_2$, $t = 2$, $n = 2^4 - 1 = 15$ betrachten wir das primitive Element α von $F' = GF(16)$, das formale Wurzel von $x^4 + x^2 + x + 1$ ist. Es gilt $a_0(x) = x^{2t} = x^4$.

Nach dem Empfang des Wortes $w(x) = x^2 + x^6$ formen wir das (direkte) Syndrom

$$\begin{aligned} s_0 &= \alpha^2 + \alpha^6 = \alpha^3 \\ s_1 &= \alpha^4 + \alpha^{12} = \alpha^6 \\ s_2 &= \alpha^6 + \alpha^{18} = \alpha^2 \\ s_3 &= \alpha^8 + \alpha^{24} = \alpha^{12} \end{aligned}$$

also $s(x) = \alpha^{12}x^3 + \alpha^2x^2 + \alpha^6x + \alpha^3 = a_1(x)$. Der Euklidische Algorithmus liefert

$$q_1(x) = \alpha^3x + \alpha^8 \quad , \quad a_2(x) = \alpha^{13}x^2 + \alpha^8x + \alpha^{11} \quad \text{sowie} \quad q_2(x) = \alpha^{14}x + \alpha^{14} \quad , \quad a_3(x) = \alpha^{12} \,,$$

woraus sich $k_0 = 3$ ergibt. Aus den Startwerten $n_0(x) = 0$ und $n_1(x) = 1$ erhalten wir dann

$$n_2(x) = n_0(x) - q_1(x)n_1(x) = \alpha^3x + \alpha^8 \quad \text{sowie} \quad n_3(x) = n_1(x) - q_2(x)n_2(x) = \alpha^2x^2 + \alpha^{12}x + \alpha^9$$

Wegen $n_3(0) = \alpha^9$ ergibt sich der Normierungsfaktor $d = \alpha^6$, und dies liefert die Polynome

$$\omega(x) = \alpha^3 \quad , \quad \sigma(x) = \alpha^8 x^2 + \alpha^3 x + 1 \quad \text{und} \quad \sigma'(x) = 2 \cdot \alpha^8 x + \alpha^3 = \alpha^3$$

Damit haben die Fehler alle den Wert $\omega(x)/\sigma'(x) = 1$. Dies ist keine Überraschung, arbeiten wir doch über $F = \mathbb{Z}_2$! Schließlich sind noch die Nullstellen von $\sigma(x)$ zu bestimmen. Diese ergeben sich zu α^9 und α^{13}. Wegen $\alpha^{-9} = \alpha^6$ und $\alpha^{-13} = \alpha^2$ sind damit die Positionen 6 und 2 des empfangenen Wortes zu korrigieren. Das gesendete Wort ist das Null-Wort. □

Index

Jürgen Koslowski (koslowj@iti.cs.tu-bs.de)
Institut für Theoretische Informatik
TU Braunschweig
Postfach 3329
D-38023 Braunschweig
Germany