FEHLERKORRIGIERENDE CODES

Inhalt der Vorlesung

Jürgen Koslowski

Institut für Theoretische Informatik

Technische Universität Braunschweig

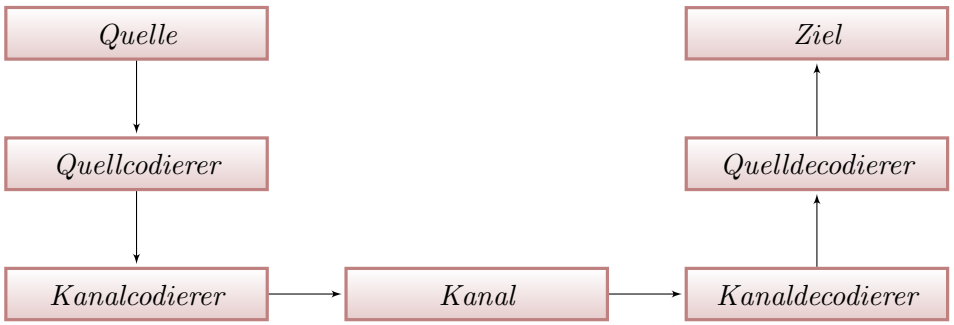
Juli 2009

Inhaltsverzeichnis

0 Einführung

0.0 Kommunikationssysteme

Das Standardmodell der Kommunikation von Quelle zu Ziel läßt sich sowohl in räumlicher wie in zeitlichen Hinsicht interpretieren:



(0.0-01)

- *räumlich*, z.B. Telefonie, Funkverkehr, Rundfunk- oder Fernsehübertragung, Kommunikation mit einem Satelliten im Weltraum;
- *zeitlich*, z.B. Abspeichern auf einem Datenträger von Textdokumenten, Bild- oder Musikdaten, auf die später wieder zugegriffen werden soll, um sie erneut zu lesen, zu betrachten oder anzuhören; Zellteilung.

Dabei dient die *Quellcodierung* vorrangig zwei Aufgaben:

- der Konvertierung der Quelldaten in ein dem Kanal angemessenes Format (etwa analog-digital, oder dezimal-binär);
- der Datenkompression.

Die *Quelldecodierung* funktioniert entsprechend umgekehrt. Dabei ist zu unterscheiden, ob die Quellinformationen *exakt* wiederhergestellt werden sollen (verlustfrei oder *lossless*), oder nur annähernd (verlustbehaftet oder *lossy*). Letzteres ist sicher immer dann der Fall, wenn ein Analog-Digital-Wandler zum Einsatz kommt. Aber auch bei digitalen Daten kann eine nur näherungsweise Wiederherstellung der Daten ausreichend sein, siehe etwas das JPEG-Format für Bilder im Gegensatz zum verlustfreien PNG-Format, oder MP3 bzw. Vorbis bei Audiodaten im Vergleich zu WAV oder auch FLAC.

Der übertragende *Kanal* wird aufgrund physikalischer Gegebenheiten oder konstruktionsbedingter Einschränkungen nicht perfekt sein. Designbedingt brauchen auch das Eingabe- und Ausgebealphabet nicht übereinzustimmen. Schließlich können beispielsweise bei magnetischen oder optischen Speichermedien bestimmte Bitsequenzen verboten sein. Die Aufgabe der Kanal-Codierung und -decodierung besteht nun darin, diese Einschränkungen zu umgehen und den Kanal damit so “transparent wie möglich” erscheinen zu lassen. Dabei kann durchaus eine weitere Übersetzung der Daten in ein anderes Alphabet auftreten.

0.1 Kanalcodierung und -decodierung

Unser Hauptmodell für einen Kanal wird *diskret* und *probabilistisch* sein. $S = \langle A, B, P\rangle$ besteht aus einem *Eingabealphabet* A, einem *Ausgabealphabet* B und einer Verteilung P *bedingter Wahrscheinlichkeiten*

$$P(\,\boldsymbol{y}\ \text{empfangen}\,|\,\boldsymbol{x}\ \text{gesendet}\,) \qquad \text{für alle } \langle \boldsymbol{x}, \boldsymbol{y}\rangle \in A^m \times B^m\,,\ m \in \mathbb{N}^+$$

Dabei nehmen wir an, daß der Kanal keine Zeichen verliert oder hinzufügt, also *längenerhaltend* arbeitet. Insofern können wir auch $P(\,\varepsilon\ \text{empfangen}\,|\,\varepsilon\ \text{gesendet}\,) = 1$ annehmen. Außerdem vereinbaren wir, daß die *Übertragungsreihenfolge* im Kanal mit der Schreibrichtung von links nach rechts übereinstimmen soll. Zweckmäßigerweise sollte ein Codierer oder ein Decoder die Bits in der Reihenfolge ihres Eintreffens bearbeiten, statt sie erst zu puffern.

Wir wollen jetzt den rechten Teil des Diagramms (0.0-01) genauer betrachten.

(0.1-02)

Die Quellcodierung möge M *Informationswörter* $\boldsymbol{u}$ liefern. Diesen werden durch die Kanalcodierung *Codewörter* $\boldsymbol{c} \in A^n$ zugeordnet und durch den Kanal geschickt. Die entsprechenden *empfangenen Wörter* $\boldsymbol{y} \in B^n$ werden mittels Kanaldecodierung zunächst (evtl. nur intern) in Codewörter $\boldsymbol{c}' \in A^n$ und anschließend in Informationswörter $\boldsymbol{u}'$ umgewandelt.

Ziel des Verfahrens ist es $\boldsymbol{u} = \boldsymbol{u}'$ sicherzustellen, wofür $\boldsymbol{c} = \boldsymbol{c}'$ eine Voraussetzung ist. Daraus folgt sofort, daß der Kanalcodierer eine *injektive Funktion* sein muß, es also ebenfalls M Codewörter gibt.

Ein nicht zu vernachlässigender Aspekt der Codierung, der Decodierung und insbesondere der Fehlerkorrektur wird die *Effizienz* dieser Operationen sein. Dazu führen wir den Begriff der *Coderate* ein, die Alphabetgröße $|A|$, Anzahl M der Informationswörter und Codewortlänge n in Beziehung setzt:

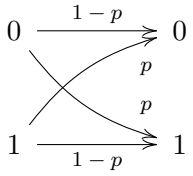
$$R := \frac{\log_{|A|} M}{n}$$

Der zunächst überraschend komplizierte Zähler erklärt sich daraus, daß die M Informationswörter häufig einen Raum der Form A^k aufspannen, dessen *Dimension* mit der Dimension des Raums A^n, in dem die Codewörter leben, in Beziehung gesetzt werden soll. Dann ergibt sich die Coderate zu $R = k/n$. Die Injektivität der Kanalcodierung garantiert $R \le 1$.

0.1.01 Beispiel. Für den *gedächtnislosen binären symmetrischen Kanal* mit *Umwandlungswahrscheinlichkeit* p (abkürzend BSC(p)) gilt $A = B = \{0,1\}$. Elemente $x, y \in \{0,1\}$ erfüllen dann

$$P(\, y \text{ empfangen} \,|\, x \text{ gesendet}\,) = \begin{cases} 1-p & \text{falls } x = y \\ p & \text{falls } x \neq y \end{cases}$$

was häufig in Diagrammform dargestellt wird als



Im Falle $p = 0$ oder $p = 1$ arbeitet der Kanal zuverlässig (im zweiten Fall sind alle Ausgabesymbole einfach durch ihr Komplement zu ersetzen), während im Fall $p = 1/2$ Ein- und Ausgabe voneinander *statistisch unabhängig* sind.

Wegen des fehlenden Gedächtnisses erstreckt sich die statistische Unabhängigkeit auch auf die Behandlung aufeinanderfolgender Zeichen. Für Wörter $\boldsymbol{x}, \boldsymbol{y} \in A^m$ gilt

$$P(\, \boldsymbol{y} \text{ empfangen} \,|\, \boldsymbol{x} \text{ gesendet}\,) = \prod_{i<m} P(\, y_i \text{ empfangen} \,|\, x_i \text{ gesendet}\,)$$

0.1.02 Beispiel. Der BSC(p) läßt sich unmittelbar auf den Fall verallgemeinern, daß Ein- und Ausgabealphabet aus je q Symbolen bestehen; wir sprechen dann vom *q-nären symmetrischen Kanal*, oder kurz q-SC(p). Er ist charakterisiert durch

$$P(\, y \text{ empfangen} \,|\, x \text{ gesendet}\,) = \begin{cases} 1-p & \text{falls } x = y \\ \frac{p}{q-1} & \text{falls } x \neq y \end{cases}$$

Falls $|A| = |B|$ gilt wollen wir ohne Beschränkung der Allgemeinheit $A = B$ annehmen. Letztlich ist nur die Größe der Alphabete von Interesse, nicht die Beschaffenheit ihrer Elemente.

Es wird sich zudem als nützlich erweisen, A mit der Struktur einer *commutativen Gruppe* zu versehen. Dies ist unabhängig von der Größe der Menge A immer möglich. Jedes q-elementige Alphabet kann mit der Menge $\mathbb{Z}_q = \{\, n \in \mathbb{N} : n < q \,\}$ identitfiziert werden, die unter der Addition modulo q eine commutative Gruppe mit neutralem Element 0 ist. (Die Multiplikation modulo q liefert sogar eine Ringstruktur, was uns momentan noch nicht interessiert). Für jedes $n \in \mathbb{N}$ ist dann auch A^n eine commutative Gruppe (unter den komponentenweisen Operationen).

Der Vorteil dieser Betrachtungsweise besteht darin, daß wir jeden Kanal als *additiven Kanal* auffassen können, der dem Eingabewert bei der Übertragung einen *Fehler* hinzufügt. Bei Eingabe $\boldsymbol{x}$ und Ausgabe $\boldsymbol{y}$ ist dieser Fehler einfach als $\boldsymbol{e} := \boldsymbol{y} - \boldsymbol{x}$ definiert, was zu $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{e}$ äquivalent ist. Im allgemeinen kann der Fehler $\boldsymbol{e}$ von der Eingabe $\boldsymbol{x}$ abhängen, aber im Falle

des q-nären symmetrischen Kanals ist $\boldsymbol{e}$ statistisch von $\boldsymbol{x}$ unabhängig ("additives Rauschen"). Genauer: jede Komponente e_i, $i < n$, eines vom q-SC(p) erzeugten Fehlerworts $\boldsymbol{e}$ erfüllt

$$P(e_i = 0) = 1 - p \qquad \text{sowie} \qquad P(e_i = a) = \frac{p}{q-1} \quad \text{für } a \in \mathbb{Z}_q \setminus \{0\}$$

Unter den *Fehlerstellen* versteht man die vom *neutralen Element* 0 der Gruppe $A = \mathbb{Z}_q$ verschiedenen Komponenten des Vektors $\boldsymbol{e}$. Unsere Aufgabe ist es, diese Fehlerstellen zu lokalisieren und zu korrigieren.

0.2 Block-Codes

Wir fassen die oben beschriebenen Aspekte eines Codes begrifflich zusammen:

0.2.01 Definition. Unter einem *Block-Code* der *Codelänge* n über dem Alphabet $\mathbb{Z}_q$ verstehen wir eine nichtleere Teilmenge $\mathcal{C} \subseteq (\mathbb{Z}_q)^n$. Mit $M := |\mathcal{C}|$ bezeichnen wir die *Codegröße*, $k := \log_q M$ steht für die *(Code-)Dimension* oder auch *Informationslänge* und $R := k/n$ für die *Coderate*, während die Differenz $n - k$ als *Redundanz* bezeichnet wird. Die Elemente von $\mathcal{C}$ heißen *Codewörter*. Abkürzend sprechen wir von einem $(n, k)_q$-Code, wobei der Index q auch entfallen kann, falls die Alphabetgröße schon bekannt ist. (Manche Autoren bevorzugen es, von $(n, M)_q$-Codes zu sprechen, da die Codegröße M garantiert ganzzahlig ist.)

Um Unterschiede zwischen Codewörtern und auch zu anderen Wörtern aus A^n messen zu können, verwenden wir den *Hamming-Abstand* $A^n \times A^n \xrightarrow{\Delta} \mathbb{N}$:

$$\Delta_{\mathrm{H}}(\boldsymbol{x}, \boldsymbol{y}) \text{ ist die Anzahl der unterschiedlichen Positionen in } \boldsymbol{x} \text{ und } \boldsymbol{y}$$

Diese Funktion ist offensichtlich eine *Metrik*, denn sie genügt den Axiomen

(0) *Positivität*: $\Delta_{\mathrm{H}}(\boldsymbol{x}, \boldsymbol{y}) \geq 0$ mit Gleichheit genau dann wenn $\boldsymbol{x} = \boldsymbol{y}$ gilt;

(1) *Symmetrie*: $\Delta_{\mathrm{H}}(\boldsymbol{x}, \boldsymbol{y}) = \Delta_{\mathrm{H}}(\boldsymbol{y}, \boldsymbol{x})$;

(2) *Dreiecks-Ungleichung*: $\Delta_{\mathrm{H}}(\boldsymbol{x}, \boldsymbol{y}) + \Delta_{\mathrm{H}}(\boldsymbol{y}, \boldsymbol{z}) \geq \Delta_{\mathrm{H}}(\boldsymbol{x}, \boldsymbol{z})$.

Da es sich bei A um eine Gruppe handelt, können wir auch das *Hamming-Gewicht* des Vektors $\boldsymbol{e} \in A^n$ definieren vermöge

$$\omega(\boldsymbol{e}) \text{ ist die Anzahl der von } 0 \text{ verschiedenen } \boldsymbol{e}\text{-Komponenten.}$$

Offenbar gilt $\Delta_{\mathrm{H}}(\boldsymbol{x}, \boldsymbol{y}) = \omega(\boldsymbol{y} - \boldsymbol{x})$.

Schließlich erlaubt es der Hamming-Abstand, eine Aussage über die allgemeine Verteilung der Codewörter in A^n zu treffen, zumindest für $M > 1$. Dann definiert man den *Minimalabstand* des Codes $\mathcal{C} \subseteq A^n$ als

$$\operatorname{dist}(\mathcal{C}) := \min\{\, \Delta_{\mathrm{H}}(\boldsymbol{c}, \boldsymbol{c}') : \langle \boldsymbol{c}, \boldsymbol{c}' \rangle \in \mathcal{C}^2 \wedge \boldsymbol{c} \neq \boldsymbol{c}' \,\} \tag{0.2-03}$$

während der *relative Minimalabstand* gegeben ist durch

$$\delta := \frac{\operatorname{dist}(\mathcal{C})}{n}$$

Der Wert $d = \operatorname{dist}(\mathcal{C})$ wird häufig in die Beschreibung eines Block-Codes als dritter Parameter aufgenommen, d.h., wir sprechen dann von $(n, k, d)_q$-Codes. Allerdings erfaßt die Formel (0.2-03) nicht den, zugegeben pathologischen, Fall $M = 1$. In Kapitel 3 wird es sich als praktisch erweisen, einelementigen Codes der Länge n einen Minimalabstand von $n + 1$ zuzuweisen.

0.2.02 Beispiele.

(0) Der binäre $(3, 1, 3)$-*Wiederholungscode* $\{000, 111\}$ hat die Länge 3, die Größe 2, die Dimension $\log_2 2 = 1$ und die Rate $1/3$. Die beiden einzigen Codewörter unterscheiden sich in allen drei Positionen.

(1) Der binäre $(3, 2, 2)$-*Paritätscode* $\{000, 110, 101, 011\}$ hat ebenfalls die Länge 3, die Größe 4, die Dimension $\log_2 4 = 2$ und die Rate $2/3$. Je zwei verschiedene Codewörter unterscheiden sich in zwei Positionen. Wir bevorzugen gerade Parität, damit das Nullwort zum Code gehört. Das wird sich später als nützlich erweisen.

0.3 Decodierung

0.3.1 Definition eines Block-Decoders

Ausgehend von einem Block-Code $\mathcal{C}$ der Länge n betrachten wir bei der Decodierung nun Wahrscheinlichkeitsräume auf der Ereignismenge $A^n \times \mathcal{C}$ mit der σ-Algebra aller Teilmengen. Jeder Kanal $S = \langle A, B, P\rangle$ und jede Verteilung P' der Codewörter, induzieren darauf ein Wahrscheinlichkeitsmaß P'' vermöge

$$P''(\ \boldsymbol{y}\ \text{empfangen},\ \boldsymbol{c}\ \text{gesendet}\ ) = P(\ \boldsymbol{y}\ \text{empfangen}\,|\,\boldsymbol{c}\ \text{gesendet}\ )P'(\ \boldsymbol{c}\ \text{gesendet}\ )$$

Ist eine Teilmenge von $A^n \times \mathcal{C}$ mittels Komprehension definiert, etwa

$$X = \{\ \langle \boldsymbol{y}, \boldsymbol{c}\rangle \in A^n \times \mathcal{C} : \varphi\langle \boldsymbol{y}, \boldsymbol{c}\rangle\ \}$$

so schreibt man häufig $P''[\varphi\langle \boldsymbol{y}, \boldsymbol{c}\rangle]$ statt $P''(X)$ für die Wahrscheinlichkeit des Ereignisses X.

Die folgende Definition bezieht sich auf die erste Stufe des Kanaldecodierers in Diagramm 0.1-02.

0.3.01 Definition. Unter einem *Decoder bzgl. eines Kanals* S für einen Block-Code $\mathcal{C} \subseteq A^n$ verstehen wie eine Funktion $B^n \xrightarrow{D} \mathcal{C}$. Seine *Fehlerwahrscheinlichkeit* für das Codewort $\boldsymbol{c} \in \mathcal{C}$ ist gegeben durch

$$P_{\text{err}}(\boldsymbol{c}) := \sum_{D(\boldsymbol{y}) \neq \boldsymbol{c}} P(\ \boldsymbol{y}\ \text{empfangen}\,|\,\boldsymbol{c}\ \text{gesendet}\ )$$

Je nach Verteilung P' der Codewörter erhalten wir eine *Blockfehlerwahrscheinlichkeit* (oder kurz *BFW*) (vergleiche [3, (9.15)]) als Durchschnitt der Einzelfehlerwahrscheinlichkeiten:

$$P_{\mathrm{B}} := P''[D(\boldsymbol{y}) \neq \boldsymbol{c}] = \sum_{\boldsymbol{c}\in\mathcal{C}} P'(\ \boldsymbol{c}\ \text{gesendet}\ ) P_{\mathrm{err}}(\boldsymbol{c})$$

Um eine Abhängigkeit von der Verteilung P' der versandten Codewörter zu vermeiden, kann man stattdessen die *maximale Blockfehlerwahrscheinlichkeit* (*maximale BFW*) betrachten

$$P_{\mathrm{BM}} := \max_{\boldsymbol{c}\in\mathcal{C}} P_{\mathrm{err}}(\boldsymbol{c})$$

Die maximale BFW könnte von "schlecht decodierbaren" Codewörtern bestimmt werden, die man genau aus diesem Grunde nur selten versenden möchte. Um diesem Problem ebenso aus dem Wege zu gehen, wie der Abhängigkeit von der im Allgemeinen unbekannten Verteilung P' der Codewörter, sind Kanäle und Decoder vorzuziehen, für die $P_{\mathrm{err}}(\boldsymbol{c})$ möglichst wenig variiert. Zudem sollten die Codewörter nach Möglichkeit gleichverteilt sein.

0.3.02 Beispiel. Für den binären $(3,1,3)$-Wiederholungscode aus Beispiel 0.2.02(0) definieren wir bzgl. des BSC(p) einen Decoder D durch *Majoritätsentscheidung*. Dann gilt

$$\begin{aligned} P_{\mathrm{B}} = P_{\mathrm{BM}} = P_{\mathrm{err}}(000) = P_{\mathrm{err}}(111) &= \binom{3}{2} p^2(1-p) + \binom{3}{3} p^3 \\ &= 3p^2 - 2p^3 = p - (p - 3p^2 + 2p^3) = p + p(2p-1)(1-p) \end{aligned}$$

Im Falle von $p < 1/2$ ist der Ausdruck $(2p-1)$ negativ. Folglich verbessert sich der Übertragungsfehler verglichen mit p bei direkter Übertragung (Coderate 1), aber um den Preis der schlechteren Coderate von $1/3$.

0.3.2 Maximum-Likelihood Decodierung

Unter allen Deocodern nehmen sogenannte *Maximum-Likelihood Decoder* (MLD) eine ausgezeichnete Stellung ein. Für jedes feste $\boldsymbol{y} \in B^n$ maximiert das Codewort $D_{\mathrm{MLD}}(\boldsymbol{y}) \in \mathcal{C}$ die bedingte Wahrscheinlichkeit $P(\ \boldsymbol{y}\ \text{empfangen} \mid \boldsymbol{c}\ \text{gesendet}\ )$. Da ein solches Codewort nicht eindeutig bestimmt zu sein braucht, wählen wir ggf. das kleinste bzgl. der *lexikographischen Ordnung* auf $A^n = (\mathbb{Z}_q)^n$. Aber während MLDs theoretisch optimale Decodierung gewährleisten, scheinen bisher nur für triviale Code-Familien Algorithmen zu existieren, die eine Maximum-Likelihood-Decodierung in polynomialer Zeit umsetzen.

Im Gegensatz zu MLDs sind *Maximum á posteriori Decoder* (MpD) dadurch charakterisiert, daß sie für festes $\boldsymbol{y} \in A^n$ die bedingte Wahrscheinlichkeit $\bar{P}(\ \boldsymbol{c}\ \text{gesendet} \mid \boldsymbol{y}\ \text{empfangen}\ )$ maximieren. Dieser Wert kann allerdings nur berechnet werden, wenn die Verteilung P' der gesendeten Codewörter bekannt ist, denn nach dem Satz von Bayes gilt

$$\bar{P}(\ \boldsymbol{c}\ \text{gesendet} \mid \boldsymbol{y}\ \text{empfangen}\ ) = P(\ \boldsymbol{y}\ \text{empfangen} \mid \boldsymbol{c}\ \text{gesendet}\ ) \frac{P'(\ \boldsymbol{c}\ \text{gesendet}\ )}{\bar{P}'(\ \boldsymbol{y}\ \text{empfangen}\ )}$$

Sind die Codewörter gleichverteilt, also $P'(\boldsymbol{c}) = 1/M$ für alle $\boldsymbol{c} \in \mathcal{C}$, leisten der Maximum á posteriori Decoder und der Maximum-Likelihood Decoders dasselbe.

0.3.03 Beispiel. Wir betrachten einen (n, k, d)-Code bzgl. des BSC(p). Gemäß Beispiel 0.1.01 erhalten wir

$$P(\ \boldsymbol{y}\ \text{empfangen}\,|\,\boldsymbol{c}\ \text{gesendet}\ ) = p^{\Delta_{\mathrm{H}}(\boldsymbol{y},\boldsymbol{c})}(1-p)^{n-\Delta_{\mathrm{H}}(\boldsymbol{y},\boldsymbol{c})} = (1-p)^n \cdot \left(\frac{p}{1-p}\right)^{\Delta_{\mathrm{H}}(\boldsymbol{y},\boldsymbol{c})}$$

Für $p < 1/2$ gilt $p/(1-p) < 1$, folglich wird die Wahrscheinlichkeit genau dann maximiert, wenn der Hamming-Abstand zwischen (festem) empfangenen Wort $\boldsymbol{y}$ und (variablen) Codewort $\boldsymbol{c}$ minimal ist.

Dieses Beispiel legt eine weitere Decodierungsstrategie nahe. Ein *nächstes-Codewort Decoder* (nCD) ist eine Funktionen $A^n \xrightarrow{D} \mathcal{C}$, die für jeden additiven Kanal (der durch $A = B$ charakterisiert ist) ein Wort $\boldsymbol{y} \in A^n$ auf das bezüglich des Hamming-Abstands nächstliegende und bzgl. der lexikographischen Ordnung kleinste Codewort abbildet. Obiges Beispiel zeigt, daß ein MLD zumindest bzgl. des BSC(p) mit $p \neq 1/2$ dieses Kriterium erfüllt (für $p > 1/2$ geht man zum Komplementärwort $\boldsymbol{1} - \boldsymbol{y}$ über).

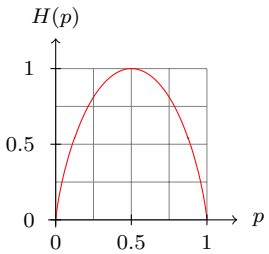
0.3.3 Kapazität des binären symmetrischen Kanals

In Beispiel 0.3.02 haben wir gesehen, daß Codierung die Fehlerwahrscheinlichkeit auf der Empfängerseite reduzieren kann, allerdings auf Kosten geringerer Coderaten. Wir werden uns jetzt davon überzeugen, daß beliebig kleine Werte von P_{err} erreicht werden können, ohne daß die Coderate gegen 0 streben muß.

Dazu betrachten wir zunächst die *binäre Entropie Funktion*

$$[0,1] \xrightarrow{H} [0,1] \quad , \quad p \mapsto -p \lg p - (1-p)\lg(1-p)\ ,$$

(wobei lg der Logarithmus zur Basis 2 ist) deren Graph folgende Form hat:



(0.3-04)

Die *Kapazität* des BSC(p) S ist dann gegeben durch

$$\mathrm{cap}(S) = 1 - H(p)$$

Die folgenden Sätze sind Spezialfälle grundlegender Ergebnisse der Informationstheorie. Sie besagen, daß die Kapazität eines Kanals die Coderaten nach oben beschränkt, mit denen Informationen zuverlässig durch diesen Kanal übertragen werden können.

0.3.04 Satz. *[Shannons Codierungs-Satz für den BSC] Für einen BSC(p)* S *und jede reelle Zahl* $R \in [0, \mathrm{cap}(S)[$ *existiert eine Folge binärer* (n_i, M_i)*-Block-Codes* $\mathcal{C}_i$, $i \in \mathbb{N}$, *mit* $\lg M_i/n_i \geq R$, *für die Fehlerwahrscheinlichkeiten* $(P_{\mathrm{err}})_i$ *bei Maximum-Likelihood Decodierung gegen* 0 *streben, für* $i \longrightarrow \infty$. □

0.3.05 Satz. *[Shannons umgekehrter Codierungs-Satz für den BSC] Gegeben sei ein BSC(p)* S *und eine reelle Zahl* $R > \mathrm{cap}(S)$. *Ist* $\mathcal{C}_i$, $i \in \mathbb{N}$ *eine Folge binärer* (n_i, M_i)*-Block-Codes mit* $\lg M_i/n_i \geq R$ *und streng monoton wachsenden Codelängen* n_i, *dann strebt für jedes Decodierungsverfahren bzgl.* S *die Fehlerwahrscheinlichkeit* $(P_{\mathrm{err}})_i$ *dieser Codes gegen* 1, *für* $i \longrightarrow \infty$. □

Die Beweise verschieben wir auf später (vielleicht). Man kann insbesondere zeigen, daß die Fehlerwahrscheinlichkeit $(P_{\mathrm{err}})_i$ in Satz 0.3.04 exponentiell mit wachsender Codelänge abnimmt. Andererseits handelt es sich zunächst nur um einen Existenzbeweis, der keine Konstruktion der Codefolge erlaubt.

Dennoch wollen wir ein Plausibilitätsargument präsentieren, warum im Falle eines BSC(p) die Coderate $\mathrm{cap}(S)$ nicht übersteigen kann, wenn $(P_{\mathrm{err}})_i$ gegen 0 strebt:

Nach korrekter Decodierung von $y \in B^n$ rekonstruiert der Empfänger

- das korrekte Codewort $\boldsymbol{c}$ unter M möglichen Codewörtern; dies entspricht $\lg M$ Informationbits;
- Das Fehlerwort $\boldsymbol{e} = \boldsymbol{y} - \boldsymbol{c}$.

Nach Definition des BSC sind diese beiden Daten statistisch unabhängig. Auch wenn der Empfänger am Fehlerwort $\boldsymbol{e}$ gar nicht interessiert ist, bekommt er es automatisch mitgeliefert. Das hat seinen Preis, der sich in der Coderate widerspiegelt.

Wir schätzen die durch $\boldsymbol{e}$ übertragenen Information ab. Von den 2^n möglichen Fehlerwörtern sind die meisten unwahrscheinlich. Nach dem Gesetz der Großen Zahlen (nach dem die relative Häufigkeit eines Zufallsereigniss bei vielfacher Wiederholung gegen seine Wahrscheinlichkeit konvergiert) wird das Fehlerwort ein Hamming-Gewicht im Bereich $n(p \pm \Delta)$ für kleines Δ haben. Ein derartiges *typisches* Fehlerwort tritt mit Wahrscheinlichkeit

$$p^{\omega(\boldsymbol{e})}(1-p)^{n-\omega(\boldsymbol{e})} = 2^{-n(H(p)+\varepsilon}$$

auf, wobei wegen der Stetigkeit $\varepsilon \longrightarrow 0$ sofern $\Delta \longrightarrow 0$. Damit liefert $2^{-nH(p)}$ eine Approximation für die Wahrscheinlichkeit des Auftretens eines typischen Fehlerworts. Also sollten $2^{nH(p)}$ typische Fehlerwörter mit ähnlicher Wahrscheinlichkeit existieren, weshalb das Fehlerwort eine Information von $nH(p)$ Bits überträgt.

Die vom Empfänger rekonstruierten $\lg M + nH(p)$ Informationsbits sind durch die n Bits des empfangenen Worts $\boldsymbol{y}$ beschränkt, also

$$\lg M + nH(p) \le n \quad \Longleftrightarrow \quad \frac{\lg M}{n} \le 1 - H(p) = \mathrm{cap}(S)$$

0.4 Varianten der Fehlerbehandlung

0.4.1 Fehlerkorrektur

Wir betrachten einen Kanal $S = \langle A, B, P\rangle$ mit $A = B$.

Für einen (n, k, d)-Code $\mathcal{C} \in A^n$ sei $\boldsymbol{c}$ das gesendete Codewort und $\boldsymbol{y}$ das empfangene Wort. Die Anzahl der aufgetretenen Fehler wird durch $\Delta_{\mathrm{H}}(\boldsymbol{y}, \boldsymbol{c})$ bestimmt. Ziel ist es, die Positionen und Werte dieser Fehler zu bestimmen.

0.4.01 Proposition. *Es gibt einen Decoder $A^n \xrightarrow{D} \mathcal{C}$ der jedes Fehlermuster von bis zu $(d-1)/2$ Fehlern für jeden Kanal $S = \langle A, A, P\rangle$ korrigiert.*

Beweis: Der nächstes-Codewort Decoder D leistet das Gewünschte. Wir nehmen $\Delta_{\mathrm{H}}(\boldsymbol{y}, \boldsymbol{c}) \le (d-1)/2$ und $\boldsymbol{c}' = D(\boldsymbol{y}) \ne \boldsymbol{c}$ an. Nach Definition von D gilt

$$\Delta_{\mathrm{H}}(\boldsymbol{y}, \boldsymbol{c}') \le \Delta_{\mathrm{H}}(\boldsymbol{y}, \boldsymbol{c}) \le (d-1)/2 \ ,$$

woraus wegen der Symmetrie und der Dreiecks-Ungleichung folgt

$$d \le \Delta_{\mathrm{H}}(\boldsymbol{c}, \boldsymbol{c}') \le \Delta_{\mathrm{H}}(\boldsymbol{y}, \boldsymbol{c}') + \Delta_{\mathrm{H}}(\boldsymbol{y}, \boldsymbol{c}) \le d - 1 \ ,$$

ein Widerspruch. □

Diese Ergebnis läßt sich auch geometrische interpretieren: die *Hamming-Kugeln* vom Radius $\tau = (d-1)/2$ um die Codewörter müssen nach Definition des Minimalabstands d paarweise disjunkt sein. Folglich wird ein empfangenes Wort, das zu einer dieser Kugeln gehört, vom nCD zu deren Zentrum decodiert. Dies stimmt genau dann mit dem gesendeten Codewort überein, wenn maximal τ Fehler aufgetreten sind. Wörter, die außerhalb aller dieser Hamming-Kugeln liegen, enthalten mehr als τ Fehler und können höchstens zufällig richtig decodiert werden.

0.4.02 Beispiel. Der binäre $(n, 2, n)$-Wiederholungscode enthält die Codewörter 0^n und 1^n. Ein nCD korrigiert jedes Muster aus bis zu $(n-1)/2$ Fehlern. Der Majoritäts-Decoder aus Beispiel 0.3.02 ist für ungerades n ein nächstes-Codewort Decoder, in diesem Fall können bis zu $(n-1)/2$ Fehler korrigiert werden. Für gerades n existieren Wörter in $\{0,1\}^n$, die zu 0^n und zu 1^n denselben Hammingabstand haben. Dann wird immer das lexikographisch kleinere Wort 0^n gewählt. Somit können nur bis zu $(n-2)/2$ Fehler korrigiert werden. Dies zeigt, daß Wiederholungscodes zweckmäßigerweise eine ungerade Länge haben sollten.

Proposition 0.4.01 kann nicht verbessert werden:

0.4.03 Proposition. *Zu jedem* (n,k,d)*-Code* $\mathcal{C}$ *über* A *und zu jedem Decoder* $A^n \xrightarrow{D} \mathcal{C}$ *existiert ein Codewort* $\boldsymbol{c} \in \mathcal{C}$ *und ein Wort* $\boldsymbol{y} \in A^n$ *mit* $\Delta_{\mathrm{H}}(\boldsymbol{y},\boldsymbol{c}) \leq (d+1)/2$ *und* $D(\boldsymbol{y}) \neq \boldsymbol{c}$*.*

Beweis: Wähle Codewörter $\boldsymbol{c}$, $\boldsymbol{c}'$ mit $\Delta_{\mathrm{H}}(\boldsymbol{c},\boldsymbol{c}') = d$ und definiere das Wort $\boldsymbol{y}$, indem ausgehend von $\boldsymbol{c}'$ die ersten $\lfloor (d+1)/2 \rfloor$ Komponenten, in denen sich $\boldsymbol{c}'$ von $\boldsymbol{c}$ unterscheidet, auf den Wert der entsprechenden Komponente von $\boldsymbol{c}$ gesetzt werden. Dann gilt

$$\Delta_{\mathrm{H}}(\boldsymbol{y},\boldsymbol{c}') = \lfloor (d+1)/2 \rfloor = \begin{cases} d/2 & \text{falls } d \text{ gerade} \\ (d+1)/2 & \text{falls } d \text{ ungerade} \end{cases}$$
$$\Delta_{\mathrm{H}}(\boldsymbol{y},\boldsymbol{c}) = \lceil (d-1)/2 \rceil = \begin{cases} d/2 & \text{falls } d \text{ gerade} \\ (d-1)/2 & \text{falls } d \text{ ungerade} \end{cases}$$

Nun enthält $\{\boldsymbol{c},\boldsymbol{c}'\} \setminus \{D(\boldsymbol{y})\}$ ein Codewort, das die geforderte Bedingung erfüllt. □

0.4.04 Beispiel. Der binäre Paritätscode aus Beispiel 0.2.02(1) erlaubt keine Fehlerkorrektur: beim Auftreten eines Fehlers kann $\boldsymbol{y} = 001$ von jedem der Codewörter 000, 101 und 011 herrühren.

Während Proposition 0.4.01 nicht von der bedingten Wahrscheinlichkeitsverteilung eines Kanals abhängt, wird beim Design eines Codierungs-Decodierungs Schemas für einen spezifischen Kanal S dessen Wahrscheinlichkeitsverteilung durchaus berücksichtigt. Sie dient zur Berechnung einer natürlichen Zahl τ, so daß die Wahrscheinlichkeit für τ Fehler die Anforderung an die Übertragung unterschreitet. Dann kann ein (n,k,d)-Code gewählt werden, der $d \geq 2\tau + 1$ erfüllt.

0.4.2 Fehlererkennung

Es dürfte keine Überraschung sein, daß mehr Fehler erkannt als korrigiert werden können. Allerdings ist Fehlererkennung nur dann von Interesse, wenn die Option zur Wiederholung der Übertragung von Blöcken besteht.

0.4.05 Proposition. *Ist* $\mathcal{C}$ *ein* (n,k,d)*-Code über* A*, so existiert ein Decoder* $A^n \xrightarrow{D} \mathcal{C}+1 = \mathcal{C} + \{\perp\}$*, der korrekt jedes Fehlermuster von bis zu* $d-1$ *Fehlern erkennt.*

Beweis: Setze

$$D(\boldsymbol{y}) = \begin{cases} \boldsymbol{y} & \text{if } \boldsymbol{y} \in \mathcal{C} \\ \perp & \text{otherwise} \end{cases}$$

Fehler werden nur dann nicht erkannt, wenn das empfangene Wort ein vom gesendeten verschiedenes Codewort ist. Aber dazu sind mindestens d Fehler nötig. □

0.4.06 Beispiel. Der binäre Paritätscode der Länge n besteht aus allen Wörtern in $2^n = \{0,1\}^n$, die eine gerade Anzahl von Einsen enthalten. Es handelt sich um einen $(n, n-1, 2)$-Code, der nur Einzelfehler erkennen kann.

Fehlerkorrektur und Fehlererkennung können auch simultan erfolgen.

0.4.07 Proposition. *Wir betrachten einen (n,k,d)-Code $\mathcal{C} \subseteq A^n$ und positive ganze Zahlen σ, τ mit*

$$2\tau + \sigma \leq d - 1$$

Dann existiert ein Decoder $A^n \xrightarrow{D} \mathcal{C} + 1$ mit folgenden Eigenschaften:

- *Maximal τ Fehler werden korrekt beseitigt;*
- *beträgt andernfalls die Fehlerzahl maximal $\tau + \sigma$, so werden diese Fehler korrekt erkannt.*

Beweis: Wir definieren den gesuchten Decoder wie folgt:

$$D(\boldsymbol{y}) = \begin{cases} \boldsymbol{c} & \text{if } \boldsymbol{c} \in \mathcal{C} \text{ mit } \Delta_{\mathrm{H}}(\boldsymbol{y}, \boldsymbol{c}) \leq \tau \\ \perp & \text{otherwise} \end{cases}$$

Liegt $\boldsymbol{y}$ in einer Hamming-Kugel vom Radius τ um ein Codewort $\boldsymbol{c}$, so wird $\boldsymbol{y}$ als $\boldsymbol{c}$ decodiert. Andernfalls liegt $\boldsymbol{y}$ außerhalb aller derartigen Kugeln und wird als $\perp$ decodiert. Bei $\leq \tau$ Fehlern ist die Decodierung korrekt.

Falls $\Delta_{\mathrm{H}}(\boldsymbol{y}, \boldsymbol{c}) \leq \tau + \sigma$ schlägt die Decodierung fehl, sofern $\boldsymbol{y}$ in einer Hamming-Kugel vom Radius τ um ein anderes Codewort $\boldsymbol{c}'$ liegt. Nach der Dreiecks-Ungleichung bedeutet das

$$d \leq \Delta_{\mathrm{H}}(\boldsymbol{c}, \boldsymbol{c}') \leq \Delta_{\mathrm{H}}(\boldsymbol{y}, \boldsymbol{c}) + \Delta_{\mathrm{H}}(\boldsymbol{y}, \boldsymbol{c}') \leq (\tau + \sigma) + \tau \leq d - 1 \ ,$$

ein Widerspruch. □

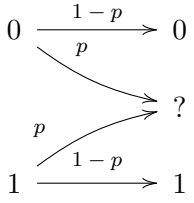
0.4.3 Löschungskorrektur

Bei einer Löschung wird ein Symbol durch den Kanal unkenntlich gemacht und kann dem Zielalphabet B nicht mehr zugeordnet werden – wir hatten ja vereinbart, daß der Kanal längenerhaltend arbeiten soll. Damit ist die Fehlerstelle bekannt, aber der Fehlerwert muß noch bestimmt werden.

Um dieses Phänomen besser mathematisch modellieren zu können, führen wir einen neuen Typ von Kanal ein.

0.4.08 Beispiel. Das folgende Diagramm repräsentiert einen gedächtnislosen *binären Löschungskanal* mit Eingabealphabet $\{0,1\}$ und Ausgabealphabet $\{0,1,?\}$. Ein Eingabesymbol wird mit Wahrscheinlichkeit

p gelöscht, unabhängig von vorangehenden oder nachfolgenden Symbolen.



Analog kann man einen *q-nären Löschungskanal* $\langle A, A+\{?\}, P\rangle$, mit $|A| = q$ definieren. Dann gilt immer noch

$$P(\,\boldsymbol{y}\ \text{empfangen}\,|\,\boldsymbol{x}\ \text{gesendet}\,) = \prod_{i<m} P(\,y_i\ \text{empfangen}\,|\,x_i\ \text{gesendet}\,)$$

für Wörter $\boldsymbol{x}, \boldsymbol{y} \in A^m$. Elemente $x, y \in A = \{0, 1\}$ erfüllen dann

$$P(\,y\ \text{empfangen}\,|\,x\ \text{gesendet}\,) = \begin{cases} 1-p & \text{falls } x = y \\ p & \text{falls } y = ? \\ 0 & \text{sonst} \end{cases}$$

Man kann zeigen, daß solch ein Kanal die Kapazität $1 - p$ hat; dies ist die maximale Coderate mit der Information durch den Kanal gesendet werden kann, wobei die Fehlerwahrscheinlichkeit gegen 0 konvergiert, wenn die Codelänge unbeschränkt wachsen darf.

Allgemein ist ein *Löschungkanal* $S = \langle A, B, P\rangle$ durch $B = A + \{?\}$ und die Bedingung charakterisiert, daß Symbole aus A nicht in andere Symbole aus A umgewandelt werden können:

$$P(\,\boldsymbol{y}\ \text{empfangen}\,|\,\boldsymbol{c}\ \text{gesendet}\,) > 0 \quad \text{nur für} \quad y_i \in \{c_i, ?\}\,,\ i < m$$

0.4.09 Proposition. *Wir betrachten einen (n, k, d)-Code $\boldsymbol{C} \subseteq A^n$ und $B := A + \{?\}$. Dann gibt es einen Decoder $B^n \xrightarrow{D} \boldsymbol{C} + \{\perp\}$, der für jeden Löschungskanal jedes Fehlermuster mit bis zu $d - 1$ Löschungen korrigiert.*

Beweis: Wir definieren den Decoder folgendermaßen:

$$D(\boldsymbol{y}) = \begin{cases} \boldsymbol{c} & \text{falls } \boldsymbol{y} \text{ mit genau einem } \boldsymbol{c} \in \mathcal{C} \text{ in den Komponenten aus } A \text{ übereinstimmt} \\ \perp & \text{sonst} \end{cases}$$

Falls $\boldsymbol{y}$ höchstens $d-1$ Löschungen aufweist, und verschiedene Codewörter sich in mindestens d Komponenten unterscheiden, kann höchstens ein Codewort in den nicht gelöschten Komponenten mit $\boldsymbol{y}$ übereinstimmen. Da $\boldsymbol{y}$ bei der Übertragung durch einen Löschungskanal entsteht, stimmt das übertragenen Codewort mit $\boldsymbol{y}$ in den nicht gelöschten Komponenten überein. □

Für einen allgemeinen Kanal $\langle A, A+\{?\}, P\rangle$, der Eingabesymbole aus A in beliebige Symbole aus $A+\{?\}$ umwandeln kann, können wir Fehlerkorrektur, Fehlererkennung und Löschungkorrektur gleichzeitig betrachten.

0.4.10 Proposition. *Wir betrachten einen (n,k,d)-Code $\boldsymbol{C} \subseteq A^n$ und $B := A+\{?\}$. Für jede natürliche Zahl $\rho < d$ von Löschungen wähle Zahlen τ_ρ und σ_ρ mit*

$$2\tau_\rho + \sigma_\rho + \rho \leq d-1$$

Dann gibt es einen Decoder $B^n \xrightarrow{D} \boldsymbol{C}+\{\perp\}$, mit folgenden Eigenschaften.

- *Beträgt die Fehlerzahl (einschließlich Löschungen) maximal τ_ρ, so werden alle Fehler korrigiert;*
- *beträgt andernfalls die Fehlerzahl maximal $\tau_\rho+\sigma_\rho$, dann werden diese Fehler korrekt erkannt.*

Beweis: Für ein empfangenes Wort $\boldsymbol{y}$ mit maximal ρ Löschungen sei $J \subseteq \{0,\ldots,n-1\}$ die Indexmenge der zu A gehörigen nicht gelöschten Komponenten. Für ein beliebiges Wort $\boldsymbol{x} \in A^n$ bezeichne $\boldsymbol{x}_J$ das durch J indizierte Teilwort. Wir setzen

$$\mathcal{C}_J := \{\, \boldsymbol{c}_J : \boldsymbol{c} \in \mathcal{C} \,\}$$

Der Minimalabstand des Codes $\mathcal{C}_J$ ist offenbar durch $d-\rho$ nach unten beschränkt. Wegen

$$2\tau_\rho + \sigma_\rho \leq (d-\rho) - 1$$

können wir den Decoder aus Proposition 0.4.07 auf $\mathcal{C}_J$ anwenden um das Wort $\boldsymbol{y}_J \in A^{n-\rho}$ zu decodieren. Erfüllt ein Codewort $\boldsymbol{c}_J \in \mathcal{C}_J$ die Bedingung $\Delta_{\mathrm{H}}(\boldsymbol{y}_J, \boldsymbol{C}_J) \leq \tau_\rho$, dann liefert dieser Decoder dieses dann eindeutig bestimmte Codewort, was eindeutig einem Codewort $\boldsymbol{c} \in \mathcal{C}$ entspricht. Andernfalls liefert der eingeschränkte Decoder $\perp$. □

1 Lineare Codes

Wir betrachten nun den speziellen Fall, daß das Codealphabet nicht nur eine Gruppe sondern sogar einen Körper bildet, weshalb es im Folgenden mit F statt A bezeichnet wird. Insbesondere interessieren uns diejenigen Codes $\mathcal{C} \in F^n$, die einen *Untervektorraum* von F^n bilden. Solche *linearen Codes* können mit Hilfe zweier Matrizen beschrieben werden: die *Generatormatrix* repräsentiert den Code auf kompakte Weise und dient gleichzeitig zur effizienten Codierung. Die *Kontrollmatrix* dient der Codeanalyse und kommt bei der Decodierung zum Einsatz.

Wiederholungs- wie auch Paritätscodes und auch die sogenannten Hamming-Codes gehören zur Klasse der linearen Codes, die ihrer einfachen Struktur wegen die größte Klasse praktisch genutzter Block-Codes bilden.

1.0 Definition

Mit $\mathrm{GF}(q)$ bezeichnen wir den endlichen (*Galois-*)Körper mit q Elementen. Ist q eine Primzahl, so stimmt $\mathrm{GF}(q)$ mit dem Ring $\mathbb{Z}_q$ der ganzzahligen Reste modulo q überein. Wir werden später sehen, daß genau für jede Primzahlpotenz q ein solcher endlicher Körper existiert.

Ein (n,k,d)-Code $\mathcal{C}$ über einem Körper $F = \mathrm{GF}(q)$ heißt *linear*, falls $\mathcal{C}$ ein *Untervektorraum* von F^n ist. Dies ist genau dann der Fall, wenn jede binäre *Linearkombination* von Codewörtern wieder ein Codewort ist, d.h.,

$$a_0 \boldsymbol{c}_0 + a_1 \boldsymbol{c}_1 \in \mathcal{C} \quad \text{für alle} \quad \boldsymbol{c}_0, \boldsymbol{c}_1 \in \mathcal{C} \quad \text{und alle} \quad a_0, a_1 \in F$$

Äquivalent ist $\mathcal{C}$ unter *skalaren Vielfachen* (speziell $\boldsymbol{0}$) und Summen abgeschlossen.

Die Dimension eines linearen Codes $\mathcal{C}$ gemäß Definition 0.2.01 stimmt mit der Vektorraum-Dimension von $\mathcal{C}$ als Untervektorraum von F^n überein; im Falle von $k = 0$ handelt es sich um den trivialen linearen Code $\{\boldsymbol{0}\} \subseteq F^n$. Wir sprechen im linearen Fall auch von einem $[n,k,d]_q$-Code (man beachte die eckigen Klammern), wobei wie zuvor die Spezifikation des Minimalabstands auch fehlen kann.

Jede Basis eines linearen $[n,k,d]$-Codes über $\mathrm{GF}(q)$ besteht aus k linear unabhängigen Codewörtern. Deren q^k Linearkombinationen sind alle verschieden und erzeugen den ganzen Code. Folglich gilt $|\mathcal{C}| = M = q^k$ und die Coderate hat den Wert $R = (\log_q M)/n = k/n$.

Zur Unterscheidung vom allgemeinen Fall werden wir bei linearen Codes die Codewörter in runde Klammern einschließen, was sie leichter als *Vektoren* kenntlich macht.

Für lineare Codes läßt sich der Minimalabstand besonders einfach bestimmen:

1.0.01 Proposition. *Für einen linearen $[n,k,d]$-Code $\mathcal{C}$ über F gilt*

$$d = \begin{cases} \min\{\,\omega(\boldsymbol{c}) : \boldsymbol{c} \in \mathcal{C} \setminus \{\boldsymbol{0}\}\,\} & \text{falls } k > 0, \\ n+1 & \text{falls } k = 0. \end{cases}$$

Beweis: Wegen der Linearität ist $\mathcal{C}$ auch unter Differenzbildung abgeschlossen. Damit folgt die Behauptung für $k > 0$ sofort aus $\Delta_{\mathrm{H}}(\boldsymbol{c}, \boldsymbol{c}') = \omega(\boldsymbol{c}' - \boldsymbol{c})$. □

1.0.02 Beispiel. Der $(3,2,2)$-Paritätscode $\mathcal{C}_{\mathrm{p}} = \{(0\,0\,0), (0\,1\,1), (1\,0\,1), (1\,1\,0)\}$ über $\{0,1\} = \mathrm{GF}(2)$ aus Beispiel 0.2.02(1) kann spezifiziert werden durch

$$\boldsymbol{c} \in \mathcal{C}_{\mathrm{p}} \mathrel{:\Longleftrightarrow} c_2 = c_0 + c_1$$

Die Abgeschlossenheit von $\mathcal{C}_{\mathrm{p}}$ unter Linearkombinationen folgt unmittelbar daraus, daß die Koordinate c_2 eine Linearkombination der ersten beiden Koordinaten ist. Wegen $k = 2$ handelt es sich bei $\mathcal{C}_{\mathrm{p}}$ um einen linearen $[3,2,2]$-Code. Eine Basis ist beispielsweise durch die Codewörter

$(1\,0\,1)$ und $(0\,1\,1)$ gegeben, deren Linearkombinationen spannen den gesamten Code auf:

$$\begin{aligned}(0\,0\,0) &= 0\cdot(1\,0\,1) + 0\cdot(0\,1\,1)\\ (0\,1\,1) &= 0\cdot(1\,0\,1) + 1\cdot(0\,1\,1)\\ (1\,0\,1) &= 1\cdot(1\,0\,1) + 0\cdot(0\,1\,1)\\ (1\,1\,0) &= 1\cdot(1\,0\,1) + 1\cdot(0\,1\,1)\end{aligned}$$

Unter einer *Generatormatrix* eines linearen $[n,k,d]$-Codes $\mathcal{C}$ über F verstehen wir eine $(k\times n)$-Matrix G, deren Zeilen eine Basis des Codes bilden. Insbesondere stimmt der *Rang* von G mit der Dimension k von $\mathcal{C}$ überein. Solch eine Matrix G ist i.A. nicht eindeutig bestimmt, insofern kann man je nach Problemstellung die günstigste Variante wählen.

1.0.03 Beispiel. Gemäß Beispiel 1.0.02 ist

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

eine Generatormatrix für den $[3,2,2]$-Paritätscode über $\mathrm{GF}(2)$. Das gilt ebenso für

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

Allgemein kann man den $[n,n-1,2]$-*Paritätscode* über einem beliebigen endlichen Körper F als einen Code mit Generatormatrix

$$G_{\mathrm{p}} = \left(\begin{array}{c|c} & -1\\ I_{n-1} & \vdots\\ & -1\end{array}\right)$$

definieren, wobei I_{n-1} eine Einheitsmatrix der Dimension $n-1$ ist. Da sich die Komponenten jeder Zeile zu 0 summieren, gilt dies auch für jede Linearkombination und folglich für jedes Codewort. Mit anderen Worten, die Komponentensumme ist eine lineare Abbildung $F^n \longrightarrow F$, deren Kern der entsprechende Paritätscode ist. Aus dieser Charakterisierung folgt leicht, daß der Minimalabstand tatsächlich den Wert 2 haben muß: man braucht nur die Hamming-Gewichte der von $\boldsymbol{0}$ verschiedenen Codewörter zu betrachten.

1.0.04 Beispiel. Der $(3,1,3)$-Wiederholungscode über $\mathrm{GF}(2)$ ist ein linearer $[3,1,3]$-Code, der von

$$G_{\mathrm{w}} = \begin{pmatrix} 1 & 1 & 1\end{pmatrix}$$

erzeugt wird. Allgemein hat der $[n,1,n]$-Wiederholungscode eine $(1\times n)$-Generatormatrix aus n Einsen.

1.1 Codierung für lineare Codes

Ist $\mathcal{C}$ ein k-dimensionaler Unterraum von F^n, so ist er zu F^k isomorph. Da es genausoviele Informationswörter wie Codewörter gibt, können wir die Elemente von F^k als Informationswörter auffassen und die durch eine Generatormatrix G für $\mathcal{C}$ bestimmte lineare Abbildung

$$F^k \xrightarrow{g} F^n \quad , \quad \boldsymbol{u} \mapsto \boldsymbol{u} \cdot G$$

als *Codierung* verwenden. Da G den Rang k hat, ist diese lineare Abbildung injektiv. Folglich ist $\mathcal{C}$ das injektive *Bild* des Vektorraums F^k der Informationswörter im Vektorraum F^n unter einer *Einbettung*.

Daher können wir G mit Hilfe elementarer Zeilenoperationen umformen, bis eine Einheitsmatrix der Dimension k als Teilmatrix auftritt. Speziell heißt eine $(k\times n)$-Generatormatrix *systematisch*, falls sie die Form

$$\left(\; I_k \;\middle|\; A \; \right)$$

hat, wobei I_k eine $(k\times k)$-Einheitsmatrix ist. Eine Generatormatrix kann genau mittels Zeilenoperationen in diese Form gebracht werden, wenn ihre ersten k Spalten linear unabhängig sind. Ist dies für $\mathcal{C}$ *nicht* der Fall, so können wir zumindest die Koordinaten von $\mathcal{C}$ (und folglich die Spalten von G) derart permutieren, daß wir einen *äquivalenten* (wenn auch verschiedenen) Code $\mathcal{C}'$ erhalten, der diese Bedingung erfüllt.

Im Falle einer systematischen Generatormatrix $G = (I_k \,|\, A)$ liefert die Codierung $\boldsymbol{u} \mapsto \boldsymbol{u} \cdot G = (\boldsymbol{u} \,|\, \boldsymbol{u} \cdot A)$, d.h., das Codewort entsteht aus dem Informationswort durch Anhängen zusätzlicher Elemente aus F. In diesem Falle wird die Decodierung besonders einfach sein.

1.2 Die Kontrollmatrix und die Konstruktion neuer linearer Codes

Achtung: Dieser Abschnitt unterscheidet sich von der konventionellen Literatur insofern, als wir konsistent mit Zeilenvektoren arbeiten.

Unter einer *Kontrollmatrix* für einen linearen $[n, k, d]$-Code $\mathcal{C}$ über F versteht man eine $(n \times r)$-Matrix H mit folgender Eigenschaft

$$\boldsymbol{c} \in \mathcal{C} \quad \text{genau dann wenn} \quad \boldsymbol{c} \cdot H = \boldsymbol{0}$$

Mit anderen Worten, $\mathcal{C}$ ist der *Kern* der durch H spezifizierten linearen Abbildung $F^n \xrightarrow{h} F^r$. Folglich stimmt der Rang von H mit $n - \dim\ker(H) = n - k$ überein, d.h., H hat $n-k$ linear unabhängige Spalten. Der Minimalwert von r ist somit durch die Redundanz von $\mathcal{C}$ gegeben, in diesem Fall sind alle Spalten von H linear unabhängig.

Der folgende Satz charakterisiert den Minimalabstand eines linearen Codes mit Hilfe der Kontrollmatrix.

1.2.01 Satz. *Ist H die Kontrollmatrix eines linearen Codes $\mathcal{C} \neq \{\boldsymbol{0}\}$. Der Minimalabstand d von $\mathcal{C}$ stimmt mit der kleinsten Zahl z linear abhängiger Zeilen von H überein.*

Beweis: Für jedes Codewort $\boldsymbol{c} \in \mathcal{C}$ verschwindet die Linearkombination $\boldsymbol{c} \cdot H$, folglich sind die von $\boldsymbol{c}$-Komponenten ungleich 0 ausgewählten Zeilen von H linear abhängig. Nach Proposition 1.0.01 existieren somit mindestens d linear abhängige Zeilen, also $z \leq d$

Umgekehrt existiert zu einer Auswahl von z linear abhängigen Zeilen eine nichttriviale Linearkombination mit Wert $\boldsymbol{0}$, wodurch ein Codewort $\boldsymbol{c} \neq \boldsymbol{0}$ mit $d \leq \omega(\boldsymbol{c}) \leq z$ bestimmt wird (vergl. Proposition 1.0.01). □

Da die Zeilen einer Generatormatrix G für $\mathcal{C}$ eine Basis für $\mathcal{C}$ bilden und somit selber Codewörter sind, folgt nach Definition der Kontrollmatrix sofort

$$G \cdot H = \boldsymbol{0}$$

und jede Matrix H mit dieser Eigenschaft ist offenbar eine Kontrollmatrix. Im speziellen Fall einer systematischen Generatormatrix $G = (I_k \,|\, A)$ erfüllt jedes Codewort $\boldsymbol{c} \in \mathcal{C}$

$$\begin{aligned} \boldsymbol{c} = (c_0 \,\ldots\, c_{k-1}) \cdot (I_k \,|\, A) &\iff (c_0 \,\ldots\, c_{k-1}) \cdot A = (c_k \,\ldots\, c_{n-1}) \\ &\iff -(c_0 \,\ldots\, c_{k-1}) \cdot A + (c_k \,\ldots\, c_{n-1}) = \boldsymbol{0} \\ &\iff \boldsymbol{c} \cdot \begin{pmatrix} -A \\ I_r \end{pmatrix} = \boldsymbol{0} \end{aligned}$$

Daher können wir

$$H := \begin{pmatrix} -A \\ I_r \end{pmatrix}$$

als *systematische* Kontrollmatrix wählen. Im allgemeinen Fall einer nicht notwendig systematischen Generatormatrix G permutieren wir zunächst die Spalten geeignet, etwa durch φ, bestimmen mittels elementarer Zeilenoperationen eine äquivalente systematische Generatormatrix $G' = (I_k \,|\, A)$, bilden nach obigem Rezept eine systematische Kontrollmatrix H' und vertauschen deren Zeilen gemäß der inversen Permutation φ^{-1}. Dies liefert eine Kontrollmatrix für $\mathcal{C}$.

1.2.02 Beispiel. Gemäß der Beispiele 1.0.03 und 1.0.04 erhalten wir folgende systematische Kontrollmatrizen für den $[n, n-1, 2]$-Paritätscode und für den $[n, 1, n]$-Wiederholungscode über F

$$H_{\mathrm{p}} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \qquad \text{bzw.} \qquad H_{\mathrm{w}} = \left(\begin{array}{ccc} -1 & \ldots & -1 \\ \hline & I_{n-1} & \end{array}\right)$$

Die Spalten von H_{w} entsprechen den Tests, ob jede der ersten $n-1$ Komponenten mit der letzten Komponente übereinstimmt. Alternativ kann man natürlich auch die erste mit den letzten $n-1$ Komponenten vergleichen, was folgende Kontrollmatrix liefert:

$$H'_{\mathrm{w}} = \left(\begin{array}{ccc} & I_{n-1} & \\ \hline -1 & \ldots & -1 \end{array}\right)$$

Zwar ist H'_{w} nicht mehr systematisch, dafür hat sie die interessante Eigenschaft, daß die transponierte Matrix $(H'_{\mathrm{w}})^\top$ mit G_{p} aus Beispiel 1.0.03 übereinstimmt. Analog liefert die Transposition von H_{p} (bei der eine Zeilenvertauschung keinen Effekt hat) die Generatormatrix G_{w} aus Beispiel 1.0.04.

Der oben festgestellte Zusammenhang zwischen Wiederholungs- und Paritätscode läßt sich verallgemeinern: ist $\mathcal{C}$ ein $[n,k,d]$-Code über F, so besteht der *duale Code* $\mathcal{C}^\perp$ aus genau den Vektoren $\boldsymbol{x} \in F^n$, die $\boldsymbol{c} \cdot \boldsymbol{x}^\top = \boldsymbol{0}$ für jedes $\boldsymbol{c} \in \mathcal{C}$ erfüllen. In Analogie zur Situation für Vektorräume über $\mathbb{R}$ nennt man die Elemente von $\mathcal{C}^\perp$ auch *orthogonal* zu $\mathcal{C}$; man sollte diese Terminologie allerdings nicht überbewerten, denn im Falle eines endlichen Körpers kann ein von $\boldsymbol{0}$ verschiedener Vektor selbstorthogonal sein. Alternativ läßt sich $\mathcal{C}^\perp$ charakterisieren durch

$$\boldsymbol{x} \in \mathcal{C}^\perp \quad \text{genau dann wenn} \quad \boldsymbol{x} \cdot G^\top = \boldsymbol{0}$$

wobei G eine Generatormatrix für $\mathcal{C}$ ist. Also ist $\mathcal{C}^\perp$ ein $[n, n-k, d^\perp]$-Code mit Kontrollmatrix $G^\top$. Analog liefert eine Kontrollmatrix H für $\mathcal{C}$ nach Transposition eine Generatormatrix für $\mathcal{C}^\perp$. Daher gilt $\left(\mathcal{C}^\perp\right)^\perp = \mathcal{C}$ und $\langle \mathcal{C}, \mathcal{C}^\perp \rangle$ wird als *duales Paar* bezeichnet.

Nach Beispiel 1.2.02 bilden der $[n,1,n]$-Wiederholungscode und der $[n, n-1, 2]$-Paritätscode ein derartiges duales Paar.

1.2.03 Beispiel. Der lineare $[7,4,3]$-*Hamming-Code* über GF(2) ist durch die Kontrollmatrizen

$$H = \begin{pmatrix} 0&0&1\\ 0&1&0\\ 0&1&1\\ 1&0&0\\ 1&0&1\\ 1&1&0\\ 1&1&1 \end{pmatrix} \quad \text{bzw.} \quad H' = \begin{pmatrix} 0&1&1\\ 1&0&1\\ 1&1&0\\ 1&1&1\\ 1&0&0\\ 0&1&0\\ 0&0&1 \end{pmatrix} \quad \text{(nach Spaltenoperationen)}$$

definiert, wobei in H die von $\boldsymbol{0}$ verschiedenen Vektoren aus F^3 in lexikographischer Reihenfolge als Zeilen auftreten, während H' systematisch ist. Aus der systematischen Generatormatrix G' (gemäß Abschnitt 1.1) erhält man durch Zeilenoperationen eine zweite sehr regelmäßige Generatormatrix G, nämlich $H^\top$ ergänzt um eine erste Zeile von Einsen.

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

Aus G' läßt sich direkt ablesen, daß Informationswörter mit Hamming-Gewicht 1 oder 2 Codewörter mit Hamming-Gewicht ≥ 3 liefern, was später zur Definiton eines allgemeinen Hamming-Codes führen wird.

1.2.04 Beispiel. Der zum linearen $[7,4,3]$-Hamming-Code duale $[7,3,4]$ *Simplex-Code* hat z.B. die Generator- bzw. Kontrollmatrix

$$G^\perp = H'^\top = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \qquad \text{sowie} \qquad H^\perp = G'^\top = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix}$$

Aus Symmetriegründen ist der Nachweis der linearen Unabhängigkeit je dreier Zeilen der Matrix $H^\perp$ noch handhabbar: man betrachte t der oberen vier und s der unteren drei Zeilen, mit $t+s=3$. Weiterhin gibt es vier linear abhängige Zeilen, etwa die mit führender Null. Also hat dieser Code tatsächlich Minimalabstand 4.

1.2.05 Beispiel. Ein linearer $[8,4,4]$ *erweiterter Hamming-Code* $\bar{\mathcal{C}}$ über GF(2) entsteht aus dem $[7,4,3]$ Hamming-Code $\mathcal{C}$ durch Anfügen eines Paritätsbits. Á priori kann dies an beliebiger Stelle erfolgen, aber im Falle systematischer Codierung ist sicherlich das Ende vorzuziehen. Dies liefert Generator- und Kontrollmatrizen

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

Fügt man ausgehend von G und H aus Beispiel 1.2.03 das Paritätsbit vorne hinzu, erhält man „hübschere“ Matrizen, die Transponate voneinander sind:

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

Die erste Spalte von $\bar{H}$ dient der Überprüfung der Parität, während die restlichen Nullen der ersten Zeile das Paritätsbit für die übrigen Tests der Matrix H ignorieren.

Wegen $H^\perp = G$ folgt $\bar{\mathcal{C}} = \bar{\mathcal{C}}^\perp$, damit ist die zweite Variante des erweiterten Hamming-Codes *selbstdual*. Dasselbe Argument funktioniert, wenn das Paritätsbit an anderer Stelle eingefügt wird, und damit auch für die erste Variante dieses Codes (warum?).

Die folgenden Beispiele verallgemeinern die Konstruktionen aus den obigen beiden Beispielen 1.2.03 und 1.2.05.

1.2.06 Beispiel. Für $m > 1$ ist ein $[2^m - 1, 2^m - 1 - m, 3]$ *Hamming-Code* über $F = \mathrm{GF}(2)$ durch eine Kontrollmatrix H definiert, deren Zeilen die Vektoren aus $F^m \setminus \{\boldsymbol{0}\}$ durchlaufen. Je zwei Zeilen sind verschieden und somit über $\mathrm{GF}(2)$ linear unabhängig. Andererseits sind die Zeilen, die der Binärdarstellung von 1, 2 und 3 entsprechen linear abhängig. Folglich hat der Minimalabstand den Wert 3.

1.2.07 Beispiel. Der $[2^m-1, 2^m-1-m, 3]$ Hamming-Code aus Beispiel 1.2.05 mit Kontrollmatrix H liefert nach Anfügen eines Paritätsbits einen $[2^m, 2^m-1-m, 4]$ erweiterten Hamming-Code. Gemäß Hausaufgabe 2 vergrößert dies den Minimalabstand um 1. Eine Kontrollmatrix $\bar{H}$ kann aus H durch Einfügen einer Null-Zeile an der durch das neue Paritätsbit bestimmten Stelle und anschließende Ergänzung durch eine Einser-Spalte gewonnen werden.

Die obigen Konstruktionen sind nicht auf den binären Fall beschränkt.

1.2.08 Beispiel. Für $F = \mathrm{GF}(q)$ und $m > 1$ setzen wir $n := (q^m - 1)/(q - 1)$. Dann wird ein $[n, n-m, 3]$ *Hamming-Code* über $\mathrm{GF}(q)$ mittels einer $(m \times n)$-Kontrollmatrix definiert, deren Zeilen aus allen Vektoren aus $F^m \setminus \{\boldsymbol{0}\}$ besteht, deren erste von 0 verschiedene Komponente den Wert 1 hat. Dann sind je zwei Zeilen wieder linear unabhängig, während drei linear abhängige Zeilen existieren (dieselben wie im binären Fall), was einen Minimalabstand von 3 garantiert.

Neben den in den obigen Beispielen informell beschriebenen Konstruktionen neuer linearer Codes aus gegebenen lineare Codes gibt es noch einige andere. Wir fassen abschließend die gängigsten derartigen Konstruktionen zusammen.

1.2.09 Definition. Für einen k-dimensionalen linearen Code $\mathcal{C} \leq F^n$ definieren wir

(0) den zu $\mathcal{C}$ *dualen Code* $\mathcal{C}^\perp := \{\, \boldsymbol{x} \in F^n : \forall \boldsymbol{c} \in \mathcal{C}.\, \boldsymbol{c} \cdot \boldsymbol{x}^\top = 0 \,\} \leq F^n$;

(1) die *Erweiterung* $\bar{\mathcal{C}} := \{\, (\boldsymbol{c} \mid -\sum_{i<n} c_i) : \boldsymbol{c} \in \mathcal{C} \,\} \leq F^{n+1}$;

(2) die *Punktierung* $\dot{\mathcal{C}} := \{\, \boldsymbol{y} \in F^{n-1} : \exists x \in F.\, (\boldsymbol{y} \mid x) \in \mathcal{C} \,\} \leq F^{n-1}$ falls $n \geq 1$;

(3) die *i-Verkürzung* $\mathcal{C}^{(i)} := \{\, (\boldsymbol{y} \mid \boldsymbol{z}) \in F^{i-1} \times F^{n-1} : (\boldsymbol{y} \mid 0 \mid \boldsymbol{z}) \in \mathcal{C} \,\} \leq F^{n-1}$ falls $i < n$;

(4) die *Augmentierung* $\mathcal{C}^{\mathrm{aug}} := \langle\!\langle \mathcal{C} \cup \{\langle 1, 1, \ldots, 1\rangle\} \rangle\!\rangle \leq F^n$;

(5) die *Spiegelung* $sp(\mathcal{C}) := \{\, (\, x_i : i < n \,) : (\, x_{n-i-1} : i < n \,) \in \mathcal{C} \,\}$;

(6) die *Ausdünnung* $\mathcal{C}^{\mathrm{par}} := \{\, \boldsymbol{c} \in \mathcal{C} : \sum_{i<n} c_i = 0\} \leq F^n$.

1.2.10 Proposition. *Hat ein k-dimensionaler linearer Code $\mathcal{C} \leq F^n$ eine Generatormatrix G und eine Kontrollmatrix H, so gilt*

(0) $\mathcal{C} = \mathcal{C}^{\perp\perp}$ *und* $H^\top$ *bzw.* $G^\top$ *sind Generator- bzw. Kontrollmatrix für* $\mathcal{C}^\perp$ *;*

(1)

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

sind Generator- bzw. Kontrollmatrix für $\bar{\mathcal{C}}$ *; ist* G *systematisch, so gilt dies auch für* $\bar{G}$ *, während für systematisches* H *die Addition der ersten* r *Spalten zur letzten* $\bar{H}$ *in eine systematische Matrix umwandelt;*

(2) *die Spiegelung aller Zeilen von* G *sowie aller Spalten von* H *liefert eine Generator- bzw, Kontrollmatrix für* $sp(\mathcal{C})$ *.* □

1.2.11 Proposition. $\mathcal{C}$ *sei ein* $[n, k, d]$*-Code.*

(0) *Für* $i < n$ *ist die* i*-Verkürzung* $\mathcal{C}^{(i)}$ *ein linearer* $[n-1, k^{(i)}, d^{(i)}]$ *Code mit* $k^{(i)} \in \{k-1, k\}$ *und* $d^{(i)} \geq d$ *.*

(1) *Entfernen der Zeile* i *aus einer Kontrollmatrix für* $\mathcal{C}$ *liefert eine Kontrollmatrix für* $\mathcal{C}^{(i)}$ *.*

Beweis:

(0) Die Projektion $F^n \xrightarrow{\pi_i} F$ auf die Koordinate i ist linear. Somit ergibt sich $\mathcal{C}^{(i)}$ als Bild des Unterraums $\mathcal{C} \cap \ker(\pi_i)$ unter der komplementären Projektion $F^n \xrightarrow{\bar{\pi}_i} F^{n-1}$, welche die Koordinate i vergißt. Insbesondere ist $\mathcal{C}^{(i)}$ ein UVR von F^{n-1} .

Betrachtet man oBdA eine systematische Codierung, so kann die Koordinate i im Bereich der ersten k Bits (Info-Bits) liegen, oder im Bereich der letzten $n-k$ Kontrollbits. Im ersten Fall verringert sich die Code-Dimension um 1 , im zweiten Fall nicht.

Sofern ein Codewort $\boldsymbol{c}$ minimalen Hamming-Gewichts mit $c_i = 0$ existiert, enthält auch $\mathcal{C}^{(i)}$ ein Wort dieses Gewichts. Falls hingegen jedes Codewort $\boldsymbol{c}$ minimalen Hamming-Gewichts die Bedingung $c_i = 1$ erfüllt, folgt für ein Codewort $\boldsymbol{c'} \neq \boldsymbol{0}$ aus $c'_i = 0$ sofort $\omega(\boldsymbol{c'}) > d$, und damit $d^{(i)} > d$.

(1) Ist H eine $(n \times r)$-Kontrollmatrix für $\mathcal{C}$, so gilt $\boldsymbol{c} \in \mathcal{C}$ genau dann wenn $\boldsymbol{c} \cdot H = \boldsymbol{0}$. Daraus folgt unmittelbar

$$\boldsymbol{c} \in \mathcal{C} \cap \ker(\pi_i) \quad \text{genau dann wenn} \quad \boldsymbol{c} \cdot H(i, \boldsymbol{a}) = \boldsymbol{0} \quad \text{für alle } \boldsymbol{a} \in F^r$$

wobei $H(i, \boldsymbol{a})$ entsteht indem die i-te Zeile durch $\boldsymbol{a}$ ersetzt wird. Damit erhalten wir

$$\boldsymbol{c}^{(i)} \in \mathcal{C}^{(i)} \quad \text{genau dann wenn} \quad \boldsymbol{c}^{(i)} \cdot H^{(i)} = \boldsymbol{0}$$

wobei $\boldsymbol{c}^{(i)}$ durch Entfernen der i-ten Komponente und $H^{(i)}$ durch entfernen der i-ten Zeile entsteht. □

1.2.12 Bemerkung. Im binären Fall ist die Erweiterung eines linearen $[n, k, d]$-Codes mit ungeradem Minimalabstand ein linearer $[n+1, k, d+1]$-Code (vergl. Hausaufgabe 2), denn jedes Codewort mit minimalem Hamming-Gewicht hat eine von 0 verschiedene Komponentensumme. Für $F = \mathrm{GF}(q)$ mit $q > 2$ können dagegen auch bei ungeradem Minimalabstand Codewörter mit minimalem Hamming-Gewicht und Komponentensumme 0 existieren. Dann vergrößert sich der Minimalabstand der Erweiterung nicht.

1.3 Decodierung linearer Codes

Wir erinnern an Beispiel 0.3.03, was leicht auf den q-nären symmetrischen Kanal mit Umwandlungswahrscheilichk $p < 1 - (1/q)$ verallgemeinert werden kann: für einen linearen $[n, k, d]$-Code $\mathcal{C}$ über F stimmt die Maximum Likelihood Decodierung mit der Decodierung zum nächsten Codewort überein. Folglich gilt es für ein empfangenes Wort $\boldsymbol{y} \in F^n$

- ein Codewort $\boldsymbol{c} \in \mathcal{C}$ zu finden, das $\Delta_{\mathrm{H}}(\boldsymbol{y}, \boldsymbol{c})$ minimiert, oder äquivalent,
- ein Fehlerwort $\boldsymbol{e} \in F^n$ minimalen Hamming-Gewichts zu finden mit $\boldsymbol{y} - \boldsymbol{e} \in \mathcal{C}$.

Im Folgenden betrachten wir zwei Methoden, diese Art der Decodierung zu implementieren: mittels sogenannter Standard-Tabellen, was unpraktisch aber konzeptionell einleuchtend ist, und mittels der effizienteren Syndrom-Decodierung.

1.3.1 Decodierung linearer Codes mittels Standard-Tabelle

Wir betrachten einen linearen $[n, k, d]$-Code $\mathcal{C}$ über $F = \mathrm{GF}(q)$. Unter einer *Standard-Tabelle* für $\mathcal{C}$ versteht man eine Auflistung aller Elemente von F^n in Form einer $(q^{n-k} \times q^k)$-Matrix, so daß

- die erste Zeile aus allen Codeworten besteht, beginnend mit $\boldsymbol{0}$;
- jede folgende Zeile mit einem bisher nicht aufgetretenem Wort e minimalen Hamming-Gewichts beginnt, gefolgt von den Wörtern $\boldsymbol{e} + \boldsymbol{c}$, wobei $\boldsymbol{c} \in \mathcal{C}$ die entsprechende Spalte anführt.

1.3.01 Beispiel. Die systematische Generatormatrix

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

erzeugt einen $[5, 2, 3]$-Code. Eine mögliche Standard-Tabelle ist

00000	10110	01011	11101
00001	10111	01010	11100
00010	10100	01001	11111
00100	10010	01111	11001
01000	11110	00011	10101
10000	00110	11011	01101
00101	10011	01110	11000
10001	00111	11010	01100

Offenbar ist die Standard-Tabelle nicht eindeutig bestimmt. Abgesehen davon, daß die Codewörter nicht lexikographisch geordnet sind, können wir die fünf Zeilen, deren erster Eintrag Hamming-Gewicht 1 hat, permutieren. Schließlich kann die siebte Zeile mit jeden der Wörter 00101, 11000, 10001 oder 01100 beginnen.

Bei jeder Zeile der Tabelle handelt es sich um eine *Nebenklasse* (englisch *coset*) des Untervektorraums $\mathcal{C} \leq F^n$. Zwei Wörter $\boldsymbol{x}, \boldsymbol{y} \in F^n$ gehören genau dann zur selben Zeile, wenn ihre Differenz in $\mathcal{C}$ liegt. Die Nebenklassen sind also genau die Äquivalenzklassen bzgl. dieser Äquivalenzrelation. Sie partitionieren F^n in q^{n-k} viele Teilmengen der Größe $|\mathcal{C}| = q^k$, die alle die Form $\boldsymbol{y} + \mathcal{C}$ haben, für $\boldsymbol{y} \in F^n$.

Die ersten Elemente jeder Zeile hat innerhalb der Zeile minimales Hamming-Gewicht und wird *(Nebenklassen-)Anführer* genannt (englisch *coset leader*). Wie die letzten beiden Zeilen des Beispiels 1.3.01 zeigen, braucht der Anführer aber nicht eindeutig bestimmt zu sein.

Wird das Wort $\boldsymbol{y}$ empfangen, so muß jede Decodierungsstrategie ein Fehlerwort $\hat{\boldsymbol{e}} \in F^n$ bestimmen mit $\boldsymbol{y} - \hat{\boldsymbol{e}} \in \mathcal{C}$. Folglich muß das Fehlerwort derselben Nebenklasse angehören wie $\boldsymbol{y}$. Wenn die Decodierung das nächstliegende Codewort bestimmt, muß $\hat{\boldsymbol{e}}$ ein Element minimalen Gewichts in der Nebenklasse des Worts $\boldsymbol{y}$ sein. Das rechtfertigt folgende Vorgehensweise:

- Wird $\boldsymbol{y} \in F^n$ empfangen, so bestimmt man die zugehörige Zeile der Standard-Tabelle und wählt deren Anführer $\boldsymbol{e}$ als Fehlerwort. Das resultierende Codewort $\boldsymbol{c} := \boldsymbol{y} - \boldsymbol{e}$ ist das erste Element der Spalte, in der $\boldsymbol{y}$ liegt; folglich braucht die Subtraktion gar nicht ausgeführt zu werden.

Wird im obigen Beispiel $\boldsymbol{y} = (0\,1\,1\,1\,1)$ empfangen, wird $\boldsymbol{e} = (0\,0\,1\,0\,0)$ als Fehler angenommen und $D(\boldsymbol{y}) = \boldsymbol{c} = (0\,1\,0\,1\,1)$ decodiert.

Solange das Hamming-Gewicht des Anführers den Wert $(d-1)/2$ nicht übersteigt, handelt es sich nach Proposition 0.4.01 um das einzige Wort minimalen Hamming-Gewichts in der entsprechenden Nebenklasse. Im obigen Beispiel trifft dies für die Anführer mit Hamming-Gewicht 1 zu.

1.3.2 Syndrom-Decodierung linearer Codes

Wir wollen jetzt eine minimale Kontrollmatrix H eines linearen $[n,k,d]$ Codes $\mathcal{C} \leq F^n$ zur Decodierung verwenden. Damit die Spalten von H linear unabhängig sind, muß es sich um eine $n \times (n-k)$-Matrix handeln.

Als *Syndrom* eines empfangenen Worts $\boldsymbol{y} \in F^n$ bezüglich H bezeichnet man den Vektor

$$\boldsymbol{s} := \boldsymbol{y} \cdot H \in F^{n-k}$$

Nach Definition der Kontrollmatrix sind Codewörter durch das Syndrom $\boldsymbol{0}$ charakterisiert. Außerdem gilt für $\boldsymbol{y}_0, \boldsymbol{y}_1 \in F^n$

$$\boldsymbol{y}_1 - \boldsymbol{y}_0 \in \mathcal{C} \qquad \text{genau dann wenn} \qquad \boldsymbol{y}_0 \cdot H = \boldsymbol{y}_1 \cdot H$$

d.h., $\boldsymbol{y}_0$ und $\boldsymbol{y}_1$ gehören genau dann zur selben Nebenklasse von $\mathcal{C}$, wenn sie dasselbe Syndrom haben. Folglich besteht eine Bijektion zwischen den q^{n-k} Nebenklassen von $\mathcal{C}$ und den q^{n-k} möglichen Werten, die das Syndrom bzgl. H annehmen kann, wobei die triviale Nebenklasse $\mathcal{C}$ dem Syndrom $\boldsymbol{0}$ entspricht.

Wie können ein nächstes Codewort mit folgender Zwei-Schritt-Strategie bestimmen:

(0) das Syndrom $\boldsymbol{s}$ des empfangenen Worts $\boldsymbol{y}$ bzgl. H wird bestimmt zu $\boldsymbol{s} = \boldsymbol{y} \cdot H$;

(1) ein Fehlerwort $\boldsymbol{e}$ minimalen Hamming-Gewichts mit $\boldsymbol{s} = \boldsymbol{e} \cdot H$ wird bestimmt und von $\boldsymbol{y}$ subtrahiert.

Schritt (0) erfordert nur die Multiplikation eines Vektor mir einer Matrix. Schritt (1) ist dagegen äquivalent dazu, die kleinste Auswahl von Zeilen der Matrix H zu finden, die den Vektor $\boldsymbol{s}$ erzeugen. Für allgemeine Vektoren $\boldsymbol{s}$ und Matrizen H ist diese Problem $\boldsymbol{NP}$-vollständig. Aber für kleine Werte von $n-k$ kann man eine *Syndromliste* erstellen, die neben den q^{n-k} Syndromen die zugehörigen Anführer vermerkt. Bei Codes, deren Kontrollmatrizen eine spezielle Struktur haben, kann Schritt (1) auch für große Werte von $n-k$ handhabbar bleiben, sofern die Anzahl der zu korrigierenden Fehler $(d-1)/2$ nicht überschreitet. Läßt sich in solchen Fällen der gesuchte Anführer effizient berechnen, so kann man auf die Syndromliste verzichten.

Das Konzept des Syndroms läßt sich auf Kontrollmatrizen verallgemeinern, die mehr als $n-k$ Spalten haben. Zwar liefert das Syndrom in diesem Fall Werte in F^r mit $r > n-k$, aber der Bildraum von F^n unter der zugehörigen linearen Abbildung bleibt $n-k$-dimensional, d.h., letztere ist nicht mehr surjektiv.

1.3.02 Beispiel. Für den $[2^m-1, 2^m-1-m, 3]$ Hamming-Code $\mathcal{C}$ über GF(2) wählen wir die Kontrollmatrix H, deren Zeile i die Binärdarstellung von $i+1$ enthält, $i < 2^m$.

Sofern das empfangene Wort $\boldsymbol{y} \in F^n$ höchstens einen Fehler enthält, gilt $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$, wobei $\boldsymbol{c}$ ein Codewort und $\boldsymbol{e}$ ein Einheitsvektor oder $\boldsymbol{0}$ ist. Die Syndrome der Wörter $\boldsymbol{y}$ und $\boldsymbol{e}$ stimmen überein

$$\boldsymbol{s} = \boldsymbol{y} \cdot H = \boldsymbol{e} \cdot H$$

Ist genau ein Fehler an der Stelle $j < n$ aufgetreten, so entspricht das Syndrom der Zeile j der Kontrollmatrix, also der Binärzahl $j + 1$. Somit kann die Korrektur direkt erfolgen.

2 Endliche Körper: algebraische Grundlagen

Vom Hauptthema abweichend müssen wir uns nun mit den mathematischen Grundlagen auseinandersetzen. Eigenschaften endlicher Körper werden in späteren Abschnitten benutzt, um Codes zu konstruieren. Zur Motivation konstruieren wir am Ende dieses Abschnitts einen binären Code, der zwei Fehler korrigieren kann und der mit Hilfe endlicher Körper beschrieben und analysiert wird. Die Konstruktion stellt sich als Spezialfall eines allgemeineren Verfahrens heraus.

Die Darstellung endlicher Körper benötigt den Begriff des irreduziblen Polynoms, vergleiche Unterabschnitt 2.2.3. In Anbetracht der praktischen Realisierbarkeit bestimmter Codierer und Decodierer werden wir sowohl Verallgemeinerungen von Polynomen (Unterabschnitte 2.2.1 und 2.2.2) als auch deren Handhabung mit Hilfe von Schieberegistermaschinen (Abschnitt 2.3) betrachten.

Zu den Haupteigenschaften endlicher Körper, die wir hier untersuchen wollen, gehört die Zyklizität der multiplikativen Gruppe. Sie erlaubt es die algebraischen Operationen in nicht zu großen endlichen Körpern mittels Tabellen zu implementieren. Wir zeigen zudem, daß als Größen endlicher Körper genau die Primzahlpotenzen vorkommen. Im binären Fall ermöglicht das auch die Behandlung von Bytes (oder größerer Blöcke) anstelle einzelner Symbole.

2.1 Primkörper

Für eine ganze Zahl $p > 0$ bezeichnet $\mathbb{Z}_p$ den Ring der ganzzahligen Reste modulo p, also $\mathbb{Z}_p = \{ n \in \mathbb{N} : n < p \}$. Nach dem Euklidischen Algorithmus gibt es zu jeder Zahl $a \in \{1, \ldots, p-1\}$ ganze Zahlen s und t mit

$$s \cdot a + t \cdot p = \mathrm{ggT}(a, p)$$

Eine Primzahl p ist zu jedem derartigen a teilerfremd, also gilt $\mathrm{ggT}(a, p) = 1$. Dann ist $s \bmod p$ das multiplikative Inverse von a in $\mathrm{GF}(p)$, was dadurch als Körper identifiziert wird.

2.1.01 Beispiel. In $\mathrm{GF}(7)$ haben wir

$$2 \cdot 4 = 3 \cdot 5 = 6 \cdot 6 = 1 \cdot 1 = 1$$

Darüberhinaus gilt $a^6 = 1$ für jedes von 0 verschiedene Element von $\mathrm{GF}(7)$, folglich also $a^{-1} = a^5$.

Die multiplikative Gruppe von $\mathrm{GF}(7)$ stellt sich als zyklisch heraus und wird sowohl von 3 wie auch von 5 (was mit 3^{-1} übereinstimmt) erzeugt:

$$\begin{aligned} 3^0 &= 1 = 5^0 & \qquad 3^3 &= 6 = 5^3 \\ 3^1 &= 3 = 5^5 & \qquad 3^4 &= 4 = 5^2 \\ 3^2 &= 2 = 5^4 & \qquad 3^5 &= 5 = 5^1 \end{aligned}$$

Für jeden Körper F werden wir die neutralen Elemente bzgl. der Addition und Multiplikation mit 0 und 1 bezeichnen. Die multiplikative Gruppe schreiben wir als $F^* = F \setminus \{0\}$, während $O(a)$ für die multiplikative *Ordnung* eines Elements $a \in F^*$ steht, die kleinste positive Zahl $n \in \mathbb{N}$ mit $a^n = 1$, sofern diese existiert. Das ist im endlichen Fall garantiert, was zu folgendem Ergebnis führt.

2.1.02 Proposition. *Jedes Element a eines endlichen Körpers F erfüllt*

$$a^{|F|} = a$$

Beweis: Für $a = 0$ ist die Behauptung trivial. Falls $a \in F^*$, betrachten wir die von a erzeugte Untergruppe $\langle a \rangle := \{ a^n : n \in \mathbb{N} \}$ von F^*. Deren Nebenklassen partitionieren F^* und haben alle dieselbe Größe $O(a)$, woraus folgt, das $|F^*|$ von $O(a)$ geteilt wird. Wegen $a^{O(a)} = 1$ folgt damit auch $a^{|F^*|} = 1$ und somit $a^{|F|} = a$. □

Wir werden in Kürze zeigen, daß die multiplikative Gruppe jedes endlichen Körpers F *zyklisch* ist, d.h., von der Form $\langle a \rangle$ für ein geeignetes Element $a \in F$. Solch ein Element heißt auch *primitive Wurzel*.

2.2 Polynome und formale Potenzreihen

Wie in anderen Teilen der Algebra, so spielen auch in der Theorie der (linearen) fehlerkorrigierenden Codes Polynome eine wichtige Rolle, als nützliches Wekrzeug. Üblicherweise wird in der Literatur davon ausgegangen, daß die Leserschaft mit den Begriff des Polynoms vertraut ist. Dabei wird meist übersehen, daß aufgrund unglücklicher Terminologie und Notation diversen Mißverständnissen Vorschub geleistet wird. Das wollen wir hier vermeiden.

2.2.1 Was sind Polynome und formale Potenzreihen?

Will man für eine unstrukturierte Menge A alle (cartesischen) Potenzen A^i, $i \in \mathbb{N}$, gleichzeitig betrachten, so kann man diese mittels disjunkter Vereinigung zusammenfassen:

$$A^* = \sum_{i \in \mathbb{N}} A^i$$

was bekanntermaßen die Menge aller endlichen Wörter über A liefert. Dies ist die "beste" Menge, die alle Potenzen A^i, $i \in \mathbb{N}$ als Teilmengen umfaßt (im kategoriellen Sinne handelt es sich um den Colimes eines diskreten Diagramms). Man beachte, daß es keine Familie kanonischer Einbettungen $A^i \longrightarrow A^{i+1}$, $i \in \mathbb{N}$, gibt.

In der Codierungstheorie haben wir es meist nicht mit unstrukturierten Mengen A zu tun, sondern mit (endlichen) Körpern F, vergl. Kapitel 1. Deren Potenzen F^n, $n \in \mathbb{N}$ bilden dann Vektorräume, und es stellt sich nun die Frage, wie man diese am geschicktesten alle gleichzeitig betrachten könnte: was ist der "beste" Vektorraum über F, der alle Räume F^n, $n \in \mathbb{N}$, als

Teilräume umfaßt? Leider ist F^* kein Vektorraum, denn es gibt keine sinnvolle Möglichkeit, Wörter unterschiedlicher Länge zu addieren.

Zur Lösung dieses Problems benötigt man gar nicht die volle algebraische Struktur eines Körpers oder eines Vektorraums, es genügt die Existenz eines ausgezeichenten Elements der Menge F, in unserem Fall wird das $0 \in F$ sein.

Im Falle einer *punktierten Menge* $A = B + \{e\}$, haben die ersten cartesischen Potenzen folgende Form (wobei wir zur Abkürzung $\times$ unterdrücken):

$$
\begin{aligned}
(B + \{e\})^0 &= \{\varepsilon\} \\
(B + \{e\})^1 &= A + \{e\} \\
(B + \{e\})^2 &= A^2 + \{e\}A + A\{e\} + \{ee\} \\
(B + \{e\})^3 &= B^3 + \{e\}B^2 + B\{e\}A + \{e\}B^2 + \{ee\}B + \{e\}B\{e\} + B\{ee\} + \{eee\} \\
&\dots
\end{aligned}
$$

In dieser Situation gibt es zwei kanonische Einbettungen von $(B + \{e\})^i$ in $(B + \{e\})^{i+1}$: *linksbündig* via $(B + \{e\})^i\{e\}$ und *rechtsbündig* via $\{e\}(B + \{e\})^i$; d.h., die einzubettenden Wörter werden nach rechts, bzw. links, mit e aufgefüllt. Die linksbündige Variante paßt besser zur Standardindexierung der Tupel von links nach rechts, daher beschränken wir uns zunächst auf sie. Die rechtsbündige Variante wird später aber ebenfalls wichtig sein.

Die angegebenen linksbündigen Einbettungen lassen sich auch in die Menge $A^{\mathbb{N}}$ aller Abbildungen von $\mathbb{N}$ nach A forsetzen; derartige "unendliche Wörter" sind auch unter dem Namen *Ströme* bekannt. Der Colimes $A^{\triangleright}$ des Diagrams der linksbündigen Einbettungen besteht nun offenbar aus denjenigen Strömen, die nur endlich viele von e verschiedene Komponenten haben. Diese wollen wir als e*-Polynome* über A bezeichnen. Unter dem *Grad* eines solchen Polynoms p verstehen wir die größte Zahl $i \in \mathbb{N}$ mit $p_i \neq e$; die Komponente $p_{\operatorname{grd} p}$ heißt auch *Leitkoeffizient*. Für das konstante Polynom $\boldsymbol{e}$, dessen Komponenten alle den Wert e haben, existiert keine derartige Zahl, so daß wir seinen Grad auf $-\infty$ festsetzen. Wir erhalten also eine Abbildung $F[x] \xrightarrow{\operatorname{grd}} \mathbb{N} + \{-\infty\}$.

Per Konstruktion erhalten wir schließlich eine kanonische Abbildung von A^* in die Menge der e-Polynome über A durch "Rechtsauffüllen" mit e. Dies ist allerdings keine Einbettung, da z.B. alle endlichen nur aus e bestehenden Wörter dasselbe Bild haben, nämlich das konstante Polynom $\boldsymbol{e}$.

Im Falle unstrukturierter Mengen A mit $|A| > 1$ gibt es keine kanonische Abbildung von A^* nach $A^{\mathbb{N}}$; wir können für jedes $e \in A$ eine Abbildung nach obigem Muster konstruieren, aber keine davon ist "besser" als die anderen.

Während es zur Konstruktion der Menge aller Polynome über A nur der Existenz eines ausgezeichneten Elements $e \in A$ bedurfte, läßt sich die traditionelle Notation sowohl von Polynomen wie auch der Menge aller Polynome mit Hilfe einer sogenannten "Variablen" (meist x genannt) erst dann sinnvoll erklären, wenn es sich bei der Grundmenge A um einen Ring handelt; als ausgezeichnetes Element fungiert dann das neutrale Element 0 der Addition. Statt

von "0-Polynomen" sprechen wir in diesem Fall nur noch von "Polynomen". Das neutrale Element 1 der Multiplikation ermöglicht zudem die Definition *normierter Polynome*: diese haben den Leitkoeffizinten 1. Wir wollen uns hier gleich auf den Fall eines Körpers $\langle F, +, 0, \cdot, 1\rangle$ beschränken.

Zunächst stellen wir fest, daß sich die Addition $+$ komponentenweise auf jede Potenz F^n, $n \in \mathbb{N}$, aber auch auf $F^{\mathbb{N}}$ fortsetzen läßt, wobei das entsprechende konstante Tupel $\boldsymbol{0}$ aus lauter Nullen die Rolle des neutralen Elements spielt. Analog verhält es sich mit der Skalarmultiplikation mit Elementen aus F. All diese Potenzen sind also Vektorräume über dem Körper F. Man stellt leicht fest, daß $F^{\triangleright}$ ein UVR von $F^{\mathbb{N}}$ ist: die Bedingung, daß sich nur endlich viele Komponenten von 0 unterscheiden, bleibt bei der Addition bzw. Skalarmultiplikation von Strömen erhalten.

Genau wie der Vektorraum F^n für $n \in \mathbb{N}$ eine *kanonische Basis* aus *Einheitsvektoren* besitzt

$$\langle\, \langle 1\,0\ldots 0\rangle, \langle 0\,1\,0\ldots 0\rangle, \ldots \langle 0\ldots 0\,1\rangle\,\rangle$$

so verfügt $F^{\triangleright}$ über eine abzählbare derartige Basis (im Gegensatz zum Vektorraum $F^{\mathbb{N}}$, dessen Dimension sogar überabzählbar ist). Jedes Polynom ist eindeutig als endliche Linearkombination von Basiselementen darstellbar.

Aber die Mengen $F^{\triangleright}$ wie auch $F^{\mathbb{N}}$ sind mehr als nur Vektrorräume: die Skalarmultiplikation $F \times F^{\mathbb{N}} \longrightarrow F^{\mathbb{N}}$ läßt sich zu einer echten Multiplikation $F^{\mathbb{N}} \times F^{\mathbb{N}} \longrightarrow F^{\mathbb{N}}$ mit Hilfe einer sogenannten *Faltung* wie folgt fortsetzen

$$(p \cdot q)_k := \sum_{i+j=k} p_i q_j \qquad \text{für alle } k \in \mathbb{N} \tag{2.2-05}$$

wobei sich die Indizes von p und q in der Summe gegenläufig verhalten. Falls alle positiven Komponenten von p den Wert 0 haben, können wir p als Element des Bildes von F in $F^{\mathbb{N}}$ bzgl. der linksbündigen Einbettung auffassen; Formel 2.2-05 reduziert sich dann zur Skalarmultiplikation von $q \in F^{\mathbb{N}}$ mit $p_0 \in F$. Neutrales Element bzgl. dieser Multiplikation bleibt der Strom $\langle 1\ 0\ 0\ldots\rangle$, der dem Element $1 \in F$ entspricht. Die Assoziativität und die Distributivgesetze bzgl. der komponentenweisen Addition sind leicht nachzuweisen, also ist $F^{\mathbb{N}}$ selbst ein Ring, und $F^{\triangleright}$ stellt sich als Unterring heraus, der sogenannte *Polynomring* über F.

Nun versehen wir das Polynom $\langle 0\,1\,0\ldots\rangle \in F^{\triangleright}$ zur Abkürzung mit dem *Namen* x. (Jedes andere Element außerhalb der Menge F könnte ebenfalls verwendet werden. Die Bezeichnung "Variable" für x ist historisch bedingt und braucht nicht weiter hinterfragt zu werden. Vergl. auch die Bemerkung 2.2.02.) Die Multiplikation mit x hat nun den Effekt einer Rechtsverschiebung des anderen Stroms:

$$(x \cdot q)_k := \sum_{i+j=k} x_i q_j = q_{k-1}$$

Insbesondere erhalten wir den Einheitsvektor mit 1 in Komponente k genau als x^k. Mit anderen Worten: die Potenzen x^k, $k \in \mathbb{N}$, bilden die kanonische Basis von $F^{\triangleright}$. Dies rechtfertigt

die traditionelle Bezeichnung $F[x]$ (die Wahl eines anderen Namens, etwa y, für den Einheitsvektor mit einer 1 in Komponente 1 würde die Bezeichnung $F[y]$ liefern). Um für ein Element p von $F^{\triangleright} = F[x]$, das sich eindeutig als Linearkombination dieser Basiselemente darstellen läßt, zum Ausdruck zu bringen, daß man (zur Zeit) den Namen x für den Einheitsvektor mit 1 in Komponente 1 verwendet, hat sich traditionell die Bezeichnung $p(x)$ eingebürgert:

$$p = p(x) = \sum_{i \leq \operatorname{grd} \boldsymbol{p}} p_i x^i \tag{2.2-06}$$

Achtung: es handelt sich hier *nicht* um die Auswertung einer Funktion namens p an der Stelle x, sondern (x) ist nur um einen Hinweis auf den aktuell gültigen Namen eines speziellen Einheitsvektors. Dieser kollidiert unglücklicherweise mit der traditionellen Notation für die Funktionsausvwertung.

2.2.01 Definition. Die von einem Polynom $p(x) = \sum_{i<n} p_i x^i \in F[x]$ induzierte *Polynomfunktion*, die $a \in F$ auf $\sum_{i<n} p_i a^i \in F$ abbildet, bezeichnen wir mit $p^{\rightarrow}$.

Anstelle von $p^{\rightarrow}(a)$ wird abkürzend meist $p(a)$ geschrieben, was einer Identifikation des Polynoms $p = p(x) \in F[x] = F^{\triangleright}$ mit der Funktion $F \xrightarrow{p^{\rightarrow}} F$ Vorschub leistet.

2.2.02 Bemerkung. Für endliche Körper F kann die Zuordnung $p \mapsto p^{\rightarrow}$ aus Definition 2.2.01 allein schon aus Mächtigkeitsgründen nicht injektiv sein, denn $F[x]$ ist offensichtlich unendlich. Beispielsweise sind die Tupel 101 und 110 über GF(2) verschieden, also auch die Polynome $1+x$ bzw. $1+x^2$. Aber die Funktionen $(1+x)^{\rightarrow}$ sowie $(1+x^2)^{\rightarrow}$ auf GF(2) stimmen überein: sie vertauschen 0 und 1.

Selbst für unendliche Körper F braucht $(\,)^{\rightarrow}$ nicht injektiv zu sein. Ist aber jedes positive Vielfache von 1 von 0 verschieden (woraus die Unendlichkeit von F folgt), so läßt sich die Injektivität von $(\,)^{\rightarrow}$ zeigen. Ringe mit dieser Eigenschaft haben die *Charakteristik* 0. Dies gilt z.B. für die Ringe $\mathbb{Z}$, $\mathbb{Q}$ und $\mathbb{R}$.

Solange man nur Polynome über einem Ring der Charakteristik 0 betrachtet, ist die Identifikation mit den entsprechenden Polynomfunktionen zulässig und wird durch die Notation nahegelegt. In diesem Fall könnte man x als "variables Element" des Ring auffassen. Im Allgemeinen bringt diese Sichtweise aber eher Nachteile und sollte vermieden werden.

Die Summendarstellung (2.2-06) läßt sich auch auf Ströme $\varphi \in F^{\mathbb{N}}$ übertragen, was zum Begriff der *formalen Potenzreihe* führt:

$$\varphi = \varphi(x) = \sum_{i \in \mathbb{N}} \varphi_i x^i$$

Der Zusatz "formal" deutet an, daß es gerade *nicht* um die Definition von Funktionen $F \longrightarrow F$ mittels potentiell unendlicher Summen geht; dabei würden sich im Gegensatz zu Polynomfunktionen Konvergenzfragen bzgl. einer geeigneten Metrik oder Topologie auf F stellen.

Um die "Variable" x auch in der Notation für die Menge aller formalen Potenzreihen bzw. Ströme kenntlich zu machen, setzt man $F[\![x]\!] := F^{\mathbb{N}}$. Man beachte, daß die Gradabbildung $F[x] \xrightarrow{\text{grd}} \mathbb{N} + \{-\infty\}$ nicht auf $F[\![x]\!]$ fortgesetzt werden kann.

Die obige Formel (2.2-05) für die Multiplikation von Polynomen läßt sich auf formale Potenzreihen erweitern und erhält mit Hilfe der "Variablen" x folgende Form

$$p(x) \cdot q(x) = \Big(\sum_{i\in\mathbb{N}} p_i x^i\Big)\Big(\sum_{j\in\mathbb{N}} q_j x^j\Big) = \sum_{k\in\mathbb{N}}\Big(\sum_{i+j=k} p_i q_j\Big)x^k \tag{2.2-07}$$

Dies funktioniert, weil die innere Summe auf der rechten Seite, die in F gebildet wird, nur aus endlich vielen Termen besteht.

2.2.2 Rationale Funktionen und Laurent-Reihen

Zwar liegt unser Hauptaugenmerk auf endlichen Körpern, aber im Zusammenhang mit Polynomen und formalen Potenzreihen bietet es sich an, auch zwei unendliche Körper zu betrachten. Zudem werden sich diese in Kaptitel 7 als wichtig erweisen.

Offenbar sind weder $F[x]$ noch $F[\![x]\!]$ Körper, in beiden Fällen hat x kein Inverses. Allerdings lassen sich beide Ringe auf kanonische Weise zu unendlichen Körpern erweitern.

Im Falle von $F[x]$ können wir dieselbe Konstruktion anwenden, mit deren Hilfe man die Menge $\mathbb{Q}$ der *rationalen Zahlen* aus der Menge $\mathbb{Z}$ der ganzen Zahlen erhält: man bildet formale Brüche mit von $\boldsymbol{0}$ verschiedenen Nennern und faktorisiert nach einer geeigneten Äquivalenzrelation. Deren Äquivalenzklassen werden kanonisch durch *reduzierte Brüche* repräsentiert, deren Zähler und Nenner relativ prim sind.

2.2.03 Definition. Auf der Menge $F[x]\times(F[x]\setminus\{\boldsymbol{0}\})$ betrachten wir folgende Äquivalenzrelation:

$$\langle h(x), g(x)\rangle \sim \langle m(x), k(x)\rangle \quad \text{iff} \quad h(x)\cdot k(x) = m(x)\cdot g(x)$$

Unter einer *rationalen Funktion* über F verstehen wir eine $\sim$-Äquivalenzklasse. Die Menge der rationalen Funktionen bezeichnen wir mit $F(x)$.

Als Repräsentanten werden Paare von Polynomen $\langle p(x), q(x)\rangle \in F[x] \times (F[x] \setminus \{\boldsymbol{0}\})$ für uns von Interesse sein mit $\mathrm{ggT}(p(x), q(x)) = 1$ und $q(x)$ normiert, d.h., mit Leitkoeffizient 1. Letzteres garantiert $q(x) \neq \boldsymbol{0}$.

Mit Hilfe des Nenners 1 lassen sich Polynome als rationale Funktionen auffassen. Der Begriff "Funktion" ist hier eigentlich irreführend: zwar induziert eine rationale Funktion $\langle p(x), q(x)\rangle$ eine partielle Funktion $F \rightharpoonup F$, die $a \in F$ auf $p(a)\cdot(q(a))^{-1}$ abbildet (was im Falle $q(a) = 0$ nicht definiert ist), aber wie im Fall von Polynomen braucht diese Zuordnung nicht injektiv zu sein.

Die Addition, Multiplikation und Division rationaler Funktionen folgt den von den rationalen Zahlen bekannten Rechenregeln; ggf. sind die resultierenden Zähler und Nenner aber nicht relativ prim und müssen noch entsprechend reduziert werden. Dennoch ergibt sich unmittelbar

2.2.04 Satz. *$F(x)$ ist ein Körper, der $F[x]$ umfaßt.* □

Zwar hat die Erweiterung von $F[\![x]\!]$ zu einem Körper kein Gegenstück im Bereich der Konstruktion von Zahlenkörpern, dennoch ist die Konstruktion recht einfach.

2.2.05 Definition. Unter einer *Laurentreihe* über F verstehen wir eine Funktion $\mathbb{Z} \xrightarrow{a} F$ mit der Eigenschaft, daß nur endlich vielen negativen Zahlen ein von 0 verscheidener Wert zugewiesen wird. Die Menge der Laurentreihen über F bezeichnen wir mit $F(\!(x)\!)$.

Die Erweiterung im Vergleich zu formalen Potenzreihen besteht also darin, daß endlich viele Terme mit negativem Index (und folglich negativen Potenzen von x) zugelassen werden. Die formalen Potenzreihen entsprechen also genau den Laurent-Reihen, bei denen alle Koeffizienten mit negativem Index den Wert 0 annehmen.

Die Multiplikation von Laurent-Reihen erfolgt ebenfalls gemäß Formel (2.2-07). Die Endlichkeit der innere Summe auf der rechten Seite wäre aber nicht mehr zu garantieren, wenn man beliebige Funktionen $\mathbb{Z} \longrightarrow F$ auf diese Weise multiplizieren wollte.

2.2.06 Satz. *$F(\!(x)\!)$ ist ein Körper, der $F[\![x]\!]$ umfaßt.*

Beweis: Der Nachweis der Ringeigenschaften erfolgt wie im Falle von $F[\![x]\!]$. Es bleibt also die Invertierbarkeit aller von $\boldsymbol{0}$ verschiedenen Elemente zu zeigen.

Für $p(x) \in F(\!(x)\!)\backslash\{\boldsymbol{0}\}$ sei $i \in \mathbb{Z}$ der erste Index mit $p_i \neq 0$. Wir definieren $q(x) \in F(\!(x)\!)$ rekursiv wie folgt

$$q_{m-i} = \begin{cases} 0 & \text{falls } m < 0 \\ (p_i)^{-1} & \text{falls } m = 0 \\ -q_{-i} \sum_{k+l=m, l>0} q_{k-i} p_{i+l} & \text{falls } m > 0 \end{cases}$$

Wegen $q_{-i}p_i = 1$ und $\sum_{k+l=m} q_{k-i}p_{i+l} = 0$ für $m > 0$ erhalten wir $p(x) \cdot q(x) = 1$, wie erforderlich. □

2.2.3 Polynomdivision, Irreduzibilität und Wurzeln

Zwar sind weder $F[x]$ noch $F[\![x]\!]$ Körper, aber zumindest in $F[x]$ können wir wie im Ring $\mathbb{Z}$ mit Rest dividieren (Polynomdivision), wobei allerdings der Größenvergleich zwischen Rest und Divisor durch einen Vergleich der entsprechenden Grade zu ersetzen ist. Dies erlaubt auch die Übertragung des Euklidischen Algorithmus zur Bestimmung des ggT von $\mathbb{Z}$ auf $F[x]$. Dagegen ist in $F[\![x]\!]$ eine Division mit Rest schon wegen der fehlenden Grad-Abbildung nicht möglich.

2.2.07 Beispiel. Über GF(2) betrachten wir $p(x) = x^4 + x2 + x + 1$ sowie $q(x) = x^3 + 1$. Deren ggT bestimmen wir mit Hilfe des Euklidischen Algorithmus wie folgt:

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

woraus wir $\mathrm{ggT}(x^4 + x^2 + x + 1, x^3 + 1) = x + 1$ ablesen.

Allgemein existieren für Polynome $p(x), q(x) \in F[x] \setminus \{0\}$ weitere Polynome $s(x), t(x) \in F[x]$ mit

$$ggT(p(x), q(x)) = s(x) \cdot p(x) + t(x) \cdot q(x)$$

Diese erhält man durch Rücksubstitution im Euklidischen Algorithmus. Im obigen Beispiel 2.2.07 etwa

$$\begin{aligned} \mathrm{ggT}(p(x), q(x)) &= x + 1 \\ &= (x^3 + 1) + x(x^2 + 1) \\ &= (x^3 + 1) + x[(x^4 + x^2 + x + 1) + x(x^3 + 1)] \\ &= x(x^4 + x^2 + x + 1) + (x^2 + 1)(x^3 + 1) \end{aligned}$$

also $s(x) = x$ und $t(x) = (x^2 + 1)$.

2.2.08 Bemerkung. Der Algorithmus für die Polynomdivision verwendet Polynome in der Reihenfolge fallender Potenzen von x. Wie wir in Abschnitt 2.3 sehen werden, läßt sich dieser Algorithmus so abwandeln, daß zu Beginn nur der Leitkoeffizient des Dividenden bekannt sein muß. Pro Iterationsschritt wird dann genau ein weiterer Koeffizient für die nächstniedrigere Potenz von x verarbeitet. Dies zeigt, daß im Hinblick auf die Übertragungsreihenfolge eines Kanals neben der bisherigen Übersetzung von Tupeln $p = \langle\, p_i : i < n \,\rangle$ in Polynome $p(x) = \sum_{i<n} p_i x^i$ auch die umgekehrte Übersetzung mit fallenden Potenzen von x sinnvoll sein kann, sofern sie das Puffern von Werten vermeidet.

Konkret unterscheiden sich die beiden Interpretationen aufsteigend indexierter Tupel als Polynome darin, ob a_0 das *Least Significant Bit* (*LSB*), also den Koeffizienten von x^0, bezeichnet ("big endian"), oder das *Most Significant Bit* (*MSB*), also den Koeffizienten von x^{n-1} ("little endian").

Die fallende Polynomdarstellung entspricht z.B. auch der Dezimaldarstellung ganzer Zahlen. 153 wird als Summe $1{\cdot}10^2{+}5{\cdot}10^1{+}3{\cdot}10^0$ ausgewertet, der Wert der Polynomfunktion $x^2{+}5x{+}3$ an der Stelle $x = 10$.

Nun stellt sich die Frage, wie die fallende Polynomdarstellung formal zu behandeln ist. Ein erster Versuch liefert

$$\bar{p}(x) = \sum_{i<n} p_i x^{n-1-i} = x^{n-1} \sum_{i<n} p_i x^{-i} = x^{\operatorname{grd} p(x)} p\left(x^{-1}\right) \tag{2.2-08}$$

Negative Exponenten dienen zunächst der formalen Vereinfachung, wir werden sie in Kürze konzeptionell richtig einordnen.

Um die Analogie zwischen $F[x]$ und dem Ring $\mathbb{Z}$ der ganzen Zahlen zu vertiefen, brauchen wir noch ein Gegenstück zum Begriff der Primzahl. Ein Polynom $p(x) \in F[x]$ heißt *irreduzibel*, wenn es einen Grad besitzt und nicht ein Produkt von Polynomen echt kleineren Grades ist. Ein nicht irreduzibles Polynom heißt *reduzibel*.

Man beachte, daß für Polynome die Begriffe der Irreduzibilität und der Reduzibilität vom Körper F abhängen:

2.2.09 Beispiel. Wie im obigen Beispiel 2.2.07 gesehen, ist x^2+1 reduzibel in $\mathrm{GF}(2)[x]$, denn dort gilt $x^2+1=(x+1)(x+1)$. Andererseits ist x^2+1 irreduzibel sowohl über $\mathrm{GF}(3)$ wie auch über $\mathbb{R}$: in beiden Fällen impliziert die Annahme der Reduzibilität, d.h., $(x^2+1)=(x-\alpha)(x-\beta)$, daß $\alpha^2+1=(\alpha-\alpha)(\beta-\alpha)=0$ gelten muß. Aber die Quadrate in $\mathrm{GF}(3)$ sind durch 0 und 1 gegeben, und in $\mathbb{R}$ gibt es ebenfalls keine Lösung.

2.2.10 Beispiel. Wir bestimmen alle irreduziblen Polynome über $\mathrm{GF}(2)$ bis zum Grad 4.

Die konstanten Polynom $p(x)=0$ und $p(x)=1$ vom Grad 0 sowie die linearen Polynome $p(x)=x$ sowie $p(x)=x+1$ vom Grad 1 sind trivialerweise irreduzibel, denn ein Produkt zweier Polynome vom Grad ≥ 1 muß mindestens Grad 2 haben.

Als einzige *reduzible* Polynome vom Grad 2 erhalten wir somit

$$\begin{aligned} x^2 &= x\cdot x \\ x^2+x &= (x+1)\cdot x \\ x^2+1 &= (x+1)(x+1) \end{aligned}$$

folglich müssen alle anderen Polynome vom Grad 2 irreduzibel sein. Da bleibt nur $p(x)=x^2+x+1$ übrig.

Die reduziblen Polynome von Grad 3 sind gegeben durch

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

was als irreduzible Polynome nur $p(x)=x^3+x^2+1$ sowie x^3+x+1 übrigläßt.

Bei 16 Polynomen von Grad 4 wird es langsam ineffizient, die irreduziblen Polynome als Komplement der reduziblen Polynome zu bestimmen. Im Vorgriff auf Lemma 2.2.14 stellen wir fest, daß $a(x)\in F[x]$ genau dann durch $x-\alpha$ teilbar ist, wenn $a(\alpha)=0$ gilt. In unserem konkreten Fall liefert dies, daß $a(x)=\sum_{i<4}a_ix^i$ genau dann in $\mathrm{GF}(2)[x]$ nicht durch x bzw $x+1$ teilbar ist, wenn $a_0=a_1+a_2+a_3=1$ gilt. Damit verbleiben unter den Polynomen vom Grad 4 noch vier Kandidaten für Irreduzibilität:

$$x^4+x+1 \quad , \quad x^4+x^2+1 \quad , \quad x^4+x^3+1 \quad \text{sowie} \quad x^4+x^3+x^2+x+1$$

Diese sind noch auf Faktorisierbarkeit durch irreduzible Polynome vom Grad 2 zu überprüfen. Dabei erhalten wir

$$x^4+x^2+1=(x^2+x+1)^2$$

Die übrigen drei Kandidaten bestehen auch diesen Test und sind folglich irreduzibel.

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

2.2.11 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 Produktdarstellung 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 *len*, 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 $p(x) = l \prod_{j<e} q_j(x)$. 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

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

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

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

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

Was die Teilbarkeit ohne Rest von Polynomen angeht, so spielen Faktoren vom Grad 1 eine besondere Rolle, die einen speziellen Namen rechtfertigt.

2.2.13 Definition. $a \in F$ heißt *Wurzel* eines Polynoms $p(x) \in F[x] \setminus \{\boldsymbol{0}s\}$, wenn $p(x)$ durch $x-a$ teilbar ist. Falls $p(x)$ durch $(x-a)^r$ teilbar ist, aber nicht durch $(x-a)^{r+1}$, so bezeichnen wir $r \in \mathbb{N}$ als *Vielfachheit* der Wurzel a.

2.2.14 Lemma. *Ein Polynom $p(x) \in F[x] \setminus \{\boldsymbol{0}\}$ hat genau dann $a \in F$ als Wurzel, wenn a Nullstelle der Polynomfunktion $p^{\rightarrow}$ ist.*

Beweis: Wegen $p(x) = q(x)(x-a) + r(x)$ mit $\operatorname{grd} r(x) < 1$ gilt $p(a) = 0$ genau dann, wenn $r(a) = 0$, und letzteres ist genau dann der Fall, wenn $r(x) = \boldsymbol{0}$ gilt. □

2.2.15 Corollar. *Irreduzible Polynome haben keine Wurzeln.* □

2.3 Schieberegistermaschinen zum Rechnen mit Polynomen und Reihen

Zur Erinnerung: wir wollen die Tupel bzw. Ströme über F, die der Kanal ausgibt, in der Übertragungsreihenfolge bearbeiten; vereinbarungsgemäß stimmt diese mit der Schreibrichtung von links nach rechts überein.

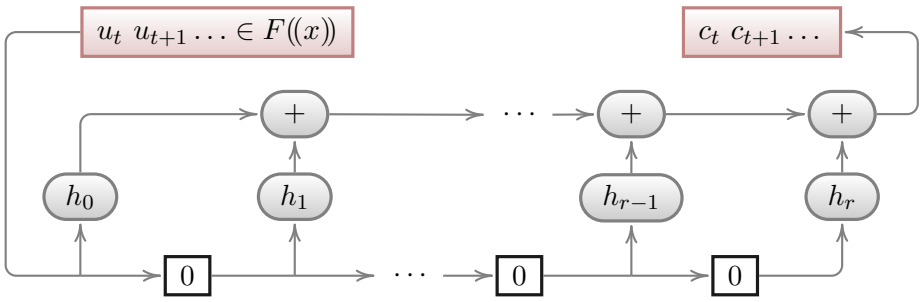
Je nach Problemstellung wird es zweckmäßig sein, die empfangenen Tupel oder Ströme als Polynome oder Reihen in auf- oder absteigender Reihenfolge der Potenzen von x zu interpretieren.

2.3.1 Multiplikation mit gegebenem Polynom

Wir betrachten ein Polynom $h(x) = \sum_{i \le r} h_i x^i$ vom Grad r. Gemäß der Faltungsformel (2.2-05) erfordert die Berechnung des Produkts $u(x) \cdot h(x)$ mit einer Laurent-Reihe in jeder Komponente eine Summe aus maximal r Termen.

$$(u \cdot h)_l := \sum_{j \le r} u_{l-j} h_j \qquad \text{für alle } l \in \mathbb{N} \tag{2.3-09}$$

Die einfachste Möglichkeit, dies umzusetzen, besteht darin, Skalarprodukte von $h \in F^{r+1}$ mit allen Teilwörtern von u der Länge $r+1$ *in umgekehrter Reihenfolge* zu bilden. In einer *Schieberegistermaschine* (oder kurz *SRM*) werden zu diesem Zweck die Koeffizienten von h fest als Faktoren "verdrahtet", während in den ursprünglich mit Nullen initialisierten Schieberegistern die Komponenten u_{k-r} bis u_{k-1} zwischenzuspeichert und im nächsten Schritt nach rechts verschoben werden. Diese bilden zusammen mit der aktuellen Komponente u_k das zu bearbeitende $r+1$-Tupel. Ist $t \in \mathbb{Z}$ ein Index mit $u_n = 0$ für $n < t$, so startet die Maschine in folgender Konfiguration:



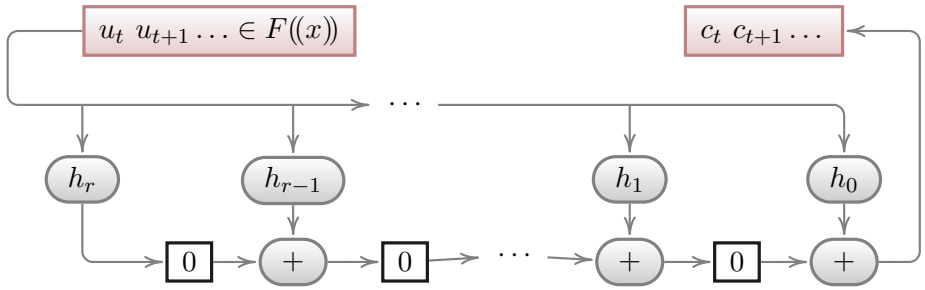
(2.3-10)

Ovale beschreiben algebraische Operationen, entweder Multiplikation mit dem spezifizierten Faktor, oder Addition. Schließlich gibt es noch Verzweigungspunkte.

Falls es sich bei $\boldsymbol{u}$ um ein Polynom des Grades $k-1$ handelt, gilt $t = 0$, und es genügt, als Eingabe das Tupel $\langle u_0 \ldots u_{k-1}\, 0 \ldots 0 \rangle$ mit r Nullen am Ende zu betrachten. Nach Abschluß der Berechnung sind die Schieberegister wieder mit Nullen gefüllt.

Ist in der verfügbaren Hardware die cascadierte Summation zur Berechnung der Faltung nicht effizient implementiert, können wir stattdessen die Teilsummen zwischenspeichern, die bei

der Multiplikation der aktuellen Komponente von $\boldsymbol{u}$ mit den Koeffizienten von $g(x)$ entstehen. Dazu verschieben wir die Schieberegister von der ersten in die dritte Zeile und vertauschen die Reihenfolge der Koeffizienten von g, damit wieder $c_t = u_t g_0$ gilt:



(2.3-11)

Falls es sich bei $\boldsymbol{u}$ um ein Polynom des Grades $k-1$ handelt, gilt $t = 0$, und es genügt, als Eingabe das Tupel $\langle u_0 \ldots u_{k-1}\, 0 \ldots 0\rangle$ mit r Nullen am Ende zu betrachten. Nach Abschluß der Berechnung sind die Schieberegister wieder mit Nullen gefüllt.

Wie in Bemerkung 2.2.08 angedeutet, kann man ein endliches Tupel u als Eingabe auch als nach fallenden Potenzen von x geordnetes Polynom $\bar{u}(x)$ interpretieren. Um die Multiplikation mit demselben festen Polynom $h(x)$ zu realisieren, wären in den obigen SRM'n nur die Koeffizienten von $h(x)$ in der umgekehrten Reihenfolge zu verwenden.

Angesichts der Notwendigkeit, die Eingabe mit Nullen von rechts aufzufüllen, stehen wir bei fallender Polynomdarstellung $\bar{u}(x)$ vor dem Problem, daß sich dadurch der Grad der Eingabe ändert: wir multiplizieren nicht mehr $\bar{u}(x)$ sondern $x^r \cdot \bar{u}(x)$ mit $h(x)$. Andererseits wollen wir im Hinblick auf eine effiziente Implementierung der Polynomdivision (mit Rest) ohne Puffer auf fallende Polynomdarstellungen nicht verzichten.

Um diese anscheinend widerstreitenden Ziele trotzdem miteinader vereinbaren zu können, erinnern wir uns an die fundamentale Asymmetrie bei der Definition der Polynommenge $A^{\triangleright}$ in Abschnitt 2.2: sie beruhte auf der Familie der linksbündigen Einbettungen $(B+\{e\})^n \longrightarrow (B+\{e\})^{n+1}$, die durch Rechts-Auffüllen mit e definiert waren. Verwendet man stattdessen die rechtsbündigen Einbettungen mittels Links-Auffüllen mit e, so ergibt sich die Menge $A^{\triangleleft}$ der *fallenden Polynome*, die mittels Spiegelung zu $A^{\triangleright}$ isomorph ist. Um diesem Umstand besser Rechnung zu tragen, bietet es sich an, statt der Indexmenge $\mathbb{N}$ die Menge $-\mathbb{N}$ der nicht positiven ganzen Zahlen zu verwenden (dabei gilt $-0 = 0$). Anstelle der etwas umständlichen Formalisierung (2.2-08) erhalten wir nun folgende Darstellung

$$\check{p}(x) = \sum_{n>i} p_{-i} x^i = \sum_{n>i} p_{-i} (x^{-1})^{-i} = \sum_{-n<j\leq 0} p_j (x^{-1})^j$$

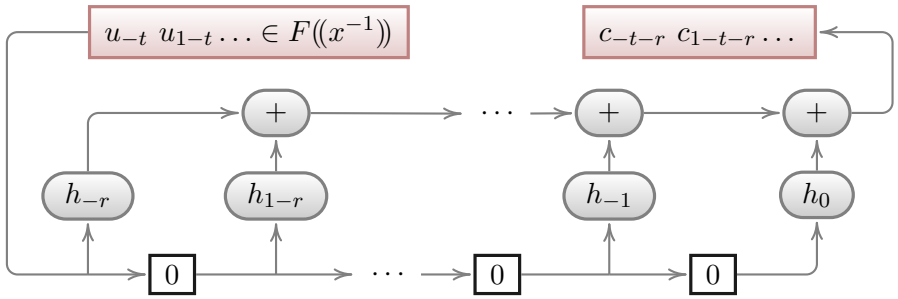
wobei die Summation von $n-1$ bzw. $1-n$ bis 0 erfolgt. Damit haben wir fallende Polynome als Laurent-Reihen in x^{-1} ohne positive Komponenten ausgedrückt. Aber $F(\!(x^{-1})\!)$ ist zweifellos die korrekte Bezeichnung für den Körper der Laurent-Reihen, der sich bei rechtsbündigen Einbettungen

ergibt. Derartige Laurent-Reihen bestehen aus einem endlichen *polynomialen Teil* mit nichtpositiven Indizes und einem potentiell unendlichen *fraktionalen Teil* mit positiven Indizes. In Tuplelform kann man derartige Laurent-Reihen in Analogie zu Dezimalzahlen mit einem "Fraktionalpunkt" schreiben, der andeutet, wo die positiven Indizes beginnen.

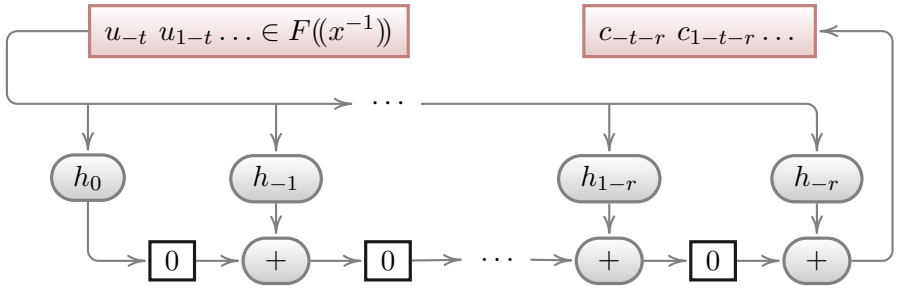
$$3425.21403\ldots \quad \text{steht für} \quad 3x^3 + 4x^2 + 2x^1 + 5x^0 + 2x^{-1} + 1x^{-2} + 4x^{-3} + 3x^{-5} + \cdots$$

Die Addition derartiger Laurent-Reihen erfolgt natürlich nach wie vor komponentenweise, *ohne* die bei Dezimalzahlen nötigen Überträge.

Mit dieser Neuinterpretation ist die Ergänzung eines fallenden Polynoms mit Nullen nach rechts unproblematisch: dies erfolgt rechts des Fraktionalpunkts und verändert somit nicht den Grad. Allerdings sollte h entsprechend behandelt und wie folgt als Polynom interpretiert werden: $\check{g}(x) = \sum_{i \geq r} h_{-i} x^i$. Die SRM'n (2.3-10) und (2.3-11) nehmen nun folgende Form an



(2.3-12)



(2.3-13)

2.3.2 Division durch gegebenes normiertes Polynom

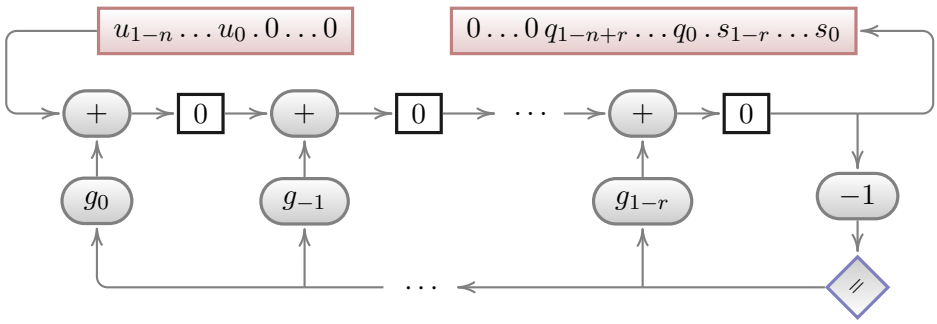
Der klassische Algorithmus zur Polynomdivision (mit Rest) beruht darauf, solange wie aus Gradgründen möglich den aktuell führenden Term mittels Subtraktion eines geeigneten Vielfachen des Divisors zu eliminieren. Die verwendeten Faktoren liefern die Terme des Quotienten; bei normiertem Divisor stimmen sie mit dem aktuellen Leitkoeffizienten überein. Zu diesem Zweck werden $\operatorname{grd} \check{g}(x)$ Koeffizienten des Dividenden in Schieberegistern (orange) gepuffert, bevor die Rechnung beginnt.

In Anbetracht unserer Definition rationaler Funktionen (Definition 2.2.03) wollen wir uns auf normierte Divisoren beschränken. Zudem erfordert im allgemeinen Fall die Behandlung des Restes zusätzlichen Schaltungsaufwand.

2.3.01 Beispiel. Wir illustrieren den klassischen Algorithmus in $\mathrm{GF}(3)$ am Beispiel der schriftlichen Division von $p = 20100202$, was fallend dem Polynom $\check{u}(x) = 2x^7 + x^5 + 2x^2 + 2$ entspricht, durch $g = 121102$, bzw. $\check{g}(x) = x^5 + 2x^4 + x^3 + x^2 + 2$: es gilt $\check{u}(x) = \check{q}(x) \cdot \check{g}(x) + \check{s}(x)$ mit $\operatorname{grd} \check{s}(x) < \operatorname{grd} \check{g}(x) = r$. Die Koeffizienten des Quotienten sind rot markiert, der Rest (grün) ergibt sich nach Aufbrauchen aller Bits, d.h., nach $\operatorname{grd} \check{u}(x) + 1 - \operatorname{grd} \check{g}(x)$ Schritten.

$$
\begin{array}{rcccccccc}
 & 2 & 0 & 1 & 0 & 0 & 2 & 0 & 2 \\
-(& 2 & 1 & 2 & 2 & 0 & 1 &) & \\
\hline
 & & 2 & 2 & 1 & 0 & 1 & 0 & \\
-(& & 2 & 1 & 2 & 2 & 0 & 1 &) \\
\hline
 & & & 1 & 2 & 1 & 1 & 2 & 2 \\
-(& & & 1 & 2 & 1 & 1 & 0 & 2 &) \\
\hline
 & & & & 0 & 0 & 0 & 2 & 0
\end{array}
$$

Die klassische Version des Algorithmus wird von folgender SRM modelliert, wobei aus darstellungstechnischen Gründen (Platzprobleme und Kombinierbarkeit mit der SRM (2.3-13)) die Schieberegister gegenüber der Handrechnung gespiegelt sind.

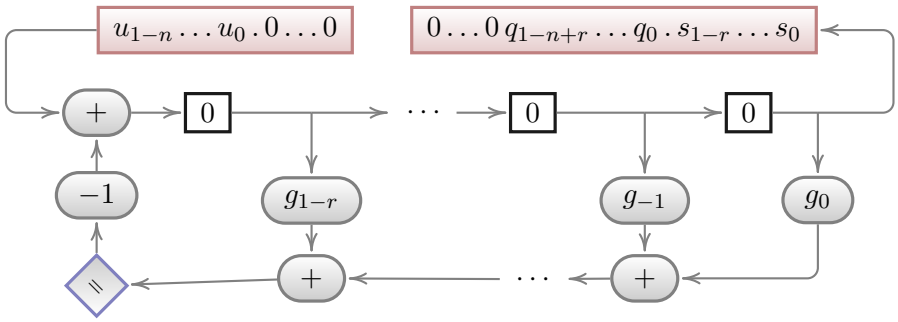


(2.3-14)

Nach n Schritten enthalten die Schieberegister die Koeffizienten des Restes, geordnet nach steigenden Potenzen von x. Nun unterbricht der Schalter den unteren Zweig, der fortan nur noch Nullen verarbeitet. In den folgenden r Schritten werden die Werte in den Schieberegistern sukzessive ausgegeben und von links mit Nullen ersetzt. Im Anschluß befindet sich die SRM wieder im Initialzustand.

Die Positionen des Faktors -1 und des Schalters können auch vertauscht werden, aber die angegebene Reihenfolge wird sich in Abschnitt 4.3, SRM (4.3-31), als vorteilhaft erweisen.

In Analogie zur SRM (2.3-12) kann auch in diesem Fall die Berechnung umorganisiert werden, so daß eine Faltung der Schieberegister und der letzten r Koeffizienten von $\check{g}(x)$ zur Anwendung kommt:



(2.3-15)

Um nachzuprüfen, daß diese Maschine tatsächlich in den ersten n Schritten den Quotienten $\check{u}(x)/\check{g}(x)$ berechnet, betrachten wir den Hilfs-Strom $\check{b}(x)$ der Werte an den Verzweigungspunkten der zweiten Reihe, der ist die führende Addition bestimmt wird. Wegen $g_{-r} = 1$ gilt

$$b_t = \begin{cases} 0 & \text{für } t \le -n \\ u_t - \sum_{0<w\le r} g_{w-r} b_{t-w} & \text{für } t > -n \end{cases} \quad \text{bzw.} \quad u_t = \begin{cases} 0 & \text{für } t \le -n \\ \sum_{w\le r} g_{w-r} b_{t-w} & \text{für } t > -n \end{cases}$$

Da $\check{q}(x)$ gegenüber $\check{b}(x)$ um r Poistionen verschoben ist, folgt die Behauptung.

2.3.02 Beispiel. (Fortsetzung von Beispiel 2.3.01) Mit dem obigen Verfahren liefert die Division von 2000202 durch 121102 mit Rest

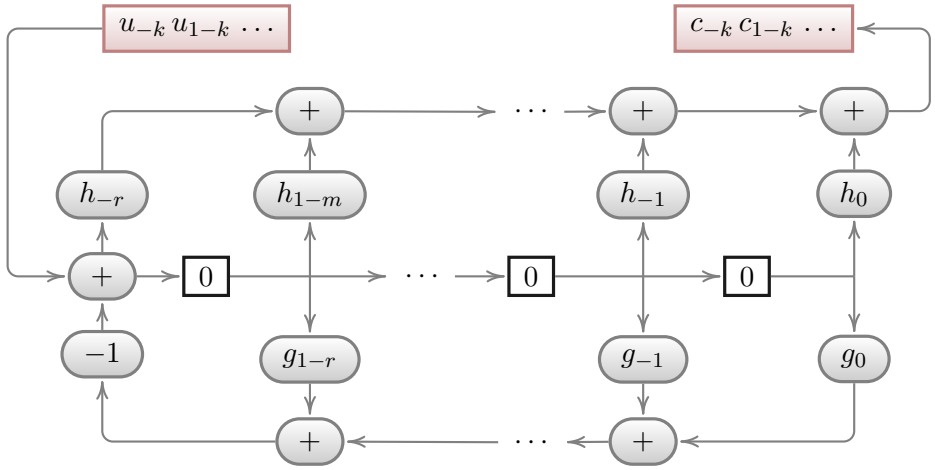
	2 1 1 0 2
	0 0 0 0 0
$2-0$	2 0 0 0 0
$0-1$	2 2 0 0 0
$1-0$	1 2 2 0 0
$0-0$	0 1 2 2 0
$0-0$	0 0 1 2 2
$2-2$	0 0 0 1 2
$0-1$	0 0 0 0 1
$2-2$	0 0 0 0 0

Die Koeffizienten von $g(x)$ ohne den Leitkoeffizienten 1 erscheinen in Blau in der ersten Zeile. Das Ergebnis ihrer Faltung mit dem bisherigen Inhalt der Schieberegister ist in Rot dargestellt, dieses ist von der Eingabe abzuziehen. Während der Berechnung des Quotienten ist das Ergebnis in das linke Schieberegister einzuspeisen, für den Rest aber nicht.

Die Effizienz beider Verfahren ist durch die Tatsache kompromittiert, daß die relevanten Ausgaben erst die r Schieberegister durchwandern müssen. Dies kann abgestellt werden, erfordert aber die Multiplikation mit der rationalen Funktion $\langle x^r, \check{g}(x)\rangle$.

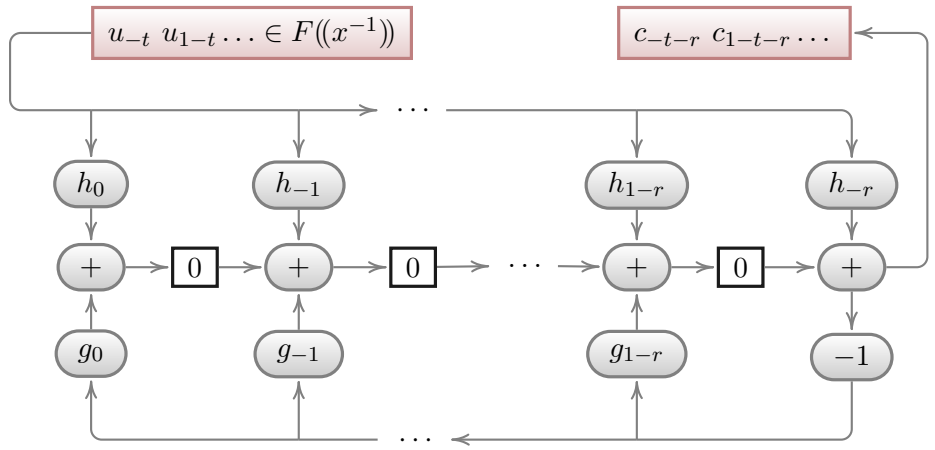
2.3.3 Multiplikation mit bestimmten rationalen Funktionen

Sofern $\operatorname{grd}\check{h}(x) \le \operatorname{grd}\check{g}(x) =: r$, lassen sich die SRM'n (2.3-12) und (2.3-15) (ohne Schalter) wie folgt zu einer SRM kombinieren, die das Produkt von $\check{u}(x)$ mit (dem Bild der) der rationalen Funktion $\langle\check{h}(x),\check{g}(x)\rangle$ in $F(\!(x^{-1})\!)$ berechnet:



(2.3-16)

Ganz analog kann man mit den SRM'n (2.3-13) und (2.3-14) verfahren, und erhält:

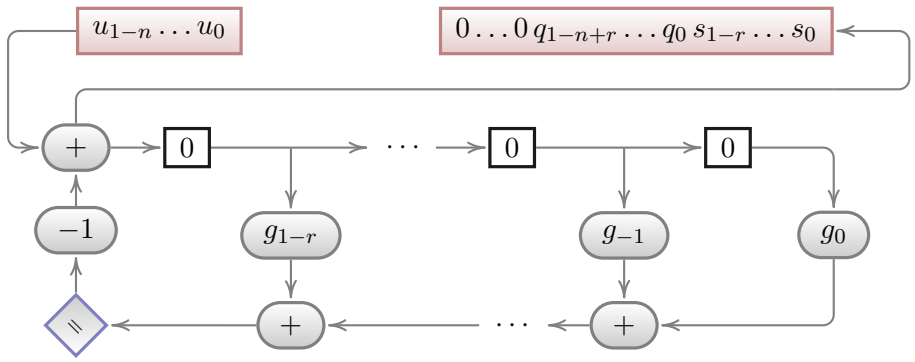


(2.3-17)

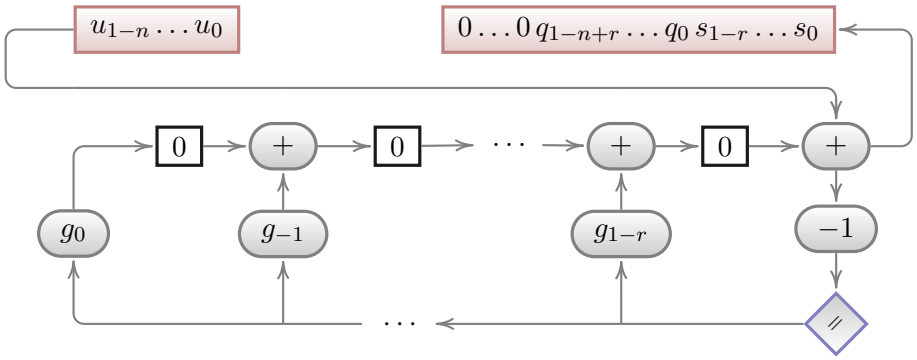
Es bleibt zu klären, was diese Maschinen leisten, wenn wir den Eingabestrom als Element von $F(\!(x)\!)$ interpretieren. Dann sind die Tupel h und g natürlich in umgekehrter Reihenfolge zu interpretieren, was insbesondere bedeutet, daß $g(x)$ nicht mehr normiert zu sein braucht, sondern stattdessen die Bedingung $g(0) = 1$ erfüllen muß. Insbesondere darf $g(x)$ nicht durch x teilbar sein.

In diesem Fall realisieren beide Maschinen die Multiplikation mit der rationalen Funktion $\langle h(x), g(x)\rangle$, aber das schließt bestimmte rationale Funktionen aus, die im vorangegangenen Fall erlaubt waren.

Wir kehren noch einmal zum Divisionsalgorithmus zurück. Ein Effizienz-Nachteil war sicherlich der Vorlauf von r Nullen vor der Ausgabe des ersten Bits des Quotienten. Im Fall der Division mit Rest waren dafür am Ende des Polynoms r weitere Nullen einzugeben, um die Schieberegister zu leeren. Um einen sofortigen Beginn der Ausgabe relevanter Bits zu erreichen, und die Eingabe überflüssiger Nullen zu vermeiden, können wir neben der Division durch $\check{g}(x)$ noch mit x^r multiplizieren, wobei $r := \operatorname{grd} \check{g}(x)$ gilt: dies verschiebt das Ergebnis der Division (mit Rest), betrachtet als Laurentreihe in $F((x^{-1}))$, um r Positionen nach links. Das funktioniert mit beiden SRM-Varianten



(2.3-18)



(2.3-19)

Was die schriftlichen Rechnung betrifft, so ist in jedem Schritt nur der aktuelle Leitkoeffizient von Interesse. Daher kann man alternativ die Koeffizienten des Dividenden auch ohne Puffern in den jeweiligen Schritten des Algorithmus einführen, geordnet nach fallenden Potenzen von x. Die Schieberegister akkumulieren dann die relevanten $\operatorname{grd} \check{g}(x)$ Bits der bisher subtrahierten Vielfachen des Divisors.

2.3.03 Beispiel. (Fortsetzung von Beispielen 2.3.01 und 2.3.02) Die Schieberegister sind wieder orange markiert, die Koeffizienten des Quotienten in rot, während sich der Rest (grün) nach $\operatorname{grd} \check{u}(x) + 1 - \operatorname{grd} \check{g}(x)$ Schritten durch Subtraktion des Inhalts der Schieberegister von den

letzten $\operatorname{grd} \check{g}(x)$ Bits der Eingabe ergibt.

$$
\begin{array}{rcccccccc}
 & \multicolumn{8}{c}{\overline{}} \\
 & 2 & 0 & 1 & 0 & 0 & 2 & 0 & 2 \\
 & \multicolumn{8}{c}{\overline{}} \\
 & 0 & 0 & 0 & 0 & 0 & & & \\
-(& 2 & 1 & 2 & 2 & 0 & 1 &) & \\
 & \multicolumn{6}{c}{\overline{}} & & \\
 & & 2 & 1 & 1 & 0 & 2 & & \\
 & -(& 2 & 1 & 2 & 2 & 0 & 1 &) \\
 & & \multicolumn{6}{c}{\overline{}} & \\
 & & & 0 & 2 & 1 & 2 & 2 & \\
 & & -(& 1 & 2 & 1 & 1 & 0 & 2 &) \\
 & & & \multicolumn{6}{c}{\overline{}} \\
 & & & & 0 & 0 & 1 & 2 & 1 \\
 & & & & \multicolumn{5}{c}{\overline{}} \\
 & & & & 0 & 0 & 0 & 2 & 0
\end{array}
$$

2.4 Algebraische Körpererweiterungen

Aus der Algebra sind wir mit der Tatsache vertraut, daß jedes Element s eines commutativen 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 kann man z.B. die Ringe $\mathbb{Z}_n = \mathbb{Z}/(n \cdot \mathbb{Z})$ erhalten, deren Elemente mit den möglichen Resten bei der Division durch n identifiziert werden können. 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 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. Die von verschiedenen Polynomen $f(x)$ und $f'(x)$ desselben Grades n induzierten Ring-Strukturen auf F^n werden aber i.A. nicht übereinstimmen. Während sie sich in der (komponentenweisen) Addition nicht unterscheiden können, werden nicht alle in $F[x]$ gebildeten Produkte bei der Division durch $f(x)$ bzw. $f'(x)$ dieselben Reste haben.

2.4.01 Proposition. *Der Faktorring $F[x]_{f(x)}$ ist genau dann ein Körper (mit $|F|^{\operatorname{grd} f(x)}$ Elementen), wenn $f(x)$ irreduzibel ist.*

Beweis: Setze $n := \operatorname{grd} f(x)$. Wir müssen zeigen, daß $F^n \backslash \{0\}$ genau dann bzg. der Multiplikation modulo $f(x)$ eine Gruppe bildet, wenn $f(x)$ irreduzibel ist.
($\Rightarrow$) Kontraposition: Ist $f(x)$ reduzibel, so existieren $a(x), b(x) \in F[x]$ mit $1 \leq \operatorname{grd} a(x), \operatorname{grd} b(x) < n$ und $a(x) \cdot b(x) = f(x)$. Gemäß der Gradbedingung gehören beide Faktoren zu $F[x]_{f(x)} \setminus \{0\}$, ihr Produkt in $F[x]_{f(x)}$ ist aber 0. Damit ist die Menge $F^n \setminus \{0\}$ nicht unter Multiplikation modulo $f(x)$ abgeschlossen.
($\Leftarrow$) Ist $f(x)$ irreduzibel und genügen $a(x), b(x) \in F[x]$ der Bedingung $1 \leq \operatorname{grd} a(x), \operatorname{grd} b(x) < n$, so ist nach Satz 2.2.11 das Produkt $a(x) \cdot b(x)$ in $F[x]$ kein Vielfaches von $f(x)$, da $f(x)$ sonst entweder $a(x)$ oder $b(x)$ teilen müßte. Damit ist die Menge $F^n \setminus \{0\}$ unter der Multiplikation modulo $f(x)$ abgeschlossen. Die Existenz von Inversen zeigen wir mittels vollständiger Induktion über den Grad. Die Polynome vom Grad 0 sind genau die Elemente

von $F \setminus \{0\}$ und somit invertierbar. Falls alle Polynome vom Grad $0 \leq i < k < n$ invertierbar sind und $a(x)$ den Grad k hat, dividieren wir zur Konstruktion von $a(x)^{-1}$ zunächst $f(x)$ durch $a(x)$ in $F[x]$ und erhalten

$$f(x) = q(x)a(x) + r(x) \quad \text{mit} \quad \operatorname{grd} r(x) < \operatorname{grd} a(x) \quad \text{und} \quad 0 \leq \operatorname{grd} q(x) < \operatorname{grd} f(x) \qquad (2.4\text{-}20)$$

Da $f(x)$ irreduzibel ist, folgt $r(x) \neq 0$. Nach Induktionsvoraussetzung existiert $r(x)^{-1} \in F^n \setminus \{0\}$. Die Interpretation von Gleichung (2.4-20) in $F[x]_{f(x)}$ liefert $0 = q(x) \cdot a(x) + r(x)$, woraus wir $a(x)^{-1} = -q(x) \cdot r(x)^{-1}$ schließen. □

Wir benötigen einen weiteren Satz aus der Algebra, dessen Beweis über den Rahmen dieses Skripts hinausgeht.

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

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

Um die Struktur dieser neuen endlichen Körper besser zu verstehen, fragen wir uns nun, ob sich die bekannte Erweiterung des Körpers $\mathbb{R}$ zum Körper $\boldsymbol{C}$ der komplexen Zahlen als Menge der Linearkombinationen von 1 und einem neuen (Basis-)Element i, das formale Nullstelle des irreduziblen Polynoms $x^2 + 1 \in \mathbb{R}[x]$ ist, auf endliche Körper und irreduzible Polynome übertragen läßt.

2.4.04 Beispiel. Ist α eine neue 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 $1 + 1 = 0$ und der angestrebten Distributivität, die $\alpha + \alpha = \alpha(1 + 1) = 0$ verlangt, 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

Hierbei handelt es sich um eine vierelementige Gruppe, die zu $\mathbb{Z}_2 \times \mathbb{Z}_2$ isomorph ist, aber nicht zu $\mathbb{Z}_4$. Was die Multiplikation angeht, so haben wir nach Konstruktion $\alpha^2 = \alpha + 1$. Daher erhalten wir die folgende Multiplikationstabelle für die von 0 verschiedenen Elemente:

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

Man weist nun leicht die Körperaxiome nach.

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

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

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

(1) Die innere Struktur von $GF(8) = \mathbb{Z}_2[x]_{x^3+x+1}$ wird aus folgender Tabelle deutlich. Als formale Wurzel von x^3+x+1 erfüllt α die Spezifikation $\alpha^3+\alpha+1 = 0$, bzw. $\alpha^3 = \alpha+1$ (der Basiskörper ist $\mathbb{Z}_2$). Damit sind Linearkombinationen nicht nur der (Basis-)Elemente 1 und α, sondern darüberhinaus α^2 zu betrachten, d.h., Polynome in α vom Grad ≤ 2. Die zweite Zeile ist eine Übersetzung dieser Polynome in Tupelschreibweise, geordnet nach fallenden Potenzen von α, während die Potenzen α^j mit $j > 2$ in der dritten Zeile aus der Gleichung $\alpha^3 = \alpha + 1$ abgeleitet sind. Die letzte Zeile listet die Ordnung der Körperelemente bzgl. der Multiplikation auf.

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

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

Da beide Körper isomorph sind, müssen wir β in der ersten und α in der zweiten Darstellung wiederfinden können. Zu diesem Zweck können wir α entweder auf β^3, oder auf β^5 oder auf $\beta^{-1} = \beta^6$ abbilden, aber nicht auf 1, β, β^2 oder β^4, denn in diesen Fällen ergibt sich kein additiver Homomorphismus.

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

2.4.06 Definition. Ist F ein Körper und ist $f(x) \in F[x]$ irreduzibel, so heißt $F[x]_{f(x)}$ *algebraische Körpererweiterung* (AKE) von F.

Lemma 2.2.14 läßt sich nun für algebraische Körpererweiterungen verallgemeinern:

2.4.07 Proposition. *Ein Element α einer AKE $\tilde{F}$ von F ist genau dann Wurzel eines Polynoms $p(x) \in F[x] \setminus \{0\}$ wenn $p(x)$ in $\tilde{F}[x]$ durch $x - \alpha$ teilbar ist.*

Beweis: Wegen $F \subseteq \tilde{F}$ ist $p(x)$ auch ein Polynom über $\tilde{F}$, und somit greift Lemma 2.2.14. □

Häufig kann man aus der Kenntnis einer Wurzel auf andere Wurzeln schließen. Dazu eignet sich die Äquivalenzrelation der *Konjugiertheit*.

2.4.08 Definition. Elemente α, β einer AKE $\tilde{F}$ von F heißen *konjugiert bzgl.* F, geschrieben $\alpha \sim_F \beta$, falls ein $r \in \mathbb{N}$ existiert mit $\alpha = \beta^{q^r}$.

2.4.09 Lemma. *Wir betrachten die AKE $\tilde{F} = \mathrm{GF}(q^m)$ von $F = \mathrm{GF}(q)$.*

(0) *Falls $\alpha = \beta^{q^r}$ für ein $r \in \mathbb{N}$, dann existiert auch ein $s < m$ mit $\alpha = \beta^{q^s}$.*

(1) *$\sim_F$ ist eine Äquivalenzrelation auf $\tilde{F}$.*

(2) *Die Äquivalenzklasse von $\beta \in \tilde{F}$ besteht aus den Potenzen β^{q^i}, wobei i durch die kleinste Zahl j mit $\beta^{q^j} = \beta$ echt beschränkt werden kann; insbesondere gilt $j \leq m$.*

(3) *Die multiplikative Ordnung ist auf Konjugationsklassen konstant.*

Beweis:

(0) Falls $m \leq r$ setzen wir $r = n \cdot m + s$ mit $s < n$. Wegen $\operatorname{ord} \beta \mid (q^m - 1)$ folgt

$$\beta^{q^{n \cdot m}} = \beta^{q^m q^{(n-1)m}} = \left(\beta^{q^m}\right)^{q^{(n-1)m}} = \beta^{q^{(n-1)m}}$$

woraus wir $\beta^{q^{n \cdot m}} = \beta$ schließen. Also folgt

$$\beta^{q^r} = \beta^{q^{n \cdot m + s}} = \left(\beta^{q^{n \cdot m}}\right)^{q^s} = \beta^{q^s}$$

(1) Die Reflexivität entspricht der Wahl von $r = 0$. Zum Nachweis der Symmetrie wählen wir im Falle $\alpha \sim_F \beta$ ein $s < m$ mit $\alpha = \beta^{q^s}$. Dann gilt nach obiger Rechnung $\beta = \alpha^{q^{m-s}}$, also $\beta \sim_F \alpha$. Schließlich folgt aus $\alpha = \beta^{q^r}$ und $\beta = \gamma^{q^t}$ sofort $\alpha = \gamma^{q^{r+t}}$, was die Transitivität zeigt.

(2) Teil (0) zeigt, daß jedes zu β äquivalente Element in der gewünschten Form dargestellt werden kann. Die Zulässigkeit der Einschränkung $j \leq m$ folgt sofort aus $\operatorname{ord} \beta \mid (q^m - 1)$.

(3) Falls $\alpha = \beta^{q^r}$, so folgt unmittelbar $\alpha^{\operatorname{ord}\beta} = 1$ und somit $\operatorname{ord}\alpha \leq \operatorname{ord}\beta$. Wegen der Symmetrie folgt aber auch die umgekehrte Ungleichung.

2.4.10 Proposition. *$\tilde{F}$ sei eine AKE des Körpers $F = \mathrm{GF}(q)$.*

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

(1) *Die Wurzelmenge von $f(x) \in F[x]$ in einer AKE $\tilde{F}$ ist eine Vereinigung von $\sim_F$-Klassen.*

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

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

Damit reduziert sich der Induktionsschritt darauf, die Behauptung für $n = 1$ explizit nachzuweisen. Nach der binomischen Formel gilt für $a, b \in F$

$$(a+b)^q = \sum_{k \leq q} \binom{q}{k} a^{q-k} b^k = a^q + \sum_{0<k<q} \binom{q}{k} a^{q-k} b^k + b^q$$

Wir wollen zeigen, daß alle Binominalkoeffizienten $\binom{q}{k} = q!/k!(q-k)!$ mit $0 < k < q$ in $F = \mathrm{GF}(q)$ verschwinden. Zunächst stellen wir fest, daß q eine Primzahlpotenz ist, etwa $q = p^\ell$ mit p prim und $\ell \in \mathbb{N}$. Unter Berücksichtigung der Symmetrie der Binimialkoeffizienten genügt es somit zu zeigen, daß $\binom{q}{k}$ für $0 < k < q/2$ durch p teilbar ist. Zu diesem Zweck schreiben wir den Binominalkoeffizienten wie folgt um

$$\binom{q}{k} = \prod_{i<k} \frac{q-i}{i+1} = \prod_{i<k} \frac{p^\ell - i}{i+1}$$

Für $k < p$, was speziell im Fall $\ell = 1$ gilt, ist dieses Produkt durch p^ℓ teilbar. Sonst ist das Produkt der ersten p Faktoren

$$\binom{q}{p} = \prod_{i<p} \frac{p^\ell - i}{i+1}$$

offenbar durch $p^{\ell-1} \geq p$ teilbar. Existieren p weitere Faktoren, so enthalten Zähler wie Nenner des entsprechenden Produkts

$$\frac{p\left(p^{\ell-1}-1\right)}{p+1} \cdot \frac{p\left(p^{\ell-1}-1\right)-1}{p+2} \cdot \ldots \cdot \frac{p\left(p^{\ell-1}-1\right)-(p+1)}{2p}$$

genau einen Faktor p. Entsprechendes gilt für jedes weitere Produkt von p Faktoren. Verbleiben zum Schluß $0 < r < p$ Faktoren, so enthält deren Produkt einen Faktor p im Zähler, der nicht gekürzt werden kann. Also ist $\binom{q}{k}$ mit $0 < k < q$ immer durch p teilbar.

(1) Da die Ordnung jedes von 0 verschiedenen Elements des endlichen Körpers F die Zahl $|F|-1=q-1$ teilt, erhalten wir für $f(x)=\sum_{i<n} f_i x^i$ wegen (0)

$$(f(x))^q=\sum_{i<n} f_i^q x^{iq}=\sum_{i<n} f_i x^{iq}=f(x^q)$$

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

2.4.11 Satz. *Ein Polynom vom Grad n über einem Körper F hat in jeder AKE von F maximal n Wurzeln, einschließlich Vielfachheit.*

Beweis: β_i, $i<k$, seien verschiedene Wurzeln von $f(x)\in F^{n+1}\leq F[x]$ in einer AKE $\tilde{F}$ von F mit Vielfachheit m_i. Dann teilt das Produkt $\prod_{i<k}(x-\beta_i)^{m_i}$, das den Grad $\sum_{i<k} m_i$ hat, $f(x)$, woraus $\sum_{i<k} m_i\leq n$ folgt. □

2.4.12 Beispiel. In Beispiel 2.4.05(1) hatten wir $\mathrm{GF}(2^3)$ auf unterschiedliche Weisen realisiert, zunächst als $\mathrm{GF}(2)[x]_{x^3+x+1}$ und dann als $\mathrm{GF}(2)[x]_{x^3+x^2+1}$. Im ersten Fall hatten wir α als formale Wurzel von x^3+x+1 eingeführt, also ist nach Proposition 2.4.07 x^3+x+1 durch $x-\alpha$ teilbar. In der Tat liefert Polynomdivision in $\mathrm{GF}(2^3)[x]$

$$x^3+x+1=(x-\alpha)(x-\alpha^2)(x-\alpha^4)$$

Analog erhält man für die formale Wurzel β von $x^3=x^2+1$

$$x^3+x^2+1=(x-\beta)(x-\beta^2)(x-\beta^4)$$

Die spezielle Form der Faktorisierungen im vorangegangenen Beispiel ist natürlich kein Zufall. Vielmehr handelt es sich bei den über $\mathrm{GF}(2)$ irreduziblen Polynomen x^3+x+1 und x^3+x^2+1 um die Minimalpolynome von α, bzw. β. Zur Erinnerung:

2.4.13 Definition. $\tilde{F}$ sei eine AKE des Körpers F. Unter dem *Minimalpolynom* eines Elements $\alpha\in\tilde{F}$ bzgl. F versteht man das normierte Polynom $M_\alpha^F(x)\in F[x]$ minimalen Grades mit Wurzel α.

Wenn der Grundkörper F außer Zweifel steht, kann der Index F weggelassen und vereinfachend $M_\alpha(x)$ geschrieben werden.

Folgender Satz aus der Algebra charakterisiert Minimalpolynome über endlichen Körpern.

2.4.14 Satz. *Das Minimalpolynom von $\alpha\in\tilde{F}$ ist das Produkt der durch $[a]_F$ bestimmten Linearfaktoren. Falls $F=\mathrm{GF}(q)$ gilt speziell für die kleinste positive Zahl $j\in\mathbb{N}$ mit $\alpha=\alpha^{p^j}$*

$$M_\alpha(x)=\prod_{\beta\in[\alpha]_F}(x-\beta)=\prod_{i<j}\left(x-\alpha^{p^i}\right)$$

Nun sind wir in der Lage, die in vielen Anwendungen nützliche Zyklizität der multiplikativen Gruppen endlicher Körper zu beweisen.

2.4.15 Satz. *Die multiplikative Gruppe eines endlichen Körpers ist zyklisch.*

Beweis: F sei ein endlicher Körper und $\alpha \in F$ ein Element maximaler Ordnung.
Behauptung: Für jedes Element $\beta \in F \setminus \{0\}$ ist $\mathrm{ord}(\alpha)$ durch $\mathrm{ord}(\beta)$ teilbar.
Beweis: Wähle $\beta \in F \setminus \{0\}$ und einen nichttrivialen Primfaktor r von $\mathrm{ord}(\beta)$. Dann existieren Zahlen $m, n, s, t \in \mathbb{N}$ mit

$$\mathrm{ord}(\beta) = r^m \cdot n \quad \text{und} \quad \mathrm{ord}(\alpha) = r^s \cdot t$$

wobei $\mathrm{ggT}(r, n) = \mathrm{ggT}(r, t) = 1$ gilt. Daraus schließen wir

$$\mathrm{ord}(\beta^n) = \frac{\mathrm{ord}(\beta)}{\mathrm{ggT}(\mathrm{ord}(\beta), n)} = \frac{\mathrm{ord}(\beta)}{n} = r^m$$

sowie

$$\mathrm{ord}(\alpha^{r^s}) = \frac{\mathrm{ord}(\alpha)}{\mathrm{ggT}(\mathrm{ord}(\alpha), r^s)} = \frac{\mathrm{ord}(\alpha)}{r^s} = t$$

Wegen $\mathrm{ggT}(r^m, t) = 1$, erhalten wir

$$\mathrm{ord}(\beta^n \alpha^{r^s}) = \mathrm{ord}(\beta^n) \cdot \mathrm{ord}(\alpha^{r^s}) = r^m \cdot t \leq \mathrm{ord}\, \alpha = r^s \cdot t$$

woraus wir $m \leq s$ folgern. Also ist $\mathrm{ord}(\alpha)$ durch r^m teilbar.

Dies funktioniert für jeden nichttrivialen Primfaktor von $\mathrm{ord}(\beta)$, also ist $\mathrm{ord}(\alpha)$ auch durch $\mathrm{ord}(\beta)$ teilbar. Folglich ist β Wurzel des Polynoms $x^{\mathrm{ord}(\alpha)} - 1 \in F[x]$.

Aber dies gilt für jedes Element aus $F \setminus \{0\}$. Also hat $x^{\mathrm{ord}(\alpha)} - 1 \in F[x]$ mindestens $|F| - 1$ viele Linearfaktoren. Mit anderen Worten, $|F| - 1 \leq \deg(x^{\mathrm{ord}(\alpha)} - 1) = \mathrm{ord}(\alpha)$. Andererseits teilt $\mathrm{ord}(\alpha)$ die Gruppenordnung $|F| - 1$, also gilt $\mathrm{ord}(\alpha) = |F| - 1$, und die multiplikative Gruppe von F ist zyklisch. □

3 Einige Schranken für Codes

In diesem Kapitel wollen wir die Größe q des Alphabets F (das nur im linearen Fall ein Körper sein muß), die Codelänge n, die Codegröße M bzw. die Codedimension $k = log_q M$ (letztere ist im linearen Fall garantiert ganzzahlig) und den Minimalabstand d in Beziehung setzen.

Zunächst betrachten wir notwendige Bedingungen für die Parameter (nicht notwendig linearer) Codes. Dazu gehören die Hamming-Schranke, die Singleton-Schranke, die Plotkin-Schranke sowie die Griesmer-Schranke. Dazu präsentieren wir Familien von Codes, die zumindest die ersten drei dieser Schranken erreichen.

Im Gegensatz dazu geht es bei beiden Varianten der Gilbert-Varshamov-Schranke um die Existenz von Codes, deren Parameter bestimmten Bedingungen genügen.

Schranken lassen sich meist für den Minimalabstand d formulieren, oder alternativ für die Codegröße M bzw. für q^{n-k}. Im allgemeinen (nicht notwendig linearen) Fall wird häufig auch die *maximale Codegröße* $A_q(n,d)$ bei vorgebebenen Parametern q, n und d betrachtet, vergl. Definition 3.1.02.

Das wesentliche geometrische Konzept bei der Formulierung der meisten Schranken sind die *Hamming-Kugeln* vom Radius t um $\boldsymbol{c} \in F^n$. Darunter versteht man die Menge aller Wörter mit Hamming-Abstand $\leq t$ von $\boldsymbol{c}$. Ihr *Volumen*, d.h., die Anzahl der enthaltenen Wörter, ist gegeben durch

$$V_q(n,t) = \sum_{i \leq t} \binom{n}{i} (q-1)^i \tag{3.0-21}$$

Die Betrachtung asymptotischer Schranken wird auf später verschoben.

3.1 Maximale und optimale Codes

3.1.01 Definition. Ein (n,k,d)-Code über einem q-elementigen Alphabet F heißt *maximal*, wenn zu jedem Element $\boldsymbol{y} \in F^n$ ein Codewort $\boldsymbol{c}$ existiert, das $\Delta_{\mathrm{H}}(\boldsymbol{y},\boldsymbol{c}) \leq d-1$ erfüllt.

Insbesondere kann zu einem maximalen Code kein neues Codewort aus F^n hinzugefügt werden, ohne den Minimalabstand zu verringern. Offenbar kann jeder $(n,k,d)_q$ Code in endlich vielen Schritten zu einem maximalen (n,k',d) Code erweitert werden, indem man sukzessive Wörter aus F^n hinzufügt, die von allen bisherigen Codewörtern mindestens den Hamming-Abstand d haben. Dies garantiert die Existenz maximaler Codes.

Allerdings ist nicht unmittelbar klar, ob alle maximalen $(n,d)_q$-Codes die gleiche Größe haben, zumal, wenn keine Linearität gefordert ist. Daher benötigen wir einen weiteren Begriff.

3.1.02 Definition. Für ein Alphabet F der Größe q bezeichnet $A_q(n,d)$ die maximale Größe eines $(n,d)_q$-Codes. Ein $(n,k,d)_q$-Code mit $q^k = M = A_q(n,d)$ heißt *optimal*.

Optimale Codes sind natürlich maximal und garantieren die bestmöglichen Coderaten. Angesichts der Schwierigkeit, derartige Codes zu konstruieren, erscheint es aber unwahrscheinlich, daß jeder (n,d)-Code über F zu einem optimalen Code erweitert werden kann.

Die folgende Charakterisierung maximaler Codes mit Hilfe von Hamming-Kugeln geht vermutlich auf Gilbert zurück:

3.1.03 Proposition. (Kugelüberdeckungsschranke) *Jeder maximale $(n,k,d)_q$-Code erfüllt*

$$\frac{q^n}{V_q(n,d-1)} \leq M = q^k \leq A_q(n,d)$$

Beweis: Nach Definition ist die Maximalität äquivalent dazu, daß die Hamming-Kugeln vom Radius $d-1$ um die Codewörter den Raum F^n ausschöpfen. Da diese Kugeln nicht notwendig paarweise disjunkt sein müssen, entspricht dies genau der Ungleichung

$$q^n \leq M \cdot V_q(n,d-1) \qquad \square$$

3.2 Die Hamming- oder Kugelpackungs-Schranke

Ein natürliches Gegenstück zur obigen Kugelüberdeckungsschranke ist gegeben durch

3.2.01 Satz. (Die Hamming- oder Kugelpackungsschranke) *Jeder* $(n,k,d)_q$ *Code erfüllt die Ungleichung*

$$M \cdot V_q\left(n,(d-1)/2\right) \le q^n \quad \text{bzw.} \quad M \le \frac{q^n}{V_q(n,(d-1)/2)} \quad \text{bzw.} \quad V_q\left(n,(d-1)/2\right) \le q^{n-k}$$

woraus unmittelbar folgt

$$A_q(n,d) \le \frac{q^n}{V_q(n,(d-1)/2)}$$

Beweis: Die Hamming-Kugeln vom Radius $t=(d-1)/2$ um die Codewörter sind nach Proposition 0.4.01 paarweise disjunkt, können also nicht mehr als q^n Elemente überdecken. □

Kombination mit der Kugelüberdeckungsschranke liefert

3.2.02 Corollar. *Sind die Länge* $n>0$ *und der Minimalabstand* $0<d\le n+1$ *vorgegeben, so genügt die Maximalgröße* $A_q(n,d)$ *eines* $(n,d)_q$*-Codes den Ungleichungen*

$$\frac{q^n}{V_q(n,d-1)} \le A_q(n,d) \le \frac{q^n}{V_q(n,(d-1)/2)}$$ □

3.2.03 Definition. Ein $(n,k,d)_q$ Code heißt *perfekt*, falls er die Hamming-Schranke mit Gleichheit erfüllt, d.h., wenn die Hamming-Kugeln vom Radius $(d-1)/2$ um die Codewörter den Raum F^n partitionieren.

Leider ist die Anzahl perfekter linearer Codes begrenzt.

3.2.04 Beispiel. Für ungerades n ist der $[n,1,n]_2$ Wiederholungscode perfekt, denn

$$V_2\left(n,(n-1)/2\right) = \sum_{i\le(n-1)/2} \binom{n}{i} = \frac{1}{2}\sum_{i\le n}\binom{n}{i} = 2^{n-1}$$

3.2.05 Beispiel. (Vergl. Beispiel 1.2.08) Für $m>1$ ist der $[n,n-m,3]_q$ Hamming-Code mit $n:=(q^m-1)/(q-1)$ ebenfalls perfekt, denn

$$V_q(n,1) = 1+n(q-1) = q^m = q^{n-k}$$

Die einzigen anderen linearen perfekten Codes sind

- der $[23,12,7]$ Golay-Code über $\mathrm{GF}(2)$, vergl. Beispiele 6.3.15 und 6.5.06(a);
- der $[11,6,5]$ Golay-Code über $\mathrm{GF}(3)$, vergl. Beispiel 6.5.06(b).

3.2.06 Beispiel. Wir betrachten binäre Codes des Minimalabstands 5. Dann gilt

$$V_2(n,2) = 1 + n + \binom{n}{2} = 1 + n + (n-1)n/2 = (n^2+n+2)/2$$

Woraus wir schließen

$$A_2(n,5) \leq \frac{2^n}{(n^2+n+2)/2} = \frac{2^{n+1}}{n^2+n+2}$$

Speziell für $n = 2^m - 1$ erhalten wir

$$A_2(n,5) \leq \frac{2^{2^m-1}}{2^{2m-1} - 2^{m-1} + 1}$$

Wegen $2^k = M \leq A_2(n,5)$ erfüllt jeder $[2^m-1, k, 5]_2$-Code folgende Abschätzung für $k \in \mathbb{N}$

$$k \leq 2^m - 1 - \lceil \log_2(2^{2m-1} - 2^{m-1} + 1) \rceil = 2^m - 1 - (2m-1) = 2^m - 2m$$

3.3 Die Singleton-Schranke

Eine weitere nützliche Abschätzung des Minimalabstands d bzw. der Codegröße M nach oben ist gegeben durch

3.3.01 Satz. (Die Singleton-Schranke [6]) *Jeder $(n,k,d)_q$ Code erfüllt*

$$d \leq n - k + 1 \quad \text{bzw.} \quad k \leq n - d + 1$$

woraus unmittelbar folgt

$$A_q(n,d) \leq q^{n-d+1}$$

Beweis: Setze $l := \lceil k \rceil - 1 \in \mathbb{N}$. Wegen $l < k$ folgt $q^l < q^k = M$, daher existieren zwei verschiedene Codewörter, die in den ersten l Komponenten übereinstimmen. Folglich gilt $d \leq n - l$. Auflösen von $\log_q M = k \leq \lceil k \rceil \leq n - d + 1$ nach M liefert $M \leq q^{n-d+1}$. Dies gilt für jeden $(n,k,d)_q$-Code, was die zweite Ungleichung impliziert. □

Im linearen Fall ist diese Ungleichung auch Konsequenz von Satz 1.2.01: der (Zeilen-)Rang einer Kontrollmatrix hat den Wert $n-k$, daher müssen $n-k+1$ linear abhängige Zeilen existieren. Also kann d höchstens den Wert $n-k+1$ annehmen. Alternativ kann man auch eine bis auf Spaltenpermutation systematische Generatormatrix für den Code betrachten: das Hamming-Gewicht jeder Zeile hat höchstens den Wert $n-k+1$ und beschränkt somit d nach oben.

3.3.02 Beispiel. Für $n = 4$ und $d = 3$ besagen die Singleton- und die Kugelpackungs-Schranke

$$A_q(n,d) \leq q^2 \quad \text{bzw.} \quad A_q(n,d) \leq \frac{q^4}{4q-3} \in O\left(q^3\right)$$

Schon für $q \geq 4$ die Singleton-Schranke viel besser als die Kugelpackungs-Schranke.

3.3.03 Definition. Ein $(n,k,d)_q$ Code heißt *maximum distance separable*, oder kurz *MDS*, falls er die Singleton-Schranke mit Gleichheit erfüllt.

3.3.04 Beispiel. Folgende Codes sind MDS:

- der $[n,n,1]_q$-Identitätscode, denn $1 = n - n + 1$;
- der $[n,n-1,2]_q$-Paritätscode, denn $2 = n - (n-1) + 1$;
- der $[n,1,n]_q$-Wiederholungscode, denn $n = n - 1 + 1$.

Andererseits handelt es sich bei binären Hamming Codes um $[2^m - 1, 2^m - 1 - m, 3]_2$-Codes (vergl. Beispiel 1.2.06); hier nimmt die Singelton-Schranke die Form $3 \leq m+1$ an. Nur für den Minimalwert $m = 2$ erhält man einen MDS Code.

Wir führen nun eine weitere wichtige Familie von MDS Codes ein.

3.3.05 Definition. q sei eine Primzahlpotenz und $n \leq q$. Paarweise verschiedene Elemente α_i, $i < n$, des Körpers $F := \mathrm{GF}(q)$ spezifizieren einen *(normalisierten verallgemeinerten) Reed Solomon Code* über F als linearen $[n,k]$-Code mit Kontrollmatrix

$$H_{\mathrm{RS}} = \begin{pmatrix} 1 & \alpha_0 & \alpha_0^2 & \dots & \alpha_0^{n-k-1} \\ 1 & \alpha_1 & \alpha_1^2 & \dots & \alpha_1^{n-k-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha_{n-1} & \alpha_{n-1}^2 & \dots & \alpha_{n-1}^{n-k-1} \end{pmatrix}$$

Die nicht normalisierte Variante verallgemeinerter Reed-Solomon Codes werden wir in Kapitel 4 kennenlernen.

3.3.06 Proposition. *Jeder normalisierte verallgemeinerte Reed Solomon Code ist MDS.*

Beweis: Jede quadratische Teilmatrix der Dimension $n-k$ von H_{RS} ist eine *Vandermonde* Matrix der Form

$$B = \begin{pmatrix} 1 & \beta_0 & \beta_0^2 & \dots & \beta_0^{n-k-1} \\ 1 & \beta_1 & \beta_1^2 & \dots & \beta_1^{n-k-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \beta_{n-k-1} & \beta_{n-k-1}^2 & \dots & \beta_{n-k-1}^{n-k-1} \end{pmatrix}$$

wobei die β_i, $i < n$, verschiedene Körperelemente sind. Wie aus der linearen Algebra bekannt, gilt nun

$$\det B = \prod_{i<j<n-k} (\beta_j - \beta_i) \neq 0$$

weshalb B invertierbar ist. Folglich ist jede Auswahl von $n-k$ Zeilen von H_{RS} linear unabhängig, woraus nach Satz 1.2.01 $d \geq n-k+1$ folgt, aufgrund der Singelton-Schranke also $d = n-k+1$.

□

3.4 Die Plotkin-Schranke

3.4.01 Satz. (Die Plotkin-Schranke [4]) *Jeder* $(n,k,d)_q$ *Code erfüllt*

$$d \le n \cdot \frac{(q-1)M}{q(M-1)} \quad \text{bzw.} \quad M \le \frac{qd}{qd-n(q-1)}$$

woraus unmittelbar folgt

$$A_q(n,d) \le \frac{qd}{qd-n(q-1)}$$

Beweis: Für einen beliebigen $(n,k,d)_q$ Code $\mathcal{C}$ schätzen wir die Summe $\sum_{\boldsymbol{c},\boldsymbol{c}'\in\mathcal{C}} \Delta_{\mathrm{H}}(\boldsymbol{c},\boldsymbol{c}')$ in beiden Richtungen ab. Nach Definition des Minimalabstands gilt

$$M(M-1)d \le \sum_{\boldsymbol{c},\boldsymbol{c}'\in\mathcal{C}} \Delta_{\mathrm{H}}(\boldsymbol{c},\boldsymbol{c}')$$

Umgekehrt wird die Wahrscheinlichkeit, daß sich verschiedene Codewörter in der festen Komponente $i < n$ unterscheiden, genau dann maximiert, wenn die Elemente des Körpers F in dieser Komponente der Codewörter gleichverteilt sind: nach Wahl eines der M Codewörter stehen $M(q-1)/q$ Codewörter mit einem anderen Wert in Komponente i zur Verfügung, und jedes solche Paar trägt den Wert 1 zur obigen Summe bei. Diese Überlegung gilt für alle Komponenten, also

$$\sum_{\boldsymbol{c},\boldsymbol{c}'\in\mathcal{C}} \Delta_{\mathrm{H}}(\boldsymbol{c},\boldsymbol{c}') \le n \cdot \frac{q-1}{q} \cdot M^2$$

Die Kombination beider Ungleichungen liefert

$$d \le n \cdot \frac{(q-1)M}{q(M-1)} \quad \text{bzw.} \quad M \le \frac{qd}{qd-n(q-1)}$$

Dies gilt für jeden $(n,k,d)_q$-Code, also resultiert die gewünschte Ungleichung für $A_q(n,d)$. □

Im linearen Fall nimmt die erste Variante der Plotkin-Schranke für einen $[n,k,d]_q$-Code folgende Form an

$$d \le n \cdot \frac{(q-1)q^{k-1}}{q^k-1}$$

Dabei kann die rechte Seite als durchschnittliches Hamming-Gewicht der von $\boldsymbol{0}$ verschiedenen Codewörter interpretiert werden; nicht alle können ein echt größeres Hamming-Gewicht haben.

Um lineare Codes zu finden, die die Plotkin-Schranke mit Gleichheit erfüllen, beschreiten wir einen kleinen Umweg.

3.4.02 Definition. Für $F = \mathrm{GF}(q)$ und $m > 0$ setze $n := q^M$. Ein *Reed-Muller Code erster Ordnung* über F ist ein linearer $[n, m+1]$ Code $\mathcal{C}$, für den alle Spalten-Vektoren in F^{m+1} mit einer Eins in der ersten Position eine $(m+1) \times n$-Generatormatrix bilden. Streichen der ersten Zeile und der resultierenden Null-Spalte liefert eine $m \times (n-1)$-Generatormatrix des zugehörigen *verkürzten* Reed-Muller-Codes erster Ordnung über F.

3.4.03 Proposition. *$\mathcal{C}$ sein ein $[n = q^m, m+1]$-Reed-Muller-Code erster Ordnung über $F = \mathrm{GF}(q)$, und $\mathcal{C}'$ sei seine Verkürzung.*

(0) *$\mathcal{C}$ besitzt genau $q-1$ Codewörter des Gewichts $n = q^m$, während alle übrigen von $\boldsymbol{0}$ verschiedenen Codewörter das Gewicht $q^{m-1}(q-1)$ aufweisen, was mit dem Minimalabstand d übereinstimmt.*

(1) *Der Minimalabstrand jedes linearen $[n, m+1]$-Code über $F = \mathrm{GF}(q)$ ist durch $q^{m-1}(q-1)$ nach oben beschränkt.*

(2) *$\mathcal{C}'$ entsteht aus $\mathcal{C}$ mittels der Verkürzungsoperation in Definition 1.2.09(3) und alle von $\boldsymbol{0}$ verschiedenen Codewörter haben das Gewicht $q^{m-1}(q-1)$, was den Minimalabstand d' bestimmt. $\mathcal{C}'$ erfüllt die Plotkin-Schranke mit Gleichheit.*

Beweis:

(0) Jedes Codewort hat die Form $\boldsymbol{c} = \boldsymbol{u} \cdot G$ für ein $\boldsymbol{u} \in F^m$, ist also eine Linearkombination der Zeilen von G. Unter den nicht-trivialen Linearkombinationen liefern diejenigen der Form $(a\,\boldsymbol{0}) \cdot G$ mit $a \in F \setminus \{0\}$ aufgrund der speziellen Form von G Codewörter des Gewichts $n = q^m$. Ihre Anzahl beträgt also $q-1$.

Die letzten m Zeilen von G haben alle das Gewicht $q^{m-1}(q-1)$, da in ihnen die Elemente von F gleichverteilt sind. Betrachte eine nicht-triviale Linearkombination $\boldsymbol{y} = (a\ \boldsymbol{v}) \cdot G$ mit $\boldsymbol{v} \in F^m \setminus \{\boldsymbol{0}\}$. Die Operation $F^m \xrightarrow{\boldsymbol{v} \cdot -} F$ ist linear und nach Voraussetzung surjektiv. Damit haben alle q Nebenklassen dieselbe Größe wie der Kern dieser Abbildung, nämlich q^{m-1}. Folglich sind in der Linearkombination der letzten m Zeilen die Elemente von F immer noch gleichverteilt, woran sich auch durch Addition eines Vielfachen der ersten Zeile nichts ändert. Demnach haben all diese Linearkombinationen das Gewicht $q^{m-1}(q-1)$, was somit den Minimalabstand bestimmt.

(1) Die ganze Zahl $q^{m-1}(q-1)$ erfüllt

$$q^{m-1}(q-1) = \frac{q^{2m}(q-1)}{q^{m+1}} < \frac{q^{2m}(q-1)}{q^{m+1}-1} < \frac{(q^{2m}+q^m)(q-1)+q^{m-1}-1}{q^{m+1}-1} = q^{m-1}(q-1)+1$$

Da die Plotkin-Schranke für $n = q^m$ und $k = m+1$ (der dritte Ausdruck) echt zwischen $q^{m-1}(q-1)$ und $q^{m-1}(q-1)+1$ liegt, ist $q^{m-1}(q-1)$ der maximal mögliche Minimalabstand eines $[n, m+1]_q$-Codes.

(2) Gemäß Proposition 1.2.11 entsteht die neue Generatormatrix G' aus der Generatormatrix G durch Entfernen der ersten Zeile (aus lauter Einsen) und anschließendes Entfernen der entstandenen Null-Spalte, etwa in Position i. Da die i-te Komponente jeder Linearkombinationen der letzten m Zeilen von G immer den Wert Null hat, ergibt sich $\mathcal{C}'$ als i-Verkürzung von $\mathcal{C}$ gemäß Definition 1.2.09(3).

Nach Teil (0) hat nun jedes nichttriviale Wort von $\mathcal{C}'$ das Gewicht $q^{m-1}(q-1)$. Mit $n = q^m - 1$ und $k = m$ wird in diesem Fall die Plotkin-Schranke genau angenommen:

$$q^{m-1}(q-1) = \frac{(q^m-1)(q-1)q^{m-1}}{q^m-1}$$ □

3.5 Die Griesmer-Schranke

Der Beweis von Teil (0) der Proposition 3.4.03 motiviert eine weitere notwendige Schranke, die im Gegensatz zu den bisherigen Schranken die Code-Länge beschränkt und ohne Bezug zum Volumen gewisser Hamming-Kugeln formuliert ist.

3.5.01 Satz. (Die Griesmer-Schranke [2]) *Jeder lineare* $[n,k,d]_q$*-Code erfüllt*

$$n \geq \sum_{j<k} \left\lceil \frac{d}{q^j} \right\rceil$$

Beweis: Für einen $[n,k,d]_q$-Code $\mathcal{C}$ wählen wir $\boldsymbol{c}_0 \in \mathcal{C}_0$ mit $\omega(\boldsymbol{c}_0) = d$. O.B.d.A. mögen die ersten d Komponenten von $\boldsymbol{c}_0$ von 0 verschieden sind, ansonsten permutieren wir die Spalten entsprechend. Damit ist $\boldsymbol{c}_0$ die Konkatenation aus $\boldsymbol{c}_{00} \in (F \setminus \{0\})^d$ und $\boldsymbol{0} \in F^{n-d}$.

Eine Generatormatrix G_0 von $\mathcal{C}_0$ mit $\boldsymbol{c}_0$ als erster Zeile muß wie folgt zerlegbar sein

$$G_0 = \left(\begin{array}{c|c} \boldsymbol{c}_{00} & \boldsymbol{0} \\ \hline A_1 & G_1 \end{array}\right)$$

Die $(k-1) \times (n-d)$-Teilmatrix G_1 erzeugt einen entsprechenden linearen Code $\mathcal{C}_1$ mit Minimalabstand d_1.

Für die erste Zeile von G_1 wählen wir wieder o.B.d.A. $\boldsymbol{c}_{11} \in \mathcal{C}_1$ so, daß die ersten d_1 Komponenten von 0 verschieden sind. Nach Konstruktion existiert $\boldsymbol{c}_{10} \in F^d$ so daß die Konkatenation $\boldsymbol{c}_1 := \boldsymbol{c}_{01}\boldsymbol{c}_{11}$ ein Codewort ist und folglich $\omega(\boldsymbol{c}_1) \geq d$ erfüllt. Mit anderen Worten: bildet die erste Zeile $\boldsymbol{c}_{10}$ der Teilmatrix A_1 darf höchstens d_1 viele Nullen enthalten.

Jede Linearkombination $\boldsymbol{c}_0 + a\boldsymbol{c}_1$ mit $a \in F$ ist ein von $\boldsymbol{0}$ verschiedenes Codewort und hat folglich mindestens das Hamming-Gewicht d. Insbesondere dürfen für jedes $a \in F \setminus \{0\}$ höchstens d_1 Positionen in $\boldsymbol{c}_{00} + a\boldsymbol{c}_{10}$ den Wert 0 annehmen. Aber diese Postionen sind für verschiedene Werte $a \in F \setminus \{0\}$ disjunkt, und ebenfalls disjunkt von den Positionen, in denen Nullen in $\boldsymbol{c}_{10}$ auftreten. Daher ist d_1 durch $\lceil d/q \rceil$ nach unten beschränkt.

Spezifiziert man nach demselben Schema eine zweite Zeile von G_1, und damit eine dritte Zeile von G_0, so kommen mindestens $\lceil (\lceil d/q \rceil)/q \rceil = \lceil d/q^2 \rceil$ neue Positionen hinzu, in denen erstmals ein Wert aus $F \setminus \{0\}$ auftritt. Induktiv ist somit die Anzahl der Zeilen von G_0 dadurch beschränkt, daß die Summe der Positionen, in denen erstmals keine Null auftritt, die Spaltenzahl, und somit die Codelänge nicht überschreiten kann. □

Die Konstruktion linearer Codes, die die Griesmer-Schranke mit Gleichheit erfüllen, gestaltet sich schwieriger als in den vorangegangenen Fällen. Für ein neueres Ergebnis in dieser Hinsicht verweisen wir den Leser auf [1].

3.6 Die Gilbert-Varshamov-Schranke

Nach drei notwendigen Bedingungen für die Parameter eines jeden Codes, betrachten wir nun Parameter-Bedingungen für die die Existenz entsprechender Codes gezeigt werden kann. Irritierenderweise wird der Name "Gilbert-Varshamov-Schranke" in der Literatur für unterschiedliche Ergebnisse verwendet, deren Zusammenhang nicht immer explizit aufgezeigt wird.

Wir beginnen mit einer Umformulierung der Kugelüberdeckungsschranke, Proposition 3.1.03.

3.6.01 Corollar. (Erste Variante der Gilbert-Varshamov-Schranke) *Für positive natürliche Zahlen* q, n, d *mit* $d \leq n+1$ *existiert ein* $(n,d)_q$*-Code der Größe* $M = q^k$, *so daß*

$$\frac{q^n}{V_q(n,d-1)} \leq M = q^k \qquad \text{bzw.} \qquad q^{n-k} \leq V_q(n,d-1)$$

Beweis: Ausgehend von einem beliebigen $(n,d)_q$ Code genügt es, einen maximalen $(n,k,d)_q$ Code zu konstruieren. □

Im Gegensatz zu den notwendigen Bedingungen können wir hier keine direkte Spezialisierung auf den linearen Fall vornehmen; es ist nicht unmittelbar klar, ob es unter den gegebenen Voraussetzungen auch einen linearen $(n,k,d)_q$-Code gibt mit $q^{n-k} \leq V_q(n,d-1)$ Dies wurde jedoch von Varshamov bewiesen. Um ein noch etwas besseres Ergebnis zu erhalten, wird zunächst die Konstruktion des allgemeinen Hamming-Codes aus Beispiel 1.2.08 auf Minimalabstände ≥ 3 ausgedehnt. Die folgende hinreichende Bedingung für die Existenz linearer Codes, das von Roth [5] als Gilbert-Varshamov-Schranke bezeichnet wird, geht anderen Quellen zufolge auf Sergey Yekhanin zurück:

3.6.02 Proposition. *Erfüllen natürliche Zahlen* $q, n, d > 1$ *und* $m \leq n$ *die Ungleichung*

$$V_q(n-1,d-2) < q^m$$

so existiert ein linearer $[n, k := n-m]$*-Code mit Minimalabstand* $\geq d$.

Beweis: Ziel: ausgehend von der $(m \times m)$-Einheitsmatrix I_m konstruieren wir durch Anfügen neuer Zeilen eine $(n \times m)$-Matrix H_n, die folgender Bedingung genügt

$$\text{jede Auswahl von } d-1 \text{ Zeilen ist linear unabhängig} \tag{3.6-22}$$

Da $V_q(n-1,d-2)$ gemäß der Volumenfunktion (3.0-21) den Summanden $\binom{n-1}{d-2}q^{d-2}$ enthält, folgt sofort $d-2 < m$. Insofern erfüllt $H_m = I_m$ Bedingung (3.6-22).

Wir nehmen an, (3.6-22) gilt für H_l bestehend aus $m \leq l$ Zeilen $h_i \in F^m$, $i < l$. Ein Vektor $h_l \in F^m$ kann genau dann als neue Zeile hinzugefügt werden ohne (3.6-22) zu verletzen, wenn er *nicht* Linearkombination von $\leq d-2$ bereits vorhandenen Vektoren ist. Äquivalent sind Vektoren der Form $\boldsymbol{u} \cdot H_l$ genau dann als Ergänzung unzulässig, wenn $\boldsymbol{u} \in F^l$ maximal ein Hamming-Gewicht von $d-2$ hat. Aber die Anzahl solcher Vektoren $\boldsymbol{u}$ ist durch $V_q(l, d-2)$ gegeben, was eine obere Schranke für die Anzahl der verbotenen Produkte darstellt (die Rechtsmultiplikation mit H_l braucht nicht injektiv zu sein). Aus $V_q(l, d-2) < q^m$ folgt also die Existenz eines Vektors $h_l \in F^m$, so daß H_{l+1} die Bedingung (3.6-22) erfüllt.

Aus der Voraussetzung $V_q(n-1, d-2) < q^m$ folgt wegen der Monotonie der Volumenfunktion (3.0-21) in der ersten Variable $V_q(l, d-2) < q^m$ für $m \leq l < n-1$, und somit die Existenz einer Matrix H_n mit der gewünschten Eigenschaft. Diese dient als als Kontrollmatrix für einen linearen $[n, n-m]$-Code $\mathcal{C}$, der nach Satz 1.2.01 einen Minimalabstand $\geq d$ hat. □

3.6.03 Corollar. (Zweite Variante der Gilbert-Varshamov-Schranke) *Für eine Primzahlpotenz q und natürliche Zahlen d, m mit $d \leq m+1$ existiert eine Zahl $n \geq m$ und ein $[n, k := n-m, d]_q$-Code, dessen Parameter folgender Ungleichung genügen*

$$q^{n-k} \leq V_q(n, d-2)$$

Beweis: Für vorgegebenes m konstruieren wir analog zum Beweis von Proposition 3.6.02 eine m-spaltige Matrix H maximaler Zeilenzahl, in der je $d-1$ Zeilen linear unabhängig sind.

Falls $d = m+1$ erhalten wir $H = I_m$, woraus $\mathcal{C} = \{\boldsymbol{0}\}$ folgt, und somit $n = m$ sowie $k = 0$, was mit der Vereinbarung $d = n+1$ aus Abschnitt 0.2 konsistent ist.

Andernfalls wählen wir für h_m die Summe der ersten $d-1$ Zeilen der Einheitsmatrix I_m und fügen dann solange Zeilen hinzu, wie die Bedingung (3.6-22) erfüllt ist. Die resultierende maximale Matrix H mit dieser Eigenschaft hat nach Konstruktion d linear abhängige Zeilen, der resultierende Code $\mathcal{C}$ also den Minimalabstand d.

Die Anzahl der Zeilen von H bezeichen wir mit n und setzen $k := n-m$. Aufgrund der Maximalität von H können die Parameter nicht der Bedingung $V_q(n, d-2) < q^{n-k}$ aus Proposition 3.6.02 genügen. Stattdessen muß $q^{n-k} \leq V_q(n, d-2)$ gelten. □

Wegen $V_q(n, d-2) < V_q(n, d-1)$ erfüllt der eben konstruierte lineare Code natürlich auch die erste Variante der Gilbert-Varshamov-Schranke.

Á priori ist weder klar, ob die Codelänge n in der zweiten Variante der Gilbert-Varshamov-Schranke eindeutig bestimmt ist, noch, ob der Minimalabstand des resultierenden Codes auch dann den Wert d hätte, wenn man nicht explizit für die Existenz von d linear abhängigen Zeilen in H sorgt. In der Tat sind beide Vermutungen falsch. Das folgende Gegenbeispiel stammt im Wesentlichen von Jyrki Lahtonen:

3.6.04 Beispiel. Für $q = 2$ und $m = 6$ betrachten wir folgende systematische Kontrollmatrizen:

$$H_0 := \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \quad \text{und} \quad H_1 := \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Gemäß Beispiel 1.2.02 ist H_0 eine Kontrollmatrix des Wiederholungscodes der Länge 7, dessen Minimalabstand auch den Wert 7 hat. Somit ist jede Auswahl von < 7 Zeilen von H_0 linear unabhängig.

Andererseits ist H_0 maximal mit der Eigenschaft, das je 4 Zeilen linear unabhängig sind: wäre das nicht der Fall, könnte $\boldsymbol{x} \in \{0,1\}^6$ hinzugefügt werden, ohne diese Eigenschaft zu verletzen. Falls $\omega(\boldsymbol{x}) \leq 3$, ist $\boldsymbol{x}$ Linearkombination von drei der ersten Zeilen von H_0. Andernfalls hat die Summe aus $\boldsymbol{x}$ und der letzten Zeile ein Gewicht ≤ 2. In jedem Fall existieren 4 linear abhängige Zeilen, Widerspruch. Wegen $V_2(6,3) = 1+6+15+20 = 42 < 64 = 2^6$ kann H_0 bei vorgegebenem Parameter $d = 5$ gemäß des Beweises von Proposition 3.6.02 konstruiert werden und liefert in der Tat einen linearen Code mit Minimalabstand ≥ 5. Andererseits gilt $V_2(7,3) = 1 + 7 + 21 + 35 = 64$, so daß Proposition 3.6.02 nicht für den Parameter $n = 8$ anwendbar ist.

Der durch H_1 spezifizierte $[8,2]_2$-Code hat die systematische Generatormatrix

$$G_1 = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

woraus man unmittelbar den Minimalabstand von 5 ablesen kann. Folglich kann H_1 tatsächlich im Verlauf der Konstruktion im Beweis zu Korollar 3.6.03 entstehen.

Es bleibt nachzuweisen, daß H_1 maximal ist mit der Eigenschaft, daß je 4 Zeilen linear unabhängig sind. Aufgrund der Griesmer-Schranke (Satz 3.5.01) kann kein $[9,3,5]_2$-Code existieren, denn $9 < \sum_{i<3} \lceil 5/q^i \rceil = 5 + 3 + 2 = 10$.

Gemäß Korollar 3.6.03 existiert also ein $[8,2,5]_2$-Code, und dessen Parameter erfüllen $64 = 2^6 \leq V_2(8,3) = 1 + 8 + 28 + 56 = 93$.

3.6.05 Bemerkung. In Kombination mit der Hamming-Schranke folgt die Existenz von linearen $[n,k,d]_q$-Codes mit

$$V_q(n,(d-1)/2) \leq q^{n-k} \leq V_q(n,d-2)$$

Im Fall $d = 3$ gilt aber $(d-1)/2 = d-2$, was die nach dem obigen Schema konstruierten Codes, nämlich die Hamming Codes, zu perfekten Codes macht.

4 Reed-Solomon Codes und verwandte Codes

Verallgemeinerte Reed-Solomon Codes (vRS-Codes) und ihre Abkömmlinge dürften die in der Praxis am häufigsten auftretenden Codes sein. Dafür sind ihre diversen Vorteile verantwortlich. Es handelt sich zunächst einmal um MDS-Codes, die die Singleton Schranke erreichen. Als lineare Codes sind sie effizient codierbar, und für die Teilklasse der konventionellen Reed-Solomon Codes existieren sehr einfache Schaltkreise für die Decodierung. Schließlich lassen sich auch vRS Codes effizient decodieren.

Ein potentieller Nachteil verallgemeinerter Reed-Solomon Codes könnte darin gesehen werden, daß ihre Codelänge durch die Größe des jeweiligen Körpers nach oben beschränkt wird. Daraus könnte man schließen, diese Codes wären nur dann nützlich, wenn Anwendungen nach relativ großen Körpern verlangen, etwa nach $\mathrm{GF}(2^8)$, wenn mit Bytes gerechnet werden soll. Dennoch eignen sich vRS Codes auch als Bausteine für Codes über kleinen Körpern. Da ist zunächst die "Konkatenation" zu nennen, die auf einer zweistufigen Codierung mit vRS Codierer in der ersten Stufe basiert. Weiterhin kann man vRS Codes über hinreichend großen algebraischen Körpererweiterungen von F betrachten und sich dann auf die Codewörter beschränken, deren Komponenten in F liegen. Derartige Codes heißen "alternant", und falls man mit einem konventionellen Reed-Solomon Code startet, liefern sie die bekannten Codes von Bose-Chaudhuri-Hocquenghem, oder kurz BCH-Codes.

4.1 Verallgemeinerte Reed-Solomon-Codes

4.1.1 Definition

Nachdem wir die normalisierte Variante verallgemeinerter Reed-Solomon Codes schon in Definition 3.3.05 eingeführt hatten, wollen wir uns nun dieser Codeklasse in voller Allgemeinheit zuwenden.

4.1.01 Definition. q sei eine Primzahlpotenz und $n \leq q$. Ein *verallgemeinerter Reed-Solomon Code* (*vRS Code*) über $F := \mathrm{GF}(q)$ ist ein linearer $[n,k]$-Code, der spezifiziert ist durch

- einen Vektor $\alpha \in F^n$, mit paarweise verschiedenen Komponenten, genannt *Lokalisierer*;
- einem Vektor $\boldsymbol{v} \in (F \setminus \{0\})^n$ aus *Gewichten*
- eine Kontrollmatrix der Form

$$H_{\mathrm{vRS}} = \begin{pmatrix} v_0 & & & 0 \\ & v_1 & & \\ 0 & & \ddots & \\ & & & v_{n-1} \end{pmatrix} \begin{pmatrix} 1 & \alpha_0 & \alpha_0^2 & \dots & \alpha_0^{n-k-1} \\ 1 & \alpha_1 & \alpha_1^2 & \dots & \alpha_1^{n-k-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha_{n-1} & \alpha_{n-1}^2 & \dots & \alpha_{n-1}^{n-k-1} \end{pmatrix} \tag{4.1-23}$$

Ein verallgemeinerter Reed-Solomon Code heißt

- *primitiv*, falls $n = q-1$ gilt und alle Komponenten von α von 0 verschieden sind;

- *normalisiert*, falls alle Gewichte den Wert 1 haben;

- vRS Code im *engeren Sinn*, falls $\alpha = \boldsymbol{v}$ gilt (was insbesondere $n < q$ impliziert).

4.1.02 Bemerkung. Besondere Erwähnung verdienen verallgemeinerte Reed-Solomon Codes, bei denen (genau) ein Lokalisierer den Wert 0 annimmt. Die entsprechende Zeile des zweiten Faktors der Kontrollmatrix enthält dann nach der führenden Eins lauter Nullen. Gilt beispielsweise $\alpha_0 = 0$, so kann H_{vRS} wie folgt umgeschrieben werden

$$\begin{pmatrix} v_0 & & & \\ & v_1\alpha_1 & & 0 \\ 0 & & \ddots & \\ & & & v_{n-1}\alpha_n \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ \alpha_1^{-1} & 1 & \alpha_1 & \dots & \alpha_1^{n-k-2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \alpha_{n-1}^{-1} & 1 & \alpha_{n-1} & \dots & \alpha_{n-1}^{n-k-2} \end{pmatrix}$$

Damit ist H_{vRS} die Kontrollmatrix der Erweiterung eines vRS Codes (vergl. Proposition 1.2.10), dessen Lokalisierer mit den von 0 verschiedenen Lokalisierern des ursprünglichen Codes übereinstimmen.

Insofern werden gelegentlich bei der Definition verallgemeinerter Reed-Solomon Codes von 0 verschiedene Lokalisierer verlangt, denn der Fall eines Lokalisierer 0 läßt sich leicht per Erweiterung erledigen. Man spricht dann auch von *einfach erweiterten* vRS Codes.

Leider sind die Lokalisierer für einen vRS Code nicht eindeutig bestimmt, nicht einmal bis auf Skalierung der Gewichte (vergl. Hausaufgabe).

4.1.03 Proposition. *Jeder vRS Code ist ein MDS Code, d.h., erfüllt die Singleton Schranke* $d = n - k + 1$.

Beweis: In Proposition 3.3.06 war dies gezeigt worden, wenn alle Gewichte den Wert 1 haben. Aber die Multiplikation der Zeilen der Kontrollmatrix mit von 0 verschiedenen Faktoren hat keinen Einfluß auf den Minimalabstand. □

Aus den Hausaufgaben wissen wir bereits, daß ein linearer Code genau dann MDS ist, wenn dies für den dualen Code gilt. Folglich sind auch die Dualcodes von vRS Codes MDS. Aber wir können sie sogar als vRS Codes identifizieren.

4.1.04 Proposition. *Der Dualcode eines* $[n, k]$ *vRS Codes ist ein* $[n, n-k]$ *vRS Code; beide Codes können mittels derselben Lokalisierer definiert werden.*

Beweis: Der $[n, k]$ vRS Code $\mathcal{C}$ habe die Kontrollmatrix H_{vRS} (4.1-23). Wir zeigen, daß es einen Vektor $\boldsymbol{v}' \in (F \setminus \{0\})^n$ gibt, so daß die Matrix

$$G_{\text{vRS}} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ \alpha_0 & \alpha_1 & \dots & \alpha_{n-1} \\ \alpha_0^2 & \alpha_1^2 & \dots & \alpha_{n-1}^2 \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_0^{k-1} & \alpha_1^{k-1} & \dots & \alpha_{n-1}^{k-1} \end{pmatrix} \begin{pmatrix} v'_0 & & & \\ & v'_1 & & 0 \\ 0 & & \ddots & \\ & & & v'_{n-1} \end{pmatrix} \tag{4.1-24}$$

orthogonal zu H_{vRS} ist, d.h., $G_{\text{vRS}} \cdot H_{\text{vRS}} = 0$ gilt. Damit ist $(G_{\text{vRS}})^\top$ Kontrollmatrix des Dualcodes mit denselben Lokalisierern.

Die definierenden Bedingungen

$$\sum_{j<n} \alpha_j^i v'_j v_j \alpha_j^l = \sum_{j<n} v'_j v_j \alpha_j^{i+l} = 0 \qquad \text{für } i+l < n-1 \tag{4.1-25}$$

lassen sich in Matrixform auch wie folgt darstellen

$$(v'_0\, v'_1\, \ldots\, v'_{n-1}) \begin{pmatrix} v_0 & & & \\ & v_1 & & 0 \\ & & \ddots & \\ & 0 & & v_{n-1} \end{pmatrix} \begin{pmatrix} 1 & \alpha_0 & \alpha_0^2 & \ldots & \alpha_0^{n-2} \\ 1 & \alpha_1 & \alpha_1^2 & \ldots & \alpha_1^{n-2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha_{n-1} & \alpha_{n-1}^2 & \ldots & \alpha_{n-1}^{n-2} \end{pmatrix} = \boldsymbol{0}$$

Möglichen Lösungen sind somit alle Codewörter eines $[n, 1, n]$ vRS Codes über F mit denselben Lokalisierern und Gewichte wie $\mathcal{C}$. Von $\boldsymbol{0}$ verschiedene Codewörter haben das Hamming-Gewicht n und können zur Definition von G_{vRS} verwendet werden. □

4.1.05 Beispiel. $\boldsymbol{v}$ beschreibe die Gewichte eines primitiven vRS Codes über $F = \mathrm{GF}(q)$. Wir zeigen, daß die Gewichte des Dualcodes als $v'_j := \alpha_j / v_j$, $j < n$ definiert werden können. Dazu wählen wir eine primitive Wurzel β aus F. Wegen $\beta^n = \beta^{q-1} = 1$ können die Gleichungen (4.1-25) dann wie folgt umgeschrieben werden:

$$\sum_{j<n} v'_j v_j \alpha_j^r = \sum_{j<n} \alpha_j^{r+1} = \sum_{j<n} \beta^{j(r+1)} = \frac{\beta^{n(r+1)} - 1}{\beta^{r+1} - 1} = 0 \qquad \text{für } r < n-1$$

Insbesondere ist der Dualcode eines normalisierten primitiven vRS Codes ein vRS Code im engeren Sinne.

4.1.2 Polynominterpretation verallgemeinerter Reed-Solomon Codes

Hat $\mathcal{C}$ die Generatormatrix (4.1-24), so liefert die Interpretation eines Informationswortes $\boldsymbol{u} \in F^k$ als Polynom $u(x) \in F[x]$ geordnet nach *steigenden* Potenzen von x

$$\boldsymbol{u} \cdot G_{\text{vRS}} = \big(u(\alpha_i) \cdot v'_i : i < n \big) \tag{4.1-26}$$

D.h., die Eingabe $u(x)$ wird an den Lokalisierern ausgewertet (was deren Namen rechtfertigt), und die Ergebnisse werden entsprechend gewichtet.

Wenn man es vorzieht, aus technischen Gründen, etwa zwecks systematischer Codierung, einem Informationswort $\boldsymbol{u} \in F^k$ ein nach *fallenden* Potenzen von x geordnetes Polynom $\tilde{u}(x) \in F_k[x]$ zuzuordnen, so ist die Reihenfolge der Zeilen von G_{vRS} umzukehren, damit eine vernünftige Polynominterpretation gewährleistet bleibt. Mit anderen Worten, die genaue Form von G_{vRS} und entsprechend H_{vRS} werden durch die Art der Übersetzung von Vektoren (oder

Tupeln) in Polynome bestimmt. Andererseits ist die rechte Seite von (4.1-26) unabhängig von der Darstellung des Polynoms $u(x) \in F_k[x]$ als Tupel.

Die Polynominterpretation liefert ein weiteres Argument, warum vRS Codes MDS sind: ein Polynom $u(x) \in F_k[x] \setminus \{0\}$ hat maximal $\operatorname{grd} u(x) \leq k-1$ verschiedene Wurzeln, was die Anzahl der Nullkomponenten von $\boldsymbol{u} \cdot G_{\mathrm{vRS}}$ durch $k-1$ nach oben, und somit das Hamming-Gewicht durch $n-(k-1)$ nach unten beschränkt.

Ist d ungerade, etwa $d = 2t+1$, so folgt $n = k+2t$. Um bis zu $t = (d-1)/2$ Fehler zu decodieren, gilt es, aus der potentiell in t Stellen gestörten Werteliste (4.1-26) das Informationswort $\boldsymbol{u}$ zu rekonstruieren. Im Falle $t = 0$ und folglich $n = k$ ist dies als das *Interpolationsproblem* bekannt: jedes Polynom aus $F_k[x]$ kann aus seinen Werten an k verschiedenen Stellen rekonstruiert werden. Für allgemeines t haben wir ein Interpolationsproblem mit *Rauschen*: jedes Polynom aus $F_k[x]$ kann aus seinen Werten an $k+2t$ verschiedenen Stellen rekonstruiert werden, sofern dabei maximal t Fehler auftreten.

4.2 Konventionelle Reed-Solomon Codes

Konventionelle Reed-Solomon Codes, oder kurz *RS Codes*, über $F = \mathrm{GF}(q)$ sind Spezialfälle verallgemeinerter Reed-Solomon Codes, bei denen neben der Codierung $\boldsymbol{u} \mapsto \boldsymbol{u} \cdot G_{\mathrm{vRS}}$ auch die Syndrom-Berechnung $\boldsymbol{y} \mapsto \boldsymbol{y} \cdot H_{\mathrm{RS}}$ auf der Auswertung von Polynomen an geeigneten Stellen beruht.

Man wählt als Codelänge einen Teiler n von $q-1$, ein Element $\beta \in F$ der Ordnung n sowie einen weiteren Parameter $b < n$. Als Lokalisierer und Gewichte dienen die Potenzen

$$\alpha_j = \beta^j \qquad \text{bzw.} \qquad v_j = \beta^{b \cdot j} \qquad \text{für} \qquad j < n$$

Die Begriffe des primitiven, normalisierten und des RS Codes im engeren Sinne vererben sich vom verallgemeinerten Fall. Insbesondere entsprechen normalisierte RS Codes und solche im engeren Sinne der Wahl $b = 0$ bzw. $b = 1$.

In Dimension k liefert diese Spezifikation in ausmultiplizierter Form eine kanonische Kontrollmatrix mit $d-1 = n-k$ Spalten für einen RS Code $\mathcal{C}_{\mathrm{RS}}$:

$$H_{\mathrm{RS}} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ \beta^b & \beta^{b+1} & \dots & \beta^{b+d-2} \\ \left(\beta^b\right)^2 & \left(\beta^{b+1}\right)^2 & \dots & \left(\beta^{b+d-2}\right)^2 \\ \vdots & \vdots & \vdots & \vdots \\ \left(\beta^b\right)^{n-1} & \left(\beta^{b+1}\right)^{n-1} & \dots & \left(\beta^{b+d-2}\right)^{n-1} \end{pmatrix}$$

Die Polynom-Interpretation des Codes erfordert bei der Matrix in der o.a. Form wieder die Übersetzung von Tupeln $\boldsymbol{c} \in F^n$ in Polynome $c(x) \in F_n[x]$, die nach steigenden Potenzen von x geordnet sind. Dann gilt

$$\boldsymbol{c} \in \mathcal{C}_{\mathrm{RS}} \iff \boldsymbol{c} \cdot H_{\mathrm{RS}} = \boldsymbol{0} \iff \left(c\left(\beta^{b+l}\right) : l < d-1 \right) = \boldsymbol{0} \tag{4.2-27}$$

Die Syndromberechnung entspricht nunmehr der Auswertung von $y(x)$ an den Stellen β^{b+l}, $l < d-1$, die wir als *Wurzeln* von $\mathcal{C}_{\mathrm{RS}}$ bezeichnen wollen. Dann läßt sich der Zusammenhang (4.2-27) einprägsam formulieren als

Ein Polynom vom Grad $< n$ gehört genau dann zu einem RS Code der Länge n, wenn es dieselben Wurzeln hat wie der Code.

Wie zuvor ist die Charakterisierung mit Hilfe von Polynomen darstellungsunabhängig. Zieht man es aber vor, einem Vektor $\boldsymbol{c}$ das nach fallenden Potenzen von x geordnete Polynom $\bar{c}(x) = x^{n-1} c\left(x^{-1}\right)$ zuzuordnen, was sich speziell im Zusammenhang mit der Polynomdivision als nützlich erweist, so ist in der Kontrollmatrix die Reihenfolge der Zeilen umzukehren:

$$\bar{H}_{\mathrm{RS}} = \begin{pmatrix} \left(\beta^b\right)^{n-1} & \left(\beta^{b+1}\right)^{n-1} & \dots & \left(\beta^{b+d-2}\right)^{n-1} \\ \vdots & \vdots & \vdots & \vdots \\ \left(\beta^b\right)^2 & \left(\beta^{b+1}\right)^2 & \dots & \left(\beta^{b+d-2}\right)^2 \\ \beta^b & \beta^{b+1} & \dots & \beta^{b+d-2} \\ 1 & 1 & \dots & 1 \end{pmatrix}$$

Der so spezifizierte Code $\bar{\mathcal{C}}_{\mathrm{RS}}$ erfordert die Lokalisierer und Gewichte

$$\alpha_j = \beta^{n-1-j} \qquad \text{bzw.} \qquad v_j = \beta^{b\cdot(n-1-j)} \qquad j < n$$

Die Charakterisierung (4.2-27) nimmt dann folgende Form an

$$\boldsymbol{c} \in \bar{\mathcal{C}}_{\mathrm{RS}} \iff \boldsymbol{c} \cdot \bar{H}_{\mathrm{RS}} = \boldsymbol{0} \iff \left(\bar{c}\left(\beta^{b+l}\right) : l < d-1 \right) = \boldsymbol{0} \tag{4.2-28}$$

Die Wurzeln des RS Codes lassen sich auch zur vereinfachten Codierung nutzen. Wir definieren das *Generatorpolynom* $g(x)$ als das Produkt der durch die Code-Wurzeln bestimmten Monome:

$$g(x) := \prod_{l<d-1} \left(x - \beta^{b+l}\right) \tag{4.2-29}$$

Die obige Charakterisierung (4.2-27) der Codewörter eines RS Codes läßt sich mittels des Generatorpolynoms wie folgt ergänzen

$$\boldsymbol{c} \in \mathcal{C}_{\mathrm{RS}} \iff g(x) \,|\, c(x)$$

Damit läßt sich der Code $\mathcal{C}_{\mathrm{RS}}$ als Hauptideal im Vektorraum $F_n[x] \cong F^n$ charakterisieren:

$$\mathcal{C}_{\mathrm{RS}} = \{\, u(x)g(x) : u(x) \in F_k[x] \,\} \tag{4.2-30}$$

Wegen $\operatorname{grd} g(x) = d-1 = n-k$ hat das Produkt $u(x)g(x)$ maximal den Grad $n-1$, gehört also zu $F_n[x]$.

Abschließend stellen wir fest, daß wegen $\operatorname{ord}(\beta) = n$ jede Wurzel von $g(x)$ auch Wurzel von $x^n - 1$ ist und folglich $g(x)$ ein Teiler von $x^n - 1$ ist. Diese der Spezifikation eines Codes wird uns in Abschnitt 6.2 im Zusammenhang mit sogenannten zyklischen Codes in etwas allgemeinerer Form wiederbegegnen.

4.3 Codierung im Falle konventioneller Reed-Solomon Codes

Wie für jeden linearen Code kann bei vRS-Codes und folglich auch bei RS-Codes die Codierung durch Rechtsmultiplikation mit einer Generatormatrix erfolgen: $\boldsymbol{u} \mapsto \boldsymbol{u} \cdot G_{\mathrm{vRS}}$. I.A. wird solch eine Codierung aber nicht systematisch sein.

Im speziellen Fall von RS Codes kann man zudem die Darstellung (4.2-30) des Codes als von $g(x)$ erzeugtes Hauptideal von $F_n[x]$ zur Codierung verwenden: $u(x) \mapsto u(x)g(x)$. Das entsprechende Tupel $\boldsymbol{u}$ kann dann von rechts mit folgender ebenfalls nichtsystematischen *Bandmatrix* multipliziert werden

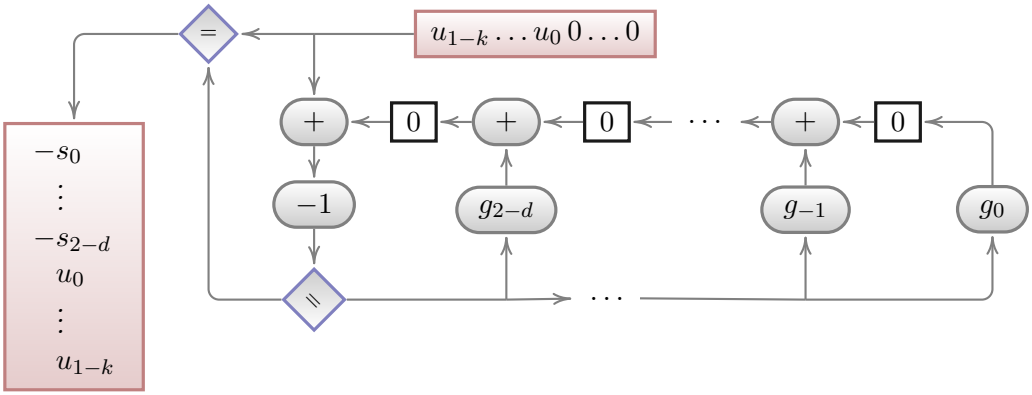
$$G = \begin{pmatrix} g_0 & g_1 & \cdots & g_{n-k} & & \\ & g_0 & g_1 & \cdots & g_{n-k} & \\ 0 & & \ddots & \ddots & \ldots & \ddots \\ & & & g_0 & g_1 & \cdots & g_{n-k} \end{pmatrix} 0$$

Im Gegensatz zur Generatormatrix G_{vRS} in Gleichung (4.1-24) ist es bei Verwendung von G irrelevant, ob die Übersetzung zwischen Vektoren und Polynomen aufsteigende oder fallende Potenzen von x verwendet, solange $\boldsymbol{u}$ und $g(x)$ gleich interpretiert werden. Daher ziehen wir eine Generatormatrix der Form G vor. Man beachte aber, daß ein Generatorpolynom $g(x)$ je nach Verwendung steigender oder fallender Potenzen von x verschiedene Bandmatrizen liefert.

Wie in Abschnitt 2.3 gezeigt, kann die Operation $u(x) \mapsto u(x)g(x)$ kann effizient mittels einer *Schieberegistermaschine* implementiert werden. Wegen $\operatorname{grd} g(x) = n-k = d-1$ verwenden wir $d-1$ Schieberegister.

Für die systematische Codierung von $\boldsymbol{u}$ verwenden wir einen Trick. Wir betrachten anstelle des Polynoms $u(x) \in F_k[x]$ das Polynom $x^{n-k}\check{u}(x) \in F(\!(x^{-1})\!)$. Division durch $g(x)$ liefert $x^{n-k}\check{u}(x) = \check{q}(x) \cdot g(x) + \check{s}(x)$ mit $\operatorname{grd} \check{s}(x) < \operatorname{grd} g(x) = d-1 = n-k$. Wegen (4.2-30) ist dann $x^{n-k}\check{u}(x) - s(x)$ ein Codewort, dessen Koeffiziententupel wir $\boldsymbol{u}$ zuordnen.

Die systematische Codierung kann durch leichte Modifikation der Maschine (2.3-19) realisiert werden: anstelle der k Koeffizienten von $q(x)$ sollen die ersten k Koeffizienten von $u(x)$ ausgegeben werden; daran sollen sich die negativen Koeffizienten des Restes anschließen. Zwecks besserer Darstellung spiegeln wir diesmal die Maschine:



(4.3-31)

Nach k Takten wechseln beide Schalter ihre Ein- bzw. Ausgabe, sodaß anstelle der letzten $n-k$ Nullen der Eingabe die negativen Inhalte der Schieberegister ausgegeben werden.

Neben dem in Abschnitt 4.1.2 angedeuteten Verfahren zur Decodierung verallgemeinerter Reed-Solomon Codes (Polynominterpolation mit Rauschen), legt das zuletzt besprochene Codierungsverfahren für konventionelle Reed-Solomon-Codes mittels Polynomdivision auch ein entsprechendes Decodierungsverfahren nahe, das sich an der Syndromdecodierung aus Abschnitt 1.3.2 orientiert. Polynome aus $F_n[x]$ auf ihren Rest bei Division durch $g(x)$ abzubilden ist offenbar linear, und der Kern dieser Operation ist $\mathcal{C}_{\mathrm{RS}}$. Also können wir den Rest bei Division durch $g(x)$ als Syndrom auffassen. Allerdings ist die Anzahl der Syndrome nur für kleine Werte von $n-k$ und folglich d handhabbar. Für eine effizientere Decodierung sind zusätzliche Eigenschaften von RS Codes heranzuziehen.

4.4 Konkatenierte Codes

Ein scheinbarer Nachteil verallgemeinerter Reed-Solomon Codes besteht in der Beschränkung ihrer Länge durch die Größe des zugrundeliegenden Körpers. Für lange vRS Codes bedarf es somit großer Körper, die einer konkreten Anwendung nicht notwendig angemessen sind. Einen Ausweg bietet die auch allgemeiner anwendbare *Konkatenation* von Codes.

Wir betrachten einen linearen $[n,k,d]_q$-Code $\mathcal{C}_{\mathrm{in}}$ über $F=\mathrm{GF}(q)$ mit Generatormatrix G_{in}, und einen linearen $[N,K,D]_{q^k}$-Code $\mathcal{C}_{\mathrm{out}}$ mit Generatormatrix G_{out} über $\Phi=\mathrm{GF}(q^k)$.

Als Vektorraum über F hat Φ die Dimension k, ist also zu F^k und folglich auch zu $\mathcal{C}_{\mathrm{in}}$ isomorph. Während die Generatormatrix G_{in} eine injektive lineare Abbildung von F^k nach F^n und folglich einen Isomorphismus von F^k nach $\mathcal{C}_{\mathrm{in}}$ liefert, benötigen wir für die folgende Anwendung noch einen konkreten Isomorphismus $\Phi \xrightarrow{\Omega} F^k$.

Im Allgemeinen wird kein kanonischer derartiger Isomorphismus existieren. Wir beschreiben nun eine Möglichkeit, trotzdem solch ein Ω konkret zu definieren. In Abschnitt 2.4 hatten wir gesehen, wie algebraische Körpererweiterungen mittels irreduzibler Polynome konstruiert werden können. Speziell Beispiel 2.4.05 zeigte verschiedene Präsentationen der Elemente einer AKE: von speziellem Interesse für uns ist hier, das für eine formale Wurzel, etwa α, eines irreduziblen Polynoms $f(x)\in F[x]$ die Potenzen α^j, $j<\operatorname{grd} f(x)$, eine Basis von $\mathrm{GF}(q^k)$ als Vektorraum über $\mathrm{GF}(q)$ bilden. Dies liefert einen Isomorphismus $(\mathrm{GF}(q))^k \xrightarrow{\Omega} \mathrm{GF}(q^k)$, der von der Wahl des irreduziblen Polynoms $f(x)$ und der Anordnung der Polynome nach fallenden oder steigenden Potenzen der formalen Wurzel abhängt.

4.4.01 Definition. Unter den oben beschriebenen Voraussetzungen besteht der $[n\cdot N, k\cdot K]$ *konkatenierte Code* $\mathcal{C}_{\mathrm{cont}}$ mit *innerem Code* $\mathcal{C}_{\mathrm{in}}$ und *äußerem Code* $\mathcal{C}_{\mathrm{out}}$ aus allen konkatenierten Wörtern in $F^{nN}=(F^n)^N$ der Form

$$\big(\,(z_i\Omega)G_{\mathrm{in}} : i<N\,\big) \quad \text{mit} \quad \boldsymbol{z}=(\,z_i : i<N\,)\in\mathcal{C}_{\mathrm{out}}$$

In Diagrammform wird die zugehörige Codierung G_{cont} definiert durch

$$\begin{array}{ccc} \left(F^k\right)^K & \xrightarrow{\quad G_{\text{conc}} \quad} & \left(F^n\right)^N \\ {\scriptstyle (\Omega^{-1})^K}\downarrow & & \uparrow{\scriptstyle (G_{\text{in}})^N} \\ \Phi^K & \xrightarrow[G_{\text{out}}]{} \Phi^N \xrightarrow[\Omega^N]{} & \left(F^k\right)^N \end{array}$$

4.4.02 Beispiel. Als inneren Code $\mathcal{C}_{\text{in}}$ wählen wir den binären $[7,3,4]$ Simplex-Code aus Beispiel 1.2.04, der zum $[7,4,3]$ Hamming-Code aus Beispiel 1.2.03 dual ist. Eine mögliche Generatormatrix ist z.B.

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

Wie in Beispiel 2.4.05 gesehen, kann $\mathrm{GF}(2^3)$ als $F[x]_{x^3+x^2+1}$ realisiert werden. Weiter sei β eine formale Wurzel von x^3+x^2+1. Als äußeren Code verwenden wir den $[6,2,5]$ vRS Code mit Generatormatrix

$$G_{\text{out}} = G_{\text{vRS}} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ \beta & \beta^2 & \beta^3 & \beta^4 & \beta^5 & \beta^6 \end{pmatrix}$$

Damit ist $\mathcal{C}_{\text{cont}}$ ein $[42,6]$-Code. Gemäß Beispiel 2.4.05(1) wird das Informationswort $(0\,1\,0\,0\,0\,1)$ von Ω^{-1} auf $(\beta^1\,\beta^0) = (\beta\,1)$ abgebildet, und dies wird codiert zu

$$(\beta\,1)\cdot G_{\text{vRS}} = \begin{pmatrix} 0 & \beta+\beta^2 & \beta+\beta^3 & \beta+\beta^4 & \beta+\beta^5 & \beta+\beta^6 \end{pmatrix}$$

Komponentenweise Anwendung von Ω und Multiplikation mit G_{in} liefert

(000 110 111 101 001 100) bzw. (0000000 0111100 1101001 1011010 1010101 0001111)

Der Minimalabstand von $\mathcal{C}_{\text{cont}}$ beträgt mindestens $d \cdot D$, denn jedes von $\boldsymbol{0} \in \Phi^N$ verschiedene Codewort von $\mathcal{C}_{\text{out}}$ enthält mindestens D von $0 \in \Phi$ verschiedene Komponenten. Diese entsprechen von $\boldsymbol{0} \in F^k$ verschiedenen Vektoren, die unter G_{in} auf Codewörter abgebildet werden, deren Hamming-Gewicht mindestens d beträgt.

Um D zu maximieren, wählt man häufig $\mathcal{C}_{\text{out}}$ als vRS Code, was im Falle $N \le q^k$ immer möglich ist. Auf diese Weise kann man bei festem q Codes beliebig großer Länge produzieren.

4.4.03 Beispiel. Der der Minimalabstand des $[42,6]_2$-Codes $\mathcal{C}_{\text{cont}}$ aus Beispiel 4.4.02 muß mindestens den Wert $4 \cdot 5 = 20$ haben. In der Tat hat das angegebene Codewort das Hamming-Gewicht 20, so daß der Minimalabstand auch nicht größer sein kann. Mit Hilfe der Griesmer-Schranke kann man zeigen, daß ein linearer Codes der Dimension $k = 6$ und des Minimalabstands $d = 21$ mindestens die Länge 44 haben muß: $\sum_{i<6} \lceil 21/2^i \rceil = 21 + 11 + 6 + 3 + 2 + 1 = 44$.

Für konkatenierte Codes gibt es eine effizienten Decodierungsalgorithmus, der Fehler mit Hamming Gewicht kleiner als $(d \cdot D)/2$ korrigiert. In theoretischer Hinsicht sind konkatenierte Codes insofern von Interesse, als man für wachsende Codelängen effektiv Folgen solcher Codes konstruieren kann, deren Coderate $(k \cdot K)/(n \cdot N)$ und deren *relativer Minimalabstand* $(d \cdot D)/(n \cdot N)$ positive Häufungspunkte haben, also nicht gegen 0 konvergieren. Darüberhinaus lassen sich mit Hilfe doppelter Konkatenation effizient Codes realisieren, die den theoretischen Grenzen von Shannons Codierungssatz für den BSC(q) nahekommen.

4.5 Alternant Codes

4.5.01 Definition. Wir betrachten den Körper $F = \mathrm{GF}(q)$ und seine algebraische Erweiterung $\Phi = \mathrm{GF}(q^m)$. Ist $\mathcal{C}_{\mathrm{vRS}}$ ein $[N, K, D]$ verallgemeinerter Reed-Solomon Code über Φ, so heißt der Durchschnitt $\mathcal{C}_{\mathrm{alt}} := \mathcal{C}_{\mathrm{vRS}} \cap F^N$ *alternant Code* über F.

Die Unterklassen primitiver und normalisierter vRS Codes sowie von vRS Codes im engeren Sinne übertragen sich auf alternant Codes.

$\mathcal{C}_{\mathrm{alt}}$ ist ein linearer $[n, k, d]$-Code über F mit $n = N$ und Minimalabstand $d \geq D$; der Wert D wird auch als *designierter Minimalabstand* bezeichnet. Allerdings ist die Dimension k des Codes a priori unbekannt.

Um k zumindest eingrenzen zu können, wenden wir uns der Frage zu, wie man aus einer $N \times (D-1)$-Kontrollmatrix H eines $[N, K, D]$ vRS Codes $\mathcal{C}_{\mathrm{vRS}}$ über Φ eine $n \times m(D-1)$-Kontrollmatrix H_{alt} des abgeleiteten alternant Codes $\mathcal{C}_{\mathrm{alt}}$ über F konstruieren kann. Da alle Komponenten von H aus $\mathrm{GF}(q^k)$ stammen, kann man sie durch die entsprechenden Ω-Urbilder aus $(\mathrm{GF}(q))^m$ (als Zeilenvektoren) ersetzen, wobei $\Phi \xrightarrow{\Omega} F^m$ ein gemäß Abschnitt 4.4 gewählter Isomorphismus ist. Daher gilt mit $h_{i,j} := \Omega(H_{i,j})$:

$$\begin{aligned} \boldsymbol{c} \cdot H = \boldsymbol{0} &\iff \forall j < D-1.\ \sum_{i<n} c_i \cdot H_{i,j} = \sum_{i<n} c_i \cdot \Omega^{-1}\boldsymbol{h}_{i,j} = \Omega^{-1}\Big(\sum_{i<n} c_i \cdot \boldsymbol{h}_{i,j}\Big) = 0 \\ &\iff \forall j < D-1.\ \sum_{i<n} c_i \cdot \boldsymbol{h}_{i,j} = 0 \\ &\iff \boldsymbol{c} \cdot H_{\mathrm{alt}} = \boldsymbol{0} \end{aligned} \tag{4.5-32}$$

Die Anzahl $m(D-1)$ der Spalten von H_{alt} liefert eine obere Schranke für die Redundanz $n-k$, oder äquivalent eine untere Schranke für die Dimension des Codes

$$n - k \leq m(D-1) \quad \text{bzw.} \quad k \geq n - m(D-1) \tag{4.5-33}$$

4.5.02 Beispiel. In Beispiel 2.2.10 hatten wir $x^4 + x + 1$ als irreduzibel über $F = \mathrm{GF}(2)$ identifiziert. Setze $\Phi = \mathrm{GF}(2^4) \cong F[x]_{x^4+x+1}$. Folgende Tabelle faßt die Struktur von Φ zusammen, wobei ξ formale Wurzel von $x^4 + x + 1$ und somit primitives Element von Φ ist. Als Referenz für später haben wir auch gleich die Minimalpolynome der Körperelemente

angegeben.

$p(\xi)$	ξ^i	ord(ξ^i)	4-Tupel	Minimalpolynom
0	$-$	$-$	0000	x
1	ξ^0	1	0001	$x+1$
ξ	ξ^1	15	0010	x^4+x+1
ξ^2	ξ^2	15	0100	x^4+x+1
ξ^3	ξ^3	5	1000	$x^4+x^3+x^2+x+1$
$\xi+1$	ξ^4	15	0011	x^4+x+1
$\xi^2+\xi$	ξ^5	3	0110	x^2+x+1
$\xi^3+\xi^2$	ξ^6	5	1100	$x^4+x^3+x^2+x+1$
$\xi^3+\xi+1$	ξ^7	15	1011	x^4+x^3+1
ξ^2+1	ξ^8	15	0101	x^4+x+1
$\xi^3+\xi$	ξ^9	5	1010	$x^4+x^3+x^2+x+1$
$\xi^2+\xi+1$	ξ^{10}	3	0111	x^2+x+1
$\xi^3+\xi^2+\xi$	ξ^{11}	15	1110	x^4+x^3+1
$\xi^3+\xi^2+\xi+1$	ξ^{12}	5	1111	$x^4+x^3+x^2+x+1$
$\xi^3+\xi^2+1$	ξ^{13}	15	1101	x^4+x^3+1
ξ^3+1	ξ^{14}	15	1001	x^4+x^3+1

Weiter sei $\mathcal{C}_{\text{vRS}}$ ein $[N=15, K=13, D=3]$-vRS-Code über Φ mit Lokalisierern $\alpha_j = \xi^j$ und Gewichten $v_j := \xi^{3j}$. Damit ist $\mathcal{C}_{\text{vRS}}$ sogar ein konventioneller Reed-Solomon Code mit $b=3$, dessen Wurzeln gegeben sind durch ξ^3 und ξ^4. Eine kanonische Kontrollmatrix hat nun die Form

$$H_{\text{vRS}} = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \xi^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \xi^6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \xi^9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \xi^{12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \xi^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \xi^6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \xi^9 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \xi^{12} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \xi^3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \xi^6 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \xi^9 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \xi^{12}
\end{pmatrix}
\begin{pmatrix}
1 & \xi^0 \\ 1 & \xi^1 \\ 1 & \xi^2 \\ 1 & \xi^3 \\ 1 & \xi^4 \\ 1 & \xi^5 \\ 1 & \xi^6 \\ 1 & \xi^7 \\ 1 & \xi^8 \\ 1 & \xi^9 \\ 1 & \xi^{10} \\ 1 & \xi^{11} \\ 1 & \xi^{12} \\ 1 & \xi^{13} \\ 1 & \xi^{14}
\end{pmatrix} =
\begin{pmatrix}
1 & 1 \\ \xi^3 & \xi^4 \\ \xi^6 & \xi^8 \\ \xi^9 & \xi^{12} \\ \xi^{12} & \xi \\ 1 & \xi^5 \\ \xi^3 & \xi^9 \\ \xi^6 & \xi^{13} \\ \xi^9 & \xi^2 \\ \xi^{12} & \xi^6 \\ 1 & \xi^{10} \\ \xi^3 & \xi^{14} \\ \xi^6 & \xi^3 \\ \xi^9 & \xi^7 \\ \xi^{12} & \xi^{11}
\end{pmatrix}$$

Nun ersetzen wir die Komponenten von H_{vRS} durch die entsprechenden Tupel aus der obigen

Tabelle, d.h., durch die Repräsentanten bzgl. der Basis $(1, \xi, \xi^2, \xi^3)$ des F-Vektorraums Φ:

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

In beiden Fällen können die Codewörter als Polynome vom Grad < 15 charakterisiert werden, die ξ^3 und ξ^4 als Wurzeln haben. Im Falle von $\mathcal{C}_{\text{RS}}$ handelt es sich um Polynome aus $\Phi[x]$, im Falle von $\mathcal{C}_{\text{alt}}$ dagegen um Polynome aus $F[x]$.

Der Minimalabstand sollte den designierten Wert 3 übersteigen. Nach Proposition 2.4.10 sind mit ξ^3 und ξ^4 ebenso $\left(\xi^3\right)^2 = \xi^6$, $\left(\xi^3\right)^4 = \xi^{12}$ und $\left(\xi^3\right)^8 = \xi^9$ sowie $\left(\xi^4\right)^2 = \xi^8$, $\left(\xi^4\right)^4 = \xi$ und $\left(\xi^4\right)^8 = \xi^2$ Wurzeln. Wie wir später sehen werden, impliziert die Existenz der Wurzeln ξ, ξ^2, ξ^3 und ξ^4 für jedes Codewort, daß $\mathcal{C}_{\text{alt}}$ Doppelfehler korrigiert, also $d \geq 5$ gilt. Dieser Wert wird tatsächlich angenommen (somit kann ξ^5 nicht Wurzel aller Codewörter sein). Als Generatorpolynom wird sich $g(x) = (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)$ herausstellen, was eine Codedimension von 7 impliziert.

Wenn man alternant Codes entwirft, sind die Größe q des Körpers, die Codelänge n und der gewünschte Minimalabstand Designparameter. Wir wollen für letzteren den designierten Minimalabstand D verwenden, der echte Minimalabstand d kann nur größer werden. Nun benötigen wir einen Erweiterungsgrad m und einen $[n, n-D-1, D]$-vRS-Code über $\text{GF}(q^m)$. Gemäß der Schranken (4.5-33) sollte m so klein wie möglich gewählt werden. Andererseits muß $n \leq q^m$ gelten, was $m \geq \lceil \log_q n \rceil$ erfordert.

Die rechte Seite der Schranken (4.5-33) läßt sich in bestimmten interessanten Fällen weiter verbessern, wie wir anhand einiger Beispiele zeigen.

4.5.03 Beispiel. Im binären Fall, d.h., $F = \text{GF}(2)$, nehmen wir an, daß der designierte Minimalabstand D ungerade ist. Wir wollen einen binären $[n, k, d \geq D]$ alternant Code $\mathcal{C}_{\text{alt}}$ im engeren Sinne unter Verwendung eines $[n, n-D+1, D]$-vRS Codes im engeren Sinne $\mathcal{C}_{\text{vRS}}$ über $\Phi = \text{GF}(2^m)$ konstruieren.

Da es sich um einen vRS-Code im engeren Sinne handelt, d.h., $\boldsymbol{v} = \alpha$, hat die ausmultiplizierte

kanonische Kontrollmatrix die besonders einfache Form

$$H_{\text{vRS}} = \begin{pmatrix} \alpha_0 & \alpha_0^2 & \dots & \alpha_0^{D-1} \\ \alpha_1 & \alpha_1^2 & \dots & \alpha_1^{D-1} \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_{n-1} & \alpha_{n-1}^2 & \dots & \alpha_{n-1}^{D-1} \end{pmatrix}$$

In Anlehnung an (4.5-32) erhalten wir nun

$$\boldsymbol{c} \cdot H = \boldsymbol{0} \iff \forall j < D-1.\ \sum_{i<n} c_i \cdot \alpha_j^i = 0$$

Nun wenden wir für $q = 2$ Proposition 2.4.10 an und berücksichtigen, daß in GF(2) jedes Element mit seinem Quadrat übereinstimmt. Für jedes $j < D-1$ gilt somit

$$\sum_{i<n} c_i \cdot \alpha_i^j = 0 \iff \sum_{i<n} c_i^2 \cdot \alpha_i^{2j} = 0 \iff \sum_{i<n} c_i \cdot \alpha_i^{2j} = 0$$

Damit sind die geraden Spalten der Kontrollmatrix H_{vRS} redundant, die $(D-1)/2$ ungeraden Spalten reichen als Tests für die Zugehörigkeit zu $\mathcal{C}_{\text{vRS}}$ aus:

$$H'_{\text{vRS}} = \begin{pmatrix} \alpha_0 & \alpha_0^3 & \alpha_0^5 & \dots & \alpha_0^{D-2} \\ \alpha_1 & \alpha_1^3 & \alpha_1^5 & \dots & \alpha_1^{D-2} \\ \vdots & \vdots & & \vdots\!\vdots & \vdots \\ \alpha_{n-1} & \alpha_{n-1}^3 & \alpha_{n-1}^5 & \dots & \alpha_{n-1}^{D-2} \end{pmatrix}$$

Die Schranke (4.5-33) verbessert sich nun zu

$$n - k \le m(D-1)/2 \quad \text{bzw.} \quad k \ge n - m(D-1)/2$$

4.5.04 Beispiel. Ist der designierte Minimalabstand im binären Fall gerade, funktioniert eine ähnliche Konstruktion wie oben, wenn wir mit einem normalisierten $[n, n-D+1, D]$-vRS-Code über GF(q^m) starten. In diesem Fall erhalten wir eine reduzierte Kontrollmatrix der Form

$$H'_{\text{vRS}} = \begin{pmatrix} 1 & \alpha_0 & \alpha_0^3 & \alpha_0^5 & \dots & \alpha_0^{D-3} \\ 1 & \alpha_1 & \alpha_1^3 & \alpha_1^5 & \dots & \alpha_1^{D-3} \\ \vdots & \vdots & \vdots & & \vdots\!\vdots & \vdots \\ 1 & \alpha_{n-1} & \alpha_{n-1}^3 & \alpha_{n-1}^5 & \dots & \alpha_{n-1}^{D-3} \end{pmatrix}$$

Von den ersten m Spalten in H_{alt} können wir $m-1$ Nullspalten streichen, was die folgende Verbesserung der Schranken (4.5-33) liefert:

$$n - k \le 1 + (D/2 - 1) \cdot m \qquad \text{bzw.} \qquad k \ge n - 1 + (D/2 - 1) \cdot m$$

4.5.05 Beispiel. Wir konstruieren einen erweiterten primitiven (d.h., alle Körperelemente treten als Lokalisierer auf) alternant Code $\mathcal{C}_{\text{alt}}$ der Länge $n = 64$ mit designiertem Minimalabstand $D = 12$ über $F = \text{GF}(2)$. Mit $m = 6$ (und folglich $n = 2^m$) wollen wir einen normalisierten erweiterten primitiven vRS-Code verwenden. Gemäß der verbesserten Schranke in Beispiel 4.5.04 ist die Redundanz von $\mathcal{C}_{\text{alt}}$ durch $1 + (D/2 - 1)m = 31$ nach oben beschränkt, d.h., $k \geq 33$.

Aber für diesen speziellen Code lassen sich noch bessere Schranken finden. Sind die Elemente von $\Phi = \text{GF}(2^6)$ durchnummeriert als α_j, $j < 64$, so erhält man eine Kontrollmatrix H_{alt} für $\mathcal{C}_{\text{alt}}$, indem man die Komponenten der Matrix

$$H = \begin{pmatrix} 1 & \alpha_0 & \alpha_0^3 & \alpha_0^5 & \alpha_0^7 & \alpha_0^9 \\ 1 & \alpha_1 & \alpha_1^3 & \alpha_1^5 & \alpha_1^7 & \alpha_1^9 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha_{63} & \alpha_{63}^3 & \alpha_{63}^5 & \alpha_{63}^7 & \alpha_{63}^9 \end{pmatrix}$$

mit den entsprechenden Zeilenvektoren aus F^6 ersetzt. Wie in Beispiel 4.5.04 können 5 der ersten 6 Spalten eliminiert werden. Betrachten wir nun die letzte Spalte von H. Jede von 0 verschiedenen Komponente hat die Ordnung 1 oder 7. Folglich handelt es sich bei allen Elementen der letzten Spalte um Wurzeln des Polynoms $Q(x) = x^8 - x$. Da die Abbildung $\Phi \longrightarrow \Phi$ mit $\beta \mapsto Q(\beta)$ über F linear ist, bilden diese Wurzeln einen Vektorraum über F, dessen Dimension höchstens 3 ist. Tatsächlich bestimmen diese Wurzeln sogar $\text{GF}(2^3)$ als Teilkörper von $\Phi = \text{GF}(2^6)$. Damit spannen die letzten 6 Spalten von H_{alt} nur einen Unterraum der Dimension 3 auf, was $n - k \leq 28$ bzw. $k \geq 36$ impliziert.

4.6 BCH-Codes

Bose-Chaudhuri-Hocquenghem, oder kurz *BCH-Codes* sind alternant Codes mit zugrundeliegendem konventionellen Reed-Solomon Code. Falls $F = \text{GF}(q)$ und falls $\mathcal{C}_{\text{RS}}$ ein $[N, K, D]$ RS-Code über $\Phi = \text{GF}(q^m)$ ist, dann ist $\mathcal{C}_{\text{RS}} \cap F^N$ ein BCH-Code, den wir mit $\mathcal{C}_{\text{BCH}}$ bezeichnen wollen.

Zur Erinnerung: gemäß Abschnitt 4.2 muß N den Wert $q^m - 1$ teilen, woraus unmittelbar $\text{ggT}(N, q) = 1$ folgt. Aber dies impliziert, daß $q \bmod N$ in $\mathbb{Z}_N$ invertierbar ist. Ist somit die Codelänge $n = N$ vorgegeben, ergibt sich die Ordnung von q in $\mathbb{Z}_N^*$ als kleinstmöglicher Wert für m.

Das liefert folgende Charakterisierung von BCH-Codes: für $F = \text{GF}(q)$ und $n > 0$ relativ prim zu q wähle $m > 0$ als (kleinste) Zahl mit $n|(q^m - 1)$. Weiter habe $\beta \in \Phi = \text{GF}(q^m)$ die Ordnung n. Mit einem designierten Minimalabstand $D \leq n$ und dem Parameter $b < n$ besteht der BCH-Code $\mathcal{C}_{\text{BCH}}$ aus denjenigen Polynomen $c(x) \in F_n[x]$, die

$$c\left(\beta^{b+l}\right) = 0 \qquad \text{für } l < D - 1$$

erfüllen. Die Folge der Wurzeln des zugrundeliegenden RS-Codes

$$\langle\, \beta^{b+l} : l < D - 1 \rangle$$

wird als *fortlaufende Wurzelfolge* des Codes $\mathcal{C}_{\text{BCH}}$ bezeichnet.

4.6.01 Beispiel. Wir konstruieren einen binären BCH-Code der Länge $n = 85$, der drei Fehler korrigieren kann. m sei die kleinste Zahl, so daß $2^m - 1$ durch 85 teilbar ist, was den Wert $m = 8$ liefert. Aufgrund des designierten Minimalabstands $D = 7$ wählen wir $b = 1$, was einen RS-Code im engeren Sinne liefert und es gemäß Beispiel 4.5.03 erlaubt, die Schranke für k zu verbessern; die Redundanz beträgt höchstens $m(D-1)/2 = 24$. Also wird der resultierende BCH-Code ein linearer binärer $[85, \geq 61, \geq 7]$-Code sein. Hat α in $\mathrm{GF}(2^8)$ die Ordnung 85, so erhält man eine 85×24-Kontrollmatrix dieses Codes, indem man die Komponenten der Matrix

$$\begin{pmatrix} 1 & 1 & 1 \\ \beta & \beta^3 & \beta^5 \\ \beta^2 & \beta^6 & \beta^{10} \\ \vdots & \vdots & \vdots \\ \beta^j & \beta^{3j} & \beta^{5j} \\ \vdots & \vdots & \vdots \\ \beta^{83} & \beta^{79} & \beta^{75} \\ \beta^{84} & \beta^{82} & \beta^{80} \end{pmatrix}$$

als Zeilenvektoren in $\mathrm{GF}(2^8)$ repräsentiert.

5 Praktische Decodierung von Reed-Solomon Codes

Ziel der Decodierung verallgemeinerter Reed-Solomon Codes ist es zunächst, alle Fehlermuster mit einem Hamming-Gewicht unterhalb der Hälfte des Minimalabstands des Codes in polynomialer Zeit zu korrigieren. (Wegen der Einschränkung der zu korrigierenden Fehlermuster handelt es sich *nicht* um eine Maximum-Likelihood-Decodierung.) Zu diesem Zweck wird ausgehend von einem Syndrompolynom eine dreiteilige "Schlüsselgleichung" (in bestimmten Polynomen) entwickelt, mit deren Lösung das obige Ziel erreicht werden kann. Die Nullstellen des einen Lösungspolynoms bestimmen die Positionen der korrigierbaren Fehler, deren Werde mit Hilfe des zweiten Polynoms bestimmt werden können.

Es werden drei Verfahren zur Lösung der Schlüsselgleichung vorgestellt. Beim ersten handelt es sich um die direkten Lösung linearer Gleichungssysteme, während das zweite auf einer Abwandlung des Euklidischen Algorithmus für Polynome beruht. Das dritte stammt von Berlekamp-Massey und kann als Lösung einer linearen Rekurrenzgleichung aufgefaßt werden, wobei die zugehörige Folge durch die Koeffizienten des Syndrom-Polynoms gegeben sind.

5.1 Einführung

Wir betrachten einen $[n, k, d]$ verallgemeinerten Reed-Solomon Code $\mathcal{C}_{\mathrm{vRS}}$ mit Kontrollmatrix H_{vRS} gemäß Formel (4.1-23). Dabei besteht der Lokalisierer $\alpha \in F^n$ aus paarweise verschiedenen Komponenten, während die Gewichte im Vektor $\boldsymbol{v}$ alle von $\boldsymbol{0}$ verschieden sind. Nach der

Übertragung durch einen additiven Kanal $\langle F, F, P\rangle$ wollen wir bis zu $(d-1)/2$ Fehler korrigieren können.

Wie üblich bezeichne $\boldsymbol{c} \in F^n$ das gesendete Codewort, $\boldsymbol{y}$ das empfangene Wort und $\boldsymbol{e} := \boldsymbol{y} - \boldsymbol{c}$ das Fehlerwort mit Hamming-Gewicht $\omega(\boldsymbol{e}) \le (d-1)/2$. Schließlich sei $J \subseteq n = \{0, 1, \ldots, n-1\}$ die Menge der Fehlerpositionen, d.h., die Menge der Indizes $\kappa < n$ mit $e_\kappa \neq 0$.

5.2 Syndrom-Berechnung

Das Syndrom des Worts $\boldsymbol{y}$ bzgl. H_{vRS} ist gegeben durch

$$S := y \cdot H_{\text{vRS}} \in F^{d-1} \qquad \text{mit den Komponenten} \qquad S_\ell := \sum_{j<n} y_j v_j \alpha_j^l \quad \text{für } \ell < d-1$$

5.2.01 Beispiel. Im Fall konventioneller Reed-Solomon Codes sind die Lokalisierer durch ein Element $\beta \in F = \mathrm{GF}(q)$ der Ordnung n bestimmt. Bei den Gewichten kommt noch ein Parameter $b < n$ zum Tragen:

$$\alpha_j = \beta^j \qquad \text{und} \qquad v_j = \beta^{bj}$$

Die Syndromkomponenten

$$S_\ell = \sum_{j<n} y_j \beta^{j(b+\ell)}$$

ergeben sich nun auch durch Auswertung des Polynoms $y(x)$ (mit steigenden Exponenten) an den durch die Lokalisierer bestimmten Positionen $\beta^{b+\ell}$, $\ell < d-1$.

Die Schlüsselgleichung wird drei Polynome in Relation setzen, deren erstes das *Syndrompolynom* ist:

$$S(x) := \sum_{\ell<d-1} S_\ell x^\ell$$

Da es sich bei $\mathcal{C}_{\text{vRS}}$ um einen linearen Code handelt, haben das empfangene Wort $\boldsymbol{y}$ und das Fehlerwort $\boldsymbol{e}$ dasselbe Syndrom. Darüberhinaus weist $\boldsymbol{e}$ nur für Indizes in J von 0 verschiedene Komponenten auf. Daher gilt

$$S(x) = \sum_{\ell<d-1} \Big(\sum_{j\in J} e_j v_j \alpha_j^l\Big) x^\ell = \sum_{j\in J} e_j v_j \sum_{\ell<d-1} (\alpha_j x)^\ell \tag{5.2-34}$$

Die Überlegung, keine Potenzen von x jenseits x^{d-2} betrachten zu müssen, veranlaßt uns, statt in $F[x]$ im Faktorring $F[x]/x^{d-1}$ zu rechnen. Aber dort ist jedes Polynom $\sum_{\ell<d-1} (\alpha_j x)^\ell$, $j \in J$, aus der obigen Formel invertierbar

$$(1-\alpha_j x) \sum_{\ell<d-1} (\alpha_j x)^\ell \bmod x^{d-1} = 1 - (\alpha_j x)^{d-1} \bmod x^{d-1} = 1 \tag{5.2-35}$$

Dies liefert folgende Darstellung des Syndrompolynoms in $F[x]/x^{d-1}$:

$$S(x) = \sum_{j\in J} \frac{e_j v_j}{1-\alpha_j x} \bmod x^{d-1} \tag{5.2-36}$$

5.3 Die Schlüsselgleichung der vRS-Decodierung

Zwei weitere Polynome dienen der Decodierung verallgemeinerter Reed-Solomon-Codes, sind aber zunächst unbekannt. Das *Fehlerstellenpolynom*, oder kurz *ELP* $\Lambda(x)$, sowie das *Fehlerwertepolynom*, oder kurz *EVP* $\Gamma(x)$, sind definiert durch

$$\Lambda(x) := \prod_{j\in J}(1-\alpha_j x) \qquad \text{bzw.} \qquad \Gamma(x) = \sum_{j\in J} e_j v_j \prod_{m\in J\setminus\{j\}} (1-\alpha_m x) = \Lambda(x) \sum_{j\in J} \frac{e_j v_j}{1-\alpha_j x}$$

Da die Lokalisierer unseres vRS Codes paarweise verschieden sind, gilt offenbar

$$\Lambda(\alpha_\kappa) = 0 \qquad \text{gdw} \qquad \kappa \in J$$

Damit kann $\Lambda(x)$ zum Auffinden der verfälschten Positionen in $\boldsymbol{y}$ verwendet werden, und zwar mittels seiner Nullstellen. Deren Inverse bestimmen diejenigen Lokalisierer, für die in der entsprechenden Komponente des Worts $\boldsymbol{y}$ ein Fehler aufgetreten ist. Weiterhin gilt für jedes $\kappa \in J$ natürlich

$$\Gamma\left(\alpha_\kappa^{-1}\right) = e_\kappa v_\kappa \prod_{m\in J\setminus\{\kappa\}} (1-\alpha_m \alpha_\kappa^{-1}) \neq 0$$

Folglich läßt sich e_κ bei Kenntnis von $\Gamma\left(\alpha_\kappa^{-1}\right)$ berechnen. Weiterhin haben $\Lambda(x)$ und $\Gamma(x)$ keine gemeinsamen Nullstellen, was auch durch die Formel

$$\boxed{\operatorname{ggT}\Big(\Lambda(x), \Gamma(x)\Big) = 1} \tag{5.3-37}$$

zum Ausdruck gebracht werden kann (es genügt, sich auf *normierte* gemeinsame Teiler zu beschränken). Speziell folgt aus $\Gamma(x) = 0$ (leere Summe) sofort $\Lambda(x) = 1$ (leeres Produkt), was einer fehlerfreien Übertragung entspricht.

Definitionsgemäß hat $\Lambda(x)$ den Grad $|J|$, während $\Gamma(x)$ aus Summanden besteht, die maximal den Grad $|J|-1$ haben können. Nach Voraussetzung gilt somit

$$\boxed{\operatorname{grd}\Gamma(x) < \operatorname{grd}\Lambda(x) \leq \frac{d-1}{2}} \tag{5.3-38}$$

Schließlich können wir aufgrund von Gleichung (5.2-36) die Polynome $\Lambda(c)$ und $\Gamma(x)$ mittels $S(x)$ in Beziehung setzen:

$$\boxed{\Lambda(x)S(x) \bmod x^{d-1} = \Gamma(x)} \tag{5.3-39}$$

Die (Un)gleichungen (5.3-37–5.3-39) machen zusammen die sogenannte "Schlüsselgleichung" der vRS-Decodierung aus. Während $S(x)$ aus dem empfangenen Wort $\boldsymbol{y}$ berechnet werden kann, ist die Schlüsselgleichung nach $\Lambda(x)$ und $\Gamma(x)$ zu lösen.

5.3.01 Bemerkung. Gleichung (5.3-39) schließt bestimmte Syndrome aus, sofern das empfangene Wort $\boldsymbol{y}$ maximal $(d-1)/2$ Fehler enthält. Beispielsweise kann in diesem Fall das Syndrom $\boldsymbol{S} = \boldsymbol{y} \cdot H_{\text{vRS}}$ nicht den Wert $(1\,0\,0\,\ldots\,0)$ und folglich $S(x)$ nicht den Wert 1 annehmen. Andererseits ist die H_{vRS} zugrundeliegende lineare Abbildung $F^n \longrightarrow F^{d-1}$ surjektiv. Also können die Hamming-Kugeln mit Radius $(d-1)/2$ den Raum F^n *nicht* ausschöpfen.

5.3.1 Lösbarkeit der Schlüsselgleichung

Zur Abkürzung setzen wir $\tau := (d-1)/2$. Dies ist eine obere Schranke für $\operatorname{grd}\Lambda(x)$.

In Analogie zu Abschnitt 4.3 läßt sich das Produkt $\Lambda(x)S(x)$ aufsteigend als Vektor realisieren, indem man den Koeffizientenvektor Λ mit einer Bandmatrix multipliziert, in der das Syndrom pro Zeile um eine Position nach rechts verschoben ist:

$$\Lambda(x)S(x) \in F[x] \qquad \text{entspricht} \qquad \Lambda \cdot \begin{pmatrix} S_0 & S_1 & \ldots & S_{d-2} & & \\ & S_0 & S_1 & \ldots & S_{d-2} & \\ 0 & & \ddots & \ddots & \ldots & \ddots \\ & & & S_0 & S_1 & \ldots & S_{d-2} \end{pmatrix}$$

Rechnet man aber modulo x^{d-1}, so sind nur die ersten $d-1$ Spalten der Bandmatrix relevant: die restlichen Spalten liefern Koeffizienten von x-Potenzen, deren Exponent mindestens den Wert $d-1$ hat. Gemäß Gleichung (5.3-39) erhalten wir dann ein Ergebnis $\Gamma \in F^{d-1}$, dessen zugehöriges Polynom $\Gamma(x)$ aber wegen (5.3-38) maximal den Grad $\tau-1$ haben kann; insbesondere nehmen die letzten $d-1-\tau = \lceil (d-1)/2 \rceil$ Komponenten den Wert 0 an:

$$(\Lambda_0\,\Lambda_1\,\ldots\,\Lambda_\tau) \cdot \begin{pmatrix} S_0 & S_1 & \ldots & S_{\tau-1} & S_\tau & \ldots & S_{d-2} \\ & S_0 & \ldots & S_{\tau-2} & S_{\tau-1} & \ldots & S_{d-3} \\ & & \ddots & \vdots & \vdots & \ddots & \vdots \\ & 0 & & S_0 & S_1 & \ldots & S_{d-\tau-1} \\ & & & & S_0 & \ldots & S_{d-\tau-2} \end{pmatrix} = (\Gamma_0\,\Gamma_1\,\ldots\,\Gamma_{\tau-1}\,0\,\ldots\,0)$$

Nach Definition gilt $\Lambda_0 = 1$, was aufgrund der ersten Gleichung (= Spalte der Matrix) $\Gamma_0 = S_0$ impliziert. Die verbleibenden τ Komponenten von Λ erfüllen nach Voraussetzung die letzten $d-1-\tau \geq \tau$ Gleichungen, können also per Elimination bestimmt werden (bei geradem d ist eine Gleichung redundant). Damit verbleiben $\tau-1$ Gleichungen zur Bestimmung der letzten $\tau-1$ Komponenten von Γ.

Wir betrachten nun $\langle \Lambda, \Gamma \rangle$ als eine Lösung des Gleichungssystems

$$\lambda \cdot \begin{pmatrix} S_0 & S_1 & \ldots & S_{\tau-1} & S_\tau & \ldots & S_{d-2} \\ & S_0 & \ldots & S_{\tau-2} & S_{\tau-1} & \ldots & S_{d-3} \\ & & \ddots & \vdots & \vdots & \ddots & \vdots \\ & 0 & & S_0 & S_1 & \ldots & S_{d-\tau-1} \\ & & & & S_0 & \ldots & S_{d-\tau-2} \end{pmatrix} = (\gamma \mid \boldsymbol{0}) \tag{5.3-40}$$

wobei λ und γ Variablenvektoren der Länge $\tau+1$ bzw. τ sind.

Beim folgenden wichtigen Ergebnis ist zu beachten, daß es zwar auch aus der Gradbedingung (5.3-38) folgt, in jener Form aber zu schwach ist um in Abschnitt 5.4 angewendet werden zu können, vergleiche Bemerkung 5.4.03. Dort kann zunächst nur die hier verwendete schwächere Voraussetzung realisiert werden.

5.3.02 Proposition. *Das Paar* $\langle\bar\Lambda(x),\bar\Gamma(x)\rangle\in F[x]^2$ *genüge der abgeschwächten Gradbedingung*

$$\boxed{\operatorname{grd}\bar\Lambda(x)\le\frac{d-1}{2}>\operatorname{grd}\bar\Gamma(x)} \tag{5.3-41}$$

(0) *Ist* $\langle\bar\Lambda(x),\bar\Gamma(x)\rangle$ *auch Lösung von (5.3-40), so existiert* $\kappa(x)\in F[x]$ *mit*

$$\bar\Lambda(x)=\kappa(x)\cdot\Lambda(x)\quad\text{und}\quad\bar\Gamma(x)=\kappa(x)\cdot\Gamma(x)$$

Insbesondere ist $\langle\Lambda(x),\Gamma(x)\rangle$ *eine Lösung, deren erste Komponente minimalen Grad hat.*

(1) *Ist* $\langle\bar\Lambda(x),\bar\Gamma(x)\rangle$ *eine Lösung von (5.3-40), deren erste Komponente minimalen Grad hat, sind beide Komponenten teilerfremd, d.h.,* $\operatorname{ggT}\Big(\bar\Lambda(x),\bar\Gamma(x)\Big)=1$.

Beweis:

(0) Als Produkt invertierbarer Elemente (vergl. (5.2-35)) von $F[x]_{x^{d-1}}$ ist $\Lambda(x)$ ebenfalls invertierbar. Damit läßt sich das Syndrompolynom wie folgt ausdrücken:

$$\Gamma(x)\big(\Lambda(x)\big)^{-1}\bmod x^{d-1}=S(x)$$

Somit ist $\langle\Lambda'(x),\Gamma'(x)\rangle$ genau dann Lösung von (5.3-40), wenn

$$\bar\Lambda(x)\cdot\Gamma(x)\bmod x^{d-1}=\Lambda(x)\cdot\bar\Gamma(x)\bmod x^{d-1}$$

gilt. Wegen der Gradbedingungen (5.3-41) und (5.3-38) für $\langle\bar\Lambda(x),\bar\Gamma(x)\rangle$ bzw. $\langle\Lambda(x),\Gamma(x)\rangle$ reduziert sich die letzte Bedingung zu einer Gleichung $\bar\Lambda(x)\cdot\Gamma(x)=\Lambda(x)\cdot\bar\Gamma(x)$ in $F[x]$. Da $\Lambda(x)$ und $\Gamma(x)$ teilerfremd sind, muß $\bar\Lambda(x)$ durch $\Lambda(x)$ teilbar sein, etwa $\bar\Lambda(x)=\kappa(x)\cdot\Lambda(x)$. Kürzen von $\Lambda(x)$ liefert dann auch $\kappa(x)\cdot\Gamma(x)=\bar\Gamma(x)$.

Da jede Lösung der Gleichung 5.3-40 die Fom $\langle\kappa(x)\Lambda(x),\kappa(x)\Gamma(x)\rangle$ hat, ist $\langle\Lambda(x),\Gamma(x)\rangle$ eine Lösung, deren erste Komponente den kleinstmöglichen Grad hat.

(1) In diesem Fall unterscheiden sich $\Lambda(x)$ und $\bar\Lambda(x)$ nur um einen skalaren Faktor. □

Damit haben wir die Lösung der Schlüsselgleichung auf die Lösung zweier durch (5.3-40) bestimmter linearer Gleichungssysteme reduziert, wobei wir an einer Teillösung $\Lambda'(x) \neq 0$ minimalen Grades interessiert sind. Bis auf einen skalaren Faktor stimmt diese mit dem ELP $\Lambda(x)$ überein.

Die Lösung von (5.3-40) mittels Gauss'scher Elimination ist als *Peterson-Gorenstein-Zierler Algorithmus* bekannt und hat eine Zeitkomplexität von $O(d^3)$. Für Systeme dieser speziellen Form existieren aber Verfahren, basierend auf dem Euklidischen Algorithmus, die quadratisch in d sind (vergl. Abschnitt 5.4). Außerdem lassen sich die Fehlerwerte mit Hilfe von $\Gamma(x)$ ebenfalls in quadratischer Zeit in d bestimmen.

5.3.03 Beispiel. Über der Erweiterung $\Phi = \mathrm{GF}(2^4) \cong F[x]_{x^4+x+1}$ (vergl. Beispiel 4.5.02) des Körpers $F = \mathrm{GF}(2)$ betrachten wir einen $[15, 11]$ normalisierten primitiven vRS-Code $\mathcal{C}$ mit canonischer Kontrollmatrix H. Welches Codewort $\boldsymbol{c}$ liefert nach Übertragung durch einen additiven Kanal (Φ, Φ, P)

$$\boldsymbol{y} = (\,\xi \quad \xi \quad \xi^2 \;\; \xi^2 \;\; \xi^4 \;\; \xi^5 \;\; \xi^6 \;\; \xi^7 \;\; \xi^8 \;\; \xi^9 \;\; \xi^{10}\, \xi^{11}\, \xi^{12}\, \xi^{13}\, \xi^{14}\,)\;?$$

In einem normalisierten vRS Code haben alle Gewichte den Wert 1. Da $\mathcal{C}$ zudem primitiv ist, kommen alle von 0 verschiedenen Körperelemente als Lokalisierer vor. Deren natürliche Reihenfolge ist durch aufsteigende Potenzen von ξ gegeben: $\alpha = (\,\xi^i{:}i < 15\,)$. Nach Voraussetzung gilt $d - 1 = n - k = 4$. Damit ergibt sich die Kontrollmatrix canonisch als $H = (\,\xi^{i\cdot j} : i < 15,\, j < 4\,)$.

Zunächst wird nun das Syndrom für $\boldsymbol{y}$ bestimmt:

$$\boldsymbol{S} = \boldsymbol{y} \cdot H = \boldsymbol{y} \cdot \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & \xi & \xi^2 & \xi^3 \\ 1 & \xi^2 & \xi^4 & \xi^6 \\ 1 & \xi^3 & \xi^6 & \xi^9 \\ 1 & \xi^4 & \xi^8 & \xi^{12} \\ 1 & \xi^5 & \xi^{10} & 1 \\ 1 & \xi^6 & \xi^{12} & \xi^3 \\ 1 & \xi^7 & \xi^{14} & \xi^6 \\ 1 & \xi^8 & \xi & \xi^9 \\ 1 & \xi^9 & \xi^3 & \xi^{12} \\ 1 & \xi^{10} & \xi^5 & 1 \\ 1 & \xi^{11} & \xi^7 & \xi^3 \\ 1 & \xi^{12} & \xi^9 & \xi^6 \\ 1 & \xi^{13} & \xi^{11} & \xi^9 \\ 1 & \xi^{14} & \xi^{13} & \xi^{12} \end{pmatrix} = (\xi^{12}\, \xi^{14}\, \xi^6 \quad \xi \quad)$$

Wegen $\tau = (d-1)/2 = 2$ ist dann folgendes Gleichungssystem zu lösen:

$$(\,\Lambda_0\, \Lambda_1\, \Lambda_2\,) \cdot \begin{pmatrix} \xi^{12} & \xi^{14} & \xi^6 & \xi \\ 0 & \xi^{12} & \xi^{14} & \xi^6 \\ 0 & 0 & \xi^{12} & \xi^{14} \end{pmatrix} = (\,\Gamma_0\, \Gamma_1\, 0\; 0\,)$$

Wir wissen bereits $\Lambda_0 = 1$ und somit $\Gamma_0 = \xi^{12}$. Aus den hinteren beiden Gleichungen

$$(\,1\;\;\Lambda_1\,\Lambda_2\,) \cdot \begin{pmatrix} \xi^6 & \xi \\ \xi^{14} & \xi^6 \\ \xi^{12} & \xi^{14} \end{pmatrix} = (\,0\,0\,) \qquad \text{bzw.} \qquad \begin{matrix} \xi^{14}\Lambda_1 + \xi^{12}\Lambda_2 = \xi^6 \\ \xi^6\Lambda_1 + \xi^{14}\Lambda_2 = \xi \end{matrix}$$

können wir z.B. Λ_2 eliminieren, was $(\xi + \xi^6)\Lambda_1 = \xi^{11}\Lambda_1 = \xi^8 + \xi = \xi^{10}$ liefert. Aus $\Lambda_1 = \xi^{14}$ folgt nach Rücksubstitution $\Lambda_2 = \xi^3$.

Als Nullstellen des ELP

$$\boxed{\Lambda(x) = 1 + \xi^{14}x + \xi^3 x^2}$$

kommen nur Kehrwerte der Lokalisierer α_i, $i < n$, in Frage. Systematisches Probieren oder geschicktes Raten liefert

$$x_0 = 1 \quad \text{und} \quad x_1 = \xi^{12}$$

Ihre Kehrwerte sind die Lokalisierer $1 = \xi^0 = \alpha_0$ bzw. $\xi^(-12) = \xi^3 = \alpha_3$. Also sind Fehler in den Positionen 0 und 3 aufgetreten.

Um das EVP vollständig zu bestimmen, betrachten wir die Gleichung:

$$(\,1 \quad \xi^{14}\,\xi^3\,) \cdot \begin{pmatrix} \xi^{14} \\ \xi^{12} \\ \xi 6 \end{pmatrix} = \Gamma_1$$

woraus wir $\Gamma_1 = \xi^{14}\left(1 + \xi^{12}\right) = \xi^{14}\xi^{11} = \xi^{10}$ folgern. Nun ist das EVP

$$\boxed{\Gamma(x) = \xi^{12} + \xi^{10}x}$$

an den oben bestimmten Nullstellen des ELP auszuwerten:

$$\Gamma(1) = \xi^{12} + \xi^{10} = \xi^3 \qquad \text{sowie} \qquad \Gamma\left(\xi^{12}\right) = \left(1 + \xi^{10}\right)\xi^{12} = \xi^2$$

Dies liefert folgende Fehlerwerte:

$$e_0 = \frac{\Gamma(1)}{1 - \alpha_3\alpha_0^{-1}} = \frac{\xi^3}{1 - \xi^3} = \xi^4 \qquad \text{sowie} \qquad e_3 = \frac{\Gamma\left(\xi^{12}\right)}{1 - \alpha_0\alpha_3^{-1}} = \frac{\xi^2}{1 - \xi^{12}} = \xi^6$$

Das korrigierte Wort $\boldsymbol{c} = \boldsymbol{y} - \boldsymbol{e}$ hat somit die Komponenten

$$c_0 = y_0 - e_0 = \xi + \xi^4 = 1 \qquad \text{bzw.} \qquad c_3 = y_3 - e_3 = \xi^2 + \xi^6 = \xi^3$$

5.4 Schnelle Lösung der Schlüsselgleichung mit Euklidischem Algorithmus

Nachdem wir uns von der Lösbarkeit der Schlüsselgleichung überzeugt haben, geht es nun darum, die Effizienz im Vergleich zu $O(d^3)$ bei der Standardelimination zu verbessern. Zu diesem Zweck betrachten wir eine Modifikation des Euklidischen Algorithmus.

Zur Erinnerung: in $\mathbb{Z}$ finden wir den größten gemeinsamen Teiler von g_0 und g_1 mit $|g_0| \geq |g_1| > 0$ mit Hilfe der Rekursion

$$\begin{aligned} g_0 &= b_1 g_1 + g_2 && \text{mit} \quad |g_2| < |g_1| \\ g_k &= b_{k+1} g_{k+1} + g_{k+2} && \text{mit} \quad |g_{k+2}| < |g_{k+1}| \end{aligned}$$

die nach endlich vielen Schritten abbricht, nämlich sobald erstmals $g_{k+2} = 0$ gilt.

5.4.01 Lemma. *Aus* $g_{K+2} = 0$ *und* $g_k \neq 0$ *für* $k < K+2$ *folgt* $g_{K+1} = \mathrm{ggT}(g_0, g_1)$.

Beweis: Wir definieren rekursiv zwei Zahlenfolgen $(m_i)_{i<K+3}$ und $(\ell_i)_{i<K+3}$ mittels

$$\begin{aligned} m_0 &= 1 & & \ell_0 = 0 \\ m_1 &= 0 & \text{bzw.} \qquad & \ell_1 = 1 \\ m_{k+2} &= m_k - b_{k+1} m_{k+1} & & \ell_{k+2} = \ell_k - b_{k+1} \ell_{k+1} \end{aligned}$$

Ausgehend von $g_0 = m_0 g_0 + \ell_0 g_1$ sowie $g_1 = m_1 g_0 + \ell_1 g_1$ liefert vollständige Induktion

$$g_{k+2} = g_k - b_{k+1} g_{k+1} = (m_k g_0 + \ell_k g_1) - b_{k+1}(m_{k+1} g_0 + \ell_{k+1} g_1) = m_{k+2} g_0 + \ell_{k+2} g_1 \quad (5.4\text{-}42)$$

Folglich teilt jeder gemeinsame Teiler von g_0 und g_1 auch g_k , $k < K+2$.

Nun gilt es, g_{K+1} als Teiler von g_0 und g_1 zu identifizieren. Aus $g_{K+2} = 0$ folgt $g_K = b_{K+1} g_{K+1}$ und somit $g_{K+1} \,|\, g_K$. Zusammen mit $g_{K-1} = b_K g_K + g_{K+1}$ erhalten wir $g_{K+1} \,|\, g_{K-1}$. Endlich viele Schritten liefern das gewünschte Ergebnis. □

Dasselbe Verfahren läßt sich fast direkt auf den Polynomring $F[x]$ übertragen, einzig die Betragsfunktion $\mathbb{Z} \xrightarrow{|\,|} I\!N$ ist durch die Gradfunktion auf Polynomen zu ersetzen, wobei wir $\mathrm{grd}\, 0 = -\infty$ vereinbaren wollen.

Startet man diesen Algorithmus mit den Polynomen $g_0(x) = x^{d-1}$ und $g_1(x) = S(x)$, so sind für jedes k vor Beendigung des Algorithmus die Polynome

$$g_k(x) = m_k(x) x^{d-1} + \ell_k(x) S(x) \qquad \text{und} \qquad \ell_k(x) S(x)$$

modulo x^{d-1} äquivalent. Existiert ein $k_0 < K+2$, so daß $\ell_{k_0}(x)$ und $g_{k_0}(x)$ der abgeschwächten Gradbedingung (5.3-41) $\mathrm{grd}\, \ell_{k_0}(x) \leq (d-1)/2 > \mathrm{grd}\, g_{k_0}(x)$ genügen, dann unterscheidet sich $\langle \ell_{k_0}(x), g_{k_0}(x) \rangle$ gemäß Proposition 5.3.02 nur um einen polynomialen Faktor von $\langle \Lambda(x), \Gamma(x) \rangle$. Um einen derartigen Index k_0 zu finden, stellen wir zunächst fest, daß $\mathrm{grd}\, g_k(x)$ nach Konstruktion strikt monoton mit k fällt. Der Nachweis des streng monotonen Wachstums von $\mathrm{grd}\, \ell_k(x)$ mit k ist dagegen aufwendiger:

5.4.02 Lemma. *Für* $0 < k < K+2$ *gilt* $\operatorname{grd} \ell_k(x) = d - 1 - \operatorname{grd} g_{k-1}(x)$.

Beweis: Wir verwenden Induktion über k : Wegen $\operatorname{grd} \ell_1(x) = 0$ und $\operatorname{grd} g_0(x) = d-1$ stimmt die Behauptung für $k = 1$. Wir nehmen an, sie ist für alle $0 < k < M+2$ mit $M < K$ erfüllt. Nachzuweisen ist nun $\operatorname{grd} \ell_{M+2}(x) = d - 1 - \operatorname{grd} g_{M+1}(x)$.

Nach Definition gilt $\ell_{M+2}(x) = \ell_M(x) - b_{M+1}(x)\ell_{M+1}(x)$. Wir betrachten die Summanden einzeln. $b_{M+1}(x)$ hat nach Konstruktion den positiven Grad $\operatorname{grd} g_M(x) - \operatorname{grd} g_{M+1}(x)$. Zusammen mit der Induktionsvoraussetzung für $\ell_{M+1}(x)$ erhalten wir

$$\begin{aligned}\operatorname{grd} b_{M+1}(x)\ell_{M+1}(x) &= \operatorname{grd} b_{M+1}(x) + \operatorname{grd} \ell_{M+1}(x)\\ &= \operatorname{grd} g_M(x) - \operatorname{grd} g_{M+1}(x) + d - 1 - \operatorname{grd} g_M(x)\\ &= d - 1 - \operatorname{grd} g_{M+1}(x)\end{aligned}$$

Wendet man die Induktionsvoraussetzung auf $\ell_M(x)$ an und berücksichtigt, daß der Grad von $g_k(x)$ streng monoton mit k fällt, so folgt

$$\operatorname{grd} \ell_M(x) = d - 1 - \operatorname{grd} g_{M-1}(x) < d - 1 - \operatorname{grd} g_M(x) < d - 1 - \operatorname{grd} g_{M+1}(x)$$

Damit kann der erste Summand nichts zum Grad von $\ell_{M+2}(x)$ beitragen, der folglich den Wert $d - 1 - \operatorname{grd} g_{M+1}(x)$ hat. □

Setzen wir nun $k_0 := \min\{\, k : \operatorname{grd} g_k(x) < (d-1)/2 \,\}$, so gilt $\operatorname{grd} g_{k_0-1}(x) \geq (d-1)/2$ und folglich $\operatorname{grd} \ell_{k_0}(x) = d - 1 - \operatorname{grd} g_{k_0-1}(x) \leq (d-1)/2$. Somit ist die abgeschwächte Gradbedingung (5.3-41) in der Tat erfüllt. Zu bestimmen bleibt der polynomiale Faktor, um den sich $\langle \ell_{k_0}(x), g_{k_0}(x) \rangle$ und $\langle \Lambda(x), \Gamma(x) \rangle$ unterscheiden.

5.4.03 Bemerkung. Es ist nicht offensichtlich, daß $\ell_{k_0}(x)$ und $g_{k_0}(x)$ auch die ursprüngliche schärfere Gradbedingung 5.3-38 erfüllen. Aus diesem Grund war es notwendig, in Proposition 5.3.02 die abgeschwächte Gradbedingung (5.3-41) als Voraussetzung zu verwenden. Sobald wir feststellen, daß sich $\langle \ell_{k_0}(x), g_{k_0}(x) \rangle$ und $\langle \Lambda(x), \Gamma(x) \rangle$ nur um einen konstanten Faktor unterscheiden, folgt für $\ell_{k_0}(x)$ und $g_{k_0}(x)$ auch die ursprüngliche Gradbedingung.

5.4.04 Proposition. *Es gilt* $\ell_{k_0}(x) = \ell_{k_0}(0)\Lambda(x)$ *und* $g_{k_0}(x) = \ell_{k_0}(0)\Gamma(x)$.

Beweis: Wir wählen $k_1 := \min\{\, k : \operatorname{grd} g_k(x) \leq \operatorname{grd} \Gamma(x) \,\} \geq k_0$, insbesondere ergibt sich daraus $\operatorname{grd} \Gamma(x) < \operatorname{grd} g_{k_1-1}(x)$. In Analogie zu (5.4-42) haben wir

$$g_{k_1}(x) = m_{k_1}(x) \cdot x^{d-1} + \ell_{k_1}(x) \cdot S(x)$$

während aufgrund von (5.3-39) ein Polynom $M(x)$ existiert mit

$$\Gamma(x) = M(x) \cdot x^{d-1} + \Lambda(x) \cdot S(x)$$

Elimination von $S(x)$ liefert nun

$$\Lambda(x) \cdot g_{k_1}(x) - \ell_{k_1}(x) \cdot \Gamma(x) = \big(\Lambda(x) \cdot m_{k_1}(x) - \ell_{k_1}(x) \cdot M(x)\big) \cdot x^{d-1}$$

Aufgrund der Gradbedingung (5.3-38) gilt

$$\operatorname{grd} \Lambda(x) + \operatorname{grd} g_{k_1}(x) < d-1$$

Andererseits liefert Lemma 5.4.02

$$\operatorname{grd} \Gamma(x) + \operatorname{grd} \ell_{k_1}(x) = \operatorname{grd} \Gamma(x) + d - 1 - \operatorname{grd} g_{k_1-1}(x) < d-1$$

Folglich hat die linken Seite einen Grad kleiner als $d-1$, woraus unmittelbar $\Lambda(x) \cdot g_{k_1}(x) = \ell_{k_1}(x) \cdot \Gamma(x)$ folgt. Da sowohl $\Lambda(x)$ als auch $\ell_{k_1}(x)$ von 0 verschieden sind, muß sich aufgrund von $\operatorname{ggT}(\Lambda(x), \Gamma(x)) = 1$ auch $\langle \ell_{k_1}(x), g_{k_1}(x)\rangle$ um einen polynomialen Faktor von $\langle \Lambda(x), \Gamma(x)\rangle$ $c(x)$ unterscheiden, ebenso wie $\langle \ell_{k_0}(x), g_{k_0}(x)\rangle$. Nach Lemma 5.4.02 folgt nun sofort $k_1 = k_0$. Wegen $\operatorname{grd} g_{k_1}(x) \le \operatorname{grd} \Gamma(x)$ und $\Lambda(0) = 1$ ist der polynomiale Faktor die Konstante $\ell_{k_0}(0)$. □

Schließlich wollen wir dir Berechnung der Fehlerwerte etwas vereinfachen.

5.4.05 Definition. Für ein Polynom $f(x) = \sum_{i<n} f_i x^i \in F[x]$ definieren wir seine *formale Ableitung*

$$f'(x) := \sum_{i<n} ((i+1) \bullet f_{i+1})\, x^i$$

wobei $(i+1) \bullet f_{i+1}$ als $(i+1)$-fache Summe von f_{i+1} in F aufzufassen ist.

5.4.06 Proposition. *Für alle von* 0 *verschiedenen Polynome* $f(x)$ *und* $g(x)$ *und jedes Körperelement* $a \in F$ *gilt:*

(0) $\big(f(x) + g(x)\big)' = f'(x) + g'(x)$;

(1) $\big(af(x)\big)' = af'(x)$;

(2) $\big(f(x)g(x)\big)' = f'(x)g(x) + f(x)g'(x)$.

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.

$$\begin{aligned}\big(x^m x^n\big)' = \big(x^{m+n}\big)' &= (m+n)x^{(m+n)-1} \\ &= mx^{m-1+n} + nx^{n-1+m} = \big(mx^{m-1}\big)x^n + x^m\big(nx^{n-1}\big)\end{aligned}$$

Der Rest ist nun trivial. □

Wir betrachten nun die Ableitung von $\Lambda(x)$:

$$\Lambda'(x) = \sum_{j\in J}(-\alpha_j) \prod_{m\in J\setminus\{j\}} (1-\alpha_m x)$$

Kombiniert man für $j \in J$

$$\Lambda'\left(\alpha_j^{-1}\right) = -\alpha_j \prod_{m\in J\setminus\{j\}} (1-\alpha_m\alpha_j^{-1}) \qquad \text{mit} \qquad \Gamma\left(\alpha_j^{-1}\right) = e_j v_j \prod_{m\in J\setminus\{j\}} (1-\alpha_m\alpha_j^{-1})$$

so erhält man schließlich

$$\boxed{e_j = -\frac{\alpha_j}{v_j}\cdot\frac{\Gamma(\alpha_j^{-1})}{\Lambda'(\alpha_j^{-1})}}$$

5.4.07 Beispiel. Mit dem hier beschriebenen Verfahren läßt sich Beispiel 5.3.03 effizienter lösen.

$$\begin{aligned} x^4 &= (\xi^{14}x+\xi^4)(\xi x^3+\xi^6x^2+\xi^{14}x+\xi^{12}) + (\xi^9x^2+\xi^5x+\xi) \\ \xi x^3+\xi^6x^2+\xi^{14}x+\xi^{12} &= (\xi^7x+\xi^{10})(\xi^9x^2+\xi^5x+\xi) + (\xi^{13}x+1) \end{aligned}$$

Damit hat erstmals $g_3(x) = x+\xi^{10}$ einen Grad kleiner als 2. Wir erhalten

$$\begin{aligned} \ell_2(x) &= 0 - b_1(x)\cdot\ell_1(x) = \xi^{14}x+\xi^4 \\ \ell_3(x) &= 1 - b_2(x)\cdot\ell_2(x) = 1-(\xi^7x+\xi^{10})(\xi^{14}x+\xi^4) = \xi^6x^2+\xi^2x+\xi^3 \end{aligned}$$

Mit $\ell_3(0) = \xi^3$ erhalten wir $\Lambda(x) = \xi^3x^2+\xi^{14}x+1$ sowie $\Gamma(x) = \xi^{10}x+\xi^{12}$, wie zuvor. Die Ableitung des ELP ergibt sich zu

$$\Lambda'(x) = \xi^{14}$$

Mit den vorher bestimmten Nullstellen $x_0 = 1$ und $x_1 = \xi^{12}$ des ELP berechnen sich die Fehlerwerte zu

$$e_0 = \frac{\Gamma(1)}{\xi^{14}} = \xi^{3-14} = \xi^4 \qquad \text{sowie} \qquad e_3 = \frac{\xi^3\cdot\Gamma\left(\xi^{12}\right)}{\xi^{14}} = \xi^{3+2-14} = \xi^6$$

5.4.08 Bemerkung. Da der Euklidische Algorithmus bei der schnellen Decodierung auf Polynomdivision beruht, ist die Polynomdarstellung mit fallenden Potenzen von x vorzuziehen. Die Standardform der Kontrollmatrix H eines vRS Codes liefert allerdings das Syndrom passend für die aufsteigende Polynomdarstellung. Insofern wäre aus technischen Gründen eine Kontrollmatrix $\bar{H}$ vorzuziehen, bei der die Reihenfolge der Spalten im Vergleich zu H umgekehrt ist.

Abschließend fassen wir den Algorithmus zur schnellen Decodierung verallgemeinerter Reed-Solomon Codes mit Lokalisierer α und Gewicht $\boldsymbol{v}$ zusammen. Er besteht aus vier Schritten:

(0) Berechnung des Syndroms $\boldsymbol{S} = \boldsymbol{y}\cdot H$ bzw. $\bar{\boldsymbol{S}} = \boldsymbol{y}\cdot\bar{H}$; dies liefert in beiden Fällen das Syndrompolynom $S(x) := \sum_{i<2\tau} S_i x^i = \sum_{i<2\tau}\bar{S}_i x^{2\tau-1-i}$;

(1) Anwendung des Euklidischen Algorithmus auf $g_0(x) = x^{2\tau}$ und $g_1(x) = S(x)$; Abbruch sobald $\operatorname{grd} g_K(x) < \tau$ gilt;

(2) rekursive Berechnung von $\ell_K(x)$; Multiplikation von $\ell_K(x)$ und $g_K(x)$ mit $(\ell_K(0))^{-1}$ liefert $\Lambda(x)$ bzw. $\Gamma(x)$;

(3) Bestimmung aller Wurzeln von $\Lambda(x)$; Korrektur von y_j genau dann, wenn α_j^{-1} eine Wurzel ist, und zwar auf den Wert $y_j + \alpha_j\Gamma(\alpha_j^{-1})/v_j\Lambda'(\alpha_j^{-1})$.

5.5 Der Berlekamp-Massey Algorithmus

Neben der oben vorgestellten Abwandlung des Euklidischen Algorithmus zur Bestimmung des Fehlerstellenpolynoms $\Lambda(x)$ und des Fehlerwertepolynoms $\Gamma(x)$, gibt es noch einen zweiten verbreiteten Algorithmus von vergleichbarer Zeitkomplexität. Der *Berlekamp-Massey Algorithmus* betrachtet zu einem vorgegebenen Polynom $S(x) \in F[x]$ und $N \in \mathbb{N}$ sogenannte *N-Rekurrenzen* $\langle\lambda(x), \gamma(x)\rangle \in F[x]^2$ für $S(x)$, die folgenden Bedingungen genügen müssen

(R0) $\lambda(0) = 1$

(R1) $\lambda(x)S(x) \bmod x^N = \gamma(x) \bmod x^N$

Unter der *Rekurrenz-Ordnung* des Paares $\langle\lambda(x), \gamma(x)\rangle$ versteht man die Zahl

$$\operatorname{ord}\langle\lambda(x), \gamma(x)\rangle := \max\{\operatorname{grd}\lambda(x), 1 + \operatorname{grd}\gamma(x)\}$$

In Anbetracht von Proposition 5.3.02 ist das folgende Ergebnis nicht wirklich überraschend:

5.5.01 Proposition. *Ist $\langle\lambda(x), \gamma(x)\rangle$ eine N-Rekurrenz für $S(x)$ minimaler Ordnung, so sind beide Komponenten teilerfremd*

$$\operatorname{ggT}(\lambda(x), \gamma(x)) = 1$$

Gilt zudem $\operatorname{ord}\langle\lambda(x), \gamma(x)\rangle \le N/2$, so ist $\langle\lambda(x), \gamma(x)\rangle$ sogar eindeutig bestimmt.

Beweis: Falls $r(x) = \operatorname{ggT}(\lambda(x), \gamma(x))$, so bilden $r(0)\lambda(x)/r(x)$ und $r(0)\gamma(x)/r(x)$ ebenfalls eine N-Rekurrenz. Diese hat genau dann eine echt kleinere Rekurrenz-Ordnung als die ursprüngliche N-Rekurrenz, wenn $\operatorname{grd} r(x) > 0$ gilt.

Falls $\operatorname{grd}\lambda(x) < N$ ist $\lambda(x)$ als Element von $F[x]_{x^N}$ wegen $\lambda(0) = 1$ invertierbar. Damit können wir (R1) umformen zu $\lambda^{-1}(x)\gamma(x) \bmod x^N = S(x) \bmod x^N$. Jede weitere N-Rekurrenz $\langle\bar\lambda(x), \bar\gamma(x)\rangle$ für $S(x)$ mit $\operatorname{grd}\bar\lambda(x) < N$ erfüllt die entsprechende Bedingung. Haben beide N-Rekurrenzen eine Ordnung $\le N/2$, so gilt $\bar\lambda(x)\gamma(x) = \lambda(x)\bar\gamma(x)$ in $F[x]$. Sind zudem beide Ordnungen minimal, so schließen wir aus der Teilerfremdheit von $\lambda(x)$ und $\gamma(x)$ bzw. von $\bar\lambda(x)$ und $\bar\gamma(x)$, daß jeder Teiler dieses Produkts entweder gemeinsamer Teiler von $\lambda(x)$ und $\bar\lambda(x)$ oder gemeinsamer Teiler von $\gamma(x)$ und $\bar\gamma(x)$ sein muß. Wegen $\lambda(0) = 1 = \bar\lambda(0)$ und der Minimalität der Ordnungen folgt somit $\lambda(x) = \bar\lambda(x)$ und $\gamma(x) = \bar\gamma(x)$. □

Der Berlekamp-Massey Algorithmus berechnet ausgehend von $\langle 1, 0\rangle$ iterativ eine Folge von i-Rekurrenzen $\langle \lambda_i(x), \gamma_i(x)\rangle$ für $S(x)$, $i \le N$, die jeweils minimale Rekurrenz-Ordnung haben. Wählt man für $S(x)$ das Syndrom eines empfangenen Worts $\boldsymbol{y}$ und setzt $N := d-1$, so erhält man schließlich $\langle \lambda_{d-1}(x), \gamma_{d-1}(x)\rangle$ mit teilerfremden Komponenten, was mit $\langle \Lambda(x), \Gamma(x)\rangle$ übereinstimmen muß.

5.5.02 Algorithmus. (Berlekamp-Massey)

- Initialisierung: Wir setzen die Startwerte auf

$$\langle \lambda_{-1}(x), \gamma_{-1}(x)\rangle := \langle 0, -x^{-1}\rangle \qquad \text{sowie} \qquad \langle \lambda_0(x), \gamma_0(x)\rangle = \langle 1, 0\rangle$$

  und initialisieren zwei Hilfsvariablen zu $\mu := -1$ und $\varphi_{-1} := 1$.

- Iteration: Für $i < N$ wird mittels des Koeffizienten φ_i von x^i im Produkt $\lambda_i(x)S(x)$ eine $(i+1)$-Rekurenz berechnet:

$$\langle \lambda_{i+1}(x), \gamma_{i+1}(x)\rangle := \langle \lambda_i(x), \gamma_i(x)\rangle - \frac{\varphi_i}{\varphi_\mu} x^{i-\mu} \langle \lambda_\mu(x), \gamma_\mu(x)\rangle$$

  Falls $\varphi_i \neq 0$ und $2\,\mathrm{ord}\langle \lambda_i(x), \gamma_i(x)\rangle \le i$ gilt, wird anschließend $\mu := i$ gesetzt.

5.5.03 Beispiel. Wir wenden den Berlekamp-Massey Algorithmus auf die in den Beispielen 5.3.03 und 5.4.07 betrachtete Situation an. Mit $S(x) = \xi^{12} + \xi^{14}x + \xi^6 x^2 + \xi x^3$ und $N = d - 1 = 4$ ergibt sich

i	-1	0	1	2	3	4
$\lambda_i(x)$	0	1	1	$1 + \xi^2 x$	$1 + \xi^2 x + \xi^{14} x^2$	$1 + \xi^{14} x + \xi^3 x^2$
$\gamma_i(x)$	x^{-1}	0	ξ^{12}	ξ^{12}	ξ^{12}	$\xi^{12} + \xi^{10} x$
ord_i		0	1	1	2	
μ		-1	0	0	2	
φ_μ		1	ξ^{12}	ξ^{12}	ξ^{11}	
φ_i	1	ξ^{12}	ξ^{14}	ξ^{11}	ξ^9	

Man beachte, daß sich $\lambda_i(x)$ nicht nur durch Anfügen höherer Potenzen von x verändert. Im letzten Schritt gilt

$$\lambda_4(x) = 1 + \xi^2 x + \xi^{14} x^2 + \xi^{14} x(1 + \xi^2 x) = 1 + \xi^{12} x + \xi^3 x^2$$

was die Koeffizienten schon vorhandener Potenzen von x modifiziert.

Bevor wir den Nachweis der korrekten Funktionsweise führen können, benötigen wir ein technisches Lemma.

5.5.04 Lemma. *Für $i \in \mathbb{N}$ bezeichne ℓ_i die minimale Ordnung einer i-Rekurrenz für $S(x)$. Die Folge ℓ wächst schwach monoton und beginnt mit $\ell_0 = 0$. Falls für ein $i \in \mathbb{N}$ eine i-Rekurrenz für $S(x)$ existiert, die keine $(i+1)$-Rekurrenz ist, gilt weiterhin*

$$\ell_{i+1} \geq \max\{\ell_i, i+1-\ell_i\}$$

Beweis: Die 0-Rekurrenz $\langle 1, 0\rangle$ für $S(x)$ hat offenbar die Ordnung 0. Und da jede $(i+1)$-Rekurrenz für $S(x)$ auch eine i-Rekurrenz ist, kann die Folge ℓ nirgends fallen.

Wir betrachten einen Index i mit $\ell_{i+1} < i+1-\ell_i$, oder äquivalent $\ell_{i+1} + \ell_i \leq i$. Sind $\langle \lambda(x), \gamma(x)\rangle$ bzw. $\langle \bar\lambda(x), \bar\gamma(x)\rangle$ minimale i- bzw. $(i+1)$-Rekurrenzen, so folgt aufgrund der Invertierbarkeit von $\lambda(x) \bmod x^i$ wie auch $\bar\lambda(x) \bmod x^i$ in $F[x]_{x^i}$

$$\lambda^{-1}(x)\gamma(x) \bmod x^i = S(x) \bmod x^i = \bar\lambda^{-1}(x)\bar\gamma(x) \bmod x^i$$

oder äquivalent

$$\bar\lambda(x)\gamma(x) \bmod x^i = \lambda(x)\bar\gamma(x) \bmod x^i$$

Nach Voraussetzung wissen wir aber

$$\max\{\operatorname{grd}\bar\lambda(x), \operatorname{grd}\bar\gamma(x)+1\} + \max\{\operatorname{grd}\lambda(x), \operatorname{grd}\gamma(x)+1\} = \ell_{i+1} + \ell_i \leq i$$

woraus $\operatorname{grd}\lambda(x) + \operatorname{grd}\bar\gamma(x) < i$ wie auch $\operatorname{grd}\bar\lambda(x) + \operatorname{grd}\gamma(x) < i$ folgt. Damit gilt sogar in $F[x]$

$$\bar\lambda(x)\gamma(x) = \lambda(x)\bar\gamma(x)$$

Wegen der Teilerfremdheit der Komponenten beider Rekurrenzen (vergl. Proposition 5.5.01) müssen sie übereinstimmen, also ist $\langle \lambda(x), \gamma(x)\rangle$ ist auch eine $(i+1)$-Rekurrenz für $S(x)$. □

5.5.05 Proposition. *Für jedes $i \leq N$ ist $\langle \lambda_i(x), \gamma_i(x)\rangle$ eine i-Rekurrenz minimaler Ordnung für $S(x)$.*

Beweis: Mit b bezeichnen wir den maximalen Exponenten, so daß $S(x)$ durch x^b teilbar ist.

Für $0 \leq i < b$ gilt $\varphi_i = 0$ und folglich $\langle \lambda_{i+1}(x), \gamma_{i+1}(x)\rangle = \langle 1, 0\rangle$, was eine $(i+1)$-Rekurrenz minimaler Ordnung für $S(x)$ ist.

Für $i = b$ gilt $\varphi_i = S_b \neq 0$ und folglich $\langle \lambda_{i+1}(x), \gamma_{i+1}(x)\rangle = \langle 1, S_i x^i\rangle$, was wiederum eine $(i+1)$-Rekurrenz minimaler Ordnung für $S(x)$ ist. Erstmals wird der Index μ erhöht, und zwar auf den Wert $\mu = i$.

Für $b < i < N$ zeigen wir mittels Induktion über $i \geq b+1$, daß $\langle \lambda_{i+1}(x), \gamma_{i+1}(x)\rangle$ eine $(i+1)$-Rekurrenz minimaler Ordnung für $S(x)$ ist, die darüberhinaus $\operatorname{grd}\gamma_{i+1}(x) < i$ erfüllt. Nach Induktionsvoraussetzung mögen diese Bedingungen für $j < i$ erfüllt sein. Nach Konstruktion gilt

$$\lambda_i(x)S(x) \bmod x^i = \gamma_i(x) \bmod x^i \quad \text{und} \quad \lambda_\mu(x)S(x) \bmod x^\mu = \gamma_\mu(x) \bmod x^\mu$$

wobei μ mindestens den Wert 0 hat. Das Produkt der μ-Rekurrenz mit $x^{i-\mu}$ erfüllt Bedingung (R1) bzgl. x^i, und da diese Eigenschaft unter Linearkombinationen erhalten bleibt, wird sie auch von $\langle\lambda_{i+1}(x), \gamma_{i+1}(x)\rangle$ erfüllt. Wegen $i - \mu > 0$ gilt weiterhin $\lambda_{i+1}(0) = \lambda_i(0) = 1$, damit ist $\langle\lambda_{i+1}(x), \gamma_{i+1}(x)\rangle$ eine i-Rekurrenz.

Um zu zeigen, daß es sich auch um eine $(i+1)$-Rekurrenz handelt, betrachten wir die Koeffizienten von x^i in $\lambda_{i+1}(x)S(x)$ und in $\gamma_{i+1}(x)$. Nach Konstruktion hat x^i in $\lambda_i(x)S(x)$ ebenso wie in $(\varphi_I/\varphi_\mu)x^{i-\mu}\lambda_\mu(x)S(x)$ den Koeffizienten φ_i, so daß der entsprechende Koeffizient der Differenz den Wert 0 hat. Andererseits folgt nach Induktionsvoraussetzung und Konstruktion von $\gamma_{i+1}(x)$

$$\operatorname{grd}\gamma_{i+1}(x) \le \max\{\operatorname{grd}\gamma_i(x), i - \mu + \operatorname{grd}\gamma_\mu(x)\} < i$$

Also hat der Koeffizient von x^i in $\gamma_{i+1}(x)$ ebenfalls den Wert 0.

Falls $\delta_i = 0$, so stimmt $\langle\lambda_{i+1}(x), \gamma_{i+1}(x)\rangle$ mit $\langle\lambda_i(x), \gamma_i(x)\rangle$ überein. Da jede $(i+1)$-Rekurrenz für $S(x)$ auch eine i-Rekurrenz ist, folgt die Minimalität der Ordnung von $\langle\lambda_i(x), \gamma_i(x)\rangle$ als $(i+1)$-Rekurrenz nach Induktionsvoraussetzung.

Schließlich betrachten wir den Fall $\delta_i \neq 0$. Zur Abkürzung setzen wir $\operatorname{ord}_j := \operatorname{ord}\langle\lambda_j(x), \gamma_j(x)\rangle$ für $j \le i+1$. Um Lemma 5.5.04 anwenden zu können, wollen wir folgende Ungleichung herleiten

$$\operatorname{ord}_{i+1} \le \max\{\operatorname{ord}_i, i + 1 - \operatorname{ord}_i\}$$

die für Indexwerte $j \le b$ offenbar erfüllt ist. Nach Konstruktion wissen wir bereits

$$\operatorname{ord}_{i+1} \le \max\{\operatorname{ord}_i, i - \mu + \operatorname{ord}_\mu\}$$

Also genügt es, $i - \mu + \operatorname{ord}_\mu \le i + 1 - \operatorname{ord}_i$ zu zeigen, was zu $\operatorname{ord}_i \le \mu + 1 - \operatorname{ord}_\mu$ äquivalent ist.

Da in Schritt b der Index μ auf 0 gesetzt wurde, erfüllt der momentan aktuelle Index μ die Bedingungen $\varphi_\mu \neq 0$ und $2\operatorname{ord}_\mu \le \mu$, nach Induktionsvoraussetzung also

$$\ell_{\mu+1} = \operatorname{ord}_{\mu+1} \le \max\{\operatorname{ord}_\mu, (\mu+1) - \mu + \operatorname{ord}_\mu\} = 1 + \operatorname{ord}_\mu \le \mu + 1 - \operatorname{ord}_\mu$$

Für jeden Index $\mu < j < i$ muß im Falle $\varphi_j \neq 0$ gelten $2\operatorname{ord}_j > j$, was nach Induktionsvoraussetzung impliziert

$$\ell_{j+1} = \operatorname{ord}_{j+1} \le \max\{\operatorname{ord}_j, j + 1 - \operatorname{ord}_j\} = \operatorname{ord}_j$$

Die Kombination dieser Ergebnisse liefert

$$\operatorname{ord}_i \le \operatorname{ord}_{i-1} \le \ldots \operatorname{ord}_{\mu+1} \le \mu + 1 - \operatorname{ord}_\mu$$

was wir zeigen wollten.

Wegen $\varphi_i \neq 0$ und $\operatorname{grd}\gamma_i < i - 1$ ist $\langle\lambda_i(x), \gamma_i(x)\rangle$ keine $(i+1)$-Rekurrenz für $S(x)$, somit gilt nach Lemma 5.5.04 nebst Induktionsvoraussetzung und obigem Ergebnis

$$\ell_{i+1} \ge \max\{\ell_i, i + 1 - \ell_i\} = \max\{\operatorname{ord}_i, i + 1 - \operatorname{ord}_i\} \ge \operatorname{ord}_{i+1} \ge \ell_{i+1}$$

also $\operatorname{ord}_{i+1} = \ell_{i+1}$. □

6 Zyklische Codes

Wir wollen nun eine Teilklasse der linearen Codes betrachten, die aufgrund zusätzlicher algebraischer Eigenschaften besonders einfach zu codieren und zu decodieren ist.

6.1 Die Definition zyklischer Codes

6.1.01 Definition. Ein linearer $[n,k]$-Code $\mathcal{C}$ über F heißt *zyklisch*, wenn er unter zyklischer Vertauschung der Codewörter abgeschlossen ist, d.h.,

$$\boldsymbol{c}^{(0)} := (\, c_0\, c_1\, \ldots\, c_{n-1}\,) \in \mathcal{C} \quad \text{impliziert} \quad \boldsymbol{c}^{(1)} := (\, c_1\, c_2\, \ldots\, c_{n-1}\, c_0\,) \in \mathcal{C}$$

Die Polynomdarstellung, ob steigend oder fallend, ermöglicht eine besonders einfache algebraische Beschreibung der zyklischen Verschiebung durch Multiplikation mit x:

$$\text{steigend: } c^{(n-1)}(x) = c_{n-1} + \sum_{i<n-1} c_i x^{i+1} = x \cdot c(x) - c_{n-1}(x^n - 1)$$

$$\text{fallend: } \quad \bar{c}^{(1)}(x) = \sum_{0<i<n} c_i x^{n-i} + c_0 = x \cdot \bar{c}(x) - c_0(x^n - 1)$$

Im Faktorring $F[x]_{x^n-1}$ entspricht die Multiplikation mit x einer zyklischen Verschiebung in Richtung steigender Exponenten.

Wegen der Linearität zyklischer Codes gehören für jedes $\boldsymbol{c} \in \mathcal{C}$ und $\boldsymbol{u} \in F^n$ die zu den Produkten $u(x) \cdot c(x) \bmod (x^n - 1)$ und $\bar{u}(x) \cdot \bar{c}(x) \bmod (x^n - 1)$ gehörigen Vektoren ebenfalls zu $\mathcal{C}$. Somit ist $\mathcal{C}$ ein *Ideal* des Faktorrings $F[x]_{x^n+1}$.

6.1.02 Beispiele.

(a) Paritätscodes sind zyklisch, denn sie sind durch die Bedingung spezifiziert, daß die Komponentensumme den Wert 0 hat, was unter zyklischer Vertauschung invariant ist.

(b) Wiederholungscodes sind trivialerweise zyklisch.

(c) Konventionelle Reed-Solomon Codes über $F = \mathrm{GF}(q)$ sind ebenfalls zyklisch, vergl. Abschnitt 4.2. Deren Codewortpolynome sind durch vorgegebene Wurzeln charakterisiert, die Potenzen β^{b+l}, $l < d-1$ für ein Element $\beta \in F$ der Ordnung n, siehe (4.2-27) bzw. (4.2-27). Nach Voraussetzung gilt $\beta^n = 1$, somit folgt aus $\bar{c}\left(\beta^{b+l}\right) = 0$ unmittelbar

$$\bar{c}^{(1)}\left(\beta^{b+l}\right) = \beta^{b+l} \cdot \bar{c}\left(\beta^{b+l}\right) - c_0 \cdot \left(\beta^{(b+l)\cdot n} - 1\right) = \beta^{b+l} \bar{c}\left(\beta^{b+l}\right) = 0$$

Bei steigender Polynomdarstellung betrachtet man mit die zyklische Verschiebung $c^{(n-1)}(x)$.

(d) Als Konsequenz aus Teil (c) sind BCH-Codes ebenfalls zyklisch.

(e) Verallgemeinerte Reed-Solomon-Codes brauchen i.A. nicht zyklisch zu sein: über $F = \mathrm{GF}(3)$ betrachten wir die Generatormatrix

$$G_{\mathrm{vRS}} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}$$

vergl. (4.1-24) mit Lokalisierer $\alpha = \langle 1\ 2 \rangle$ und Spaltengewicht $\boldsymbol{v} = \langle 1\ 1 \rangle$. Offenbar ist $\langle 2\ 1 \rangle$ kein Codewort.

6.1.03 Beispiel. Wir wollen eine hinreichende Bedingung für die Zyklizität der Hamming Codes über $G = \mathrm{GF}(q)$ aus Beispiel 1.2.08 finden. Für $m > 1$ setzen wir $n := (q^m - 1)/(q - 1)$ und wählen $\beta \in \mathrm{GF}(q^m)$ mit der Ordnung n.

Nun betrachten wir den gemäß Beispiel 6.1.02(d) zyklischen BCH-Code $\mathcal{C}$ der Länge n über F mit der $\mathrm{GF}(q^m)$-wertigen Kontrollmatrix

$$\begin{pmatrix} 1 \\ \beta \\ \vdots \\ \beta^{n-1} \end{pmatrix}$$

Diese entspricht einer $(n \times m)$-Kontrollmatrix über F. Um den Minimalabstand $d \geq 3$ zu garantieren, müssen je zwei ihrer Zeilen linear unabhängig sein, das heißt, β^i und β^j dürfen sich für $i < j < n$ nicht um einen skalaren Faktor aus F unterscheiden. Wegen $\beta^0 = 1$ ist dies äquivalent zur Forderung $\beta^l \notin F$, für $0 < l < n$.

Da die multiplikative Gruppe von F die Ordnung $q-1$ hat, impliziert $\beta^l \in F$ natürlich $\left(\beta^l\right)^{q-1} = 1$. Um $\beta^l \notin F$ garantieren zu können, genügt es also, $\left(\beta^l\right)^{q-1} \neq 1$ zu fordern.

Sofern also $\operatorname{ord} \beta = n$ keines der Produkte $l(q-1)$, $l < n$, teilt, mit anderen Worten, falls $\mathrm{ggT}(n, q-1) = 1$, sind tatsächlich je zwei Zeilen der F-wertigen Kontrollmatrix linear unabhängig. $\mathcal{C}$ ist also ein $[n,k,d]_q$-Code mit $k \geq n - m$ und $d \geq 3$. Gemäß Beispiel 1.2.08 muß es sich bei $\mathcal{C}$ aber um den $[n, m-n, 3]_q$- Hamming-Code handeln, da bei vorgegebenem Minimalabstand $d = 3$ die Codedimension k den Wert $n - m$ nicht überschreiten kann.

6.2 Generatorpolynome

In Abschnitt 4.2 hatten wir für konventionelle Reed-Solomon Codes den Begriff des Generatorpolynoms eingeführt. Dieser erweist sich auch für allgemeine zyklische Codes als nützlich.

6.2.01 Satz. *Zu jedem zyklischen $[n,k]$-Code $\mathcal{C}$ über F, aufgefaßt als Unterraum von $F_n[x]$, gibt es ein Polynom $g(x) \in \mathcal{C}$ mit $\operatorname{grd} g(x) = n - k$ und $\mathcal{C} = g(x) \cdot F[x]_{x^n-1}$. Damit ist $\mathcal{C}$ ein Hauptideal von $F[x]_{x^n-1}$.*

Beweis: Wir wählen $g \in \mathcal{C} \setminus \{\boldsymbol{0}\}$ so daß $\operatorname{grd} g(x) = r$ minimal ist. Da wir $\mathcal{C}$ oben bereits als Ideal von $F[x]_{x^n-1}$ identifiziert haben, folgt $g(x) \cdot F[x]_{x^n-1} \subseteq \mathcal{C}$.

Umgekehrt stellen wir $v(x) \in \mathcal{C}$ als $q(x)g(x) + s(x)$ mit $\operatorname{grd} s(x) < r$ dar. Wegen $q(x)g(x) \in \mathcal{C}$ folgt auch $s(x) \in \mathcal{C}$. Die Minimalität von $\operatorname{grd} g(x)$ impliziert also $s(x) = 0$.

Folglich gilt $\mathcal{C} = \{\, q(x)g(x) : \operatorname{grd} q(x) < n-r \,\}$. Damit stimmt $k = \dim \mathcal{C}$ mit der Anzahl der frei wählbaren Koeffizienten von $q(x)$ überein, woraus $r = n - k$ folgt. □

6.2.02 Definition. Ist $\mathcal{C}$ ein zyklischer Code über F, so heißt ein von 0 verschiedenes Polynom $g(x) \in \mathcal{C}$ minimalen Grades ein *Generatorpolynom* von $\mathcal{C}$.

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

Offenbar bilden für ein Generatorpolynom $g(x)$ eines $[n,k]$-Codes $\mathcal{C}$ über F die Produkte $x^i \cdot g(x)$, $i < k$, eine Basis von $\mathcal{C}$. Somit erhalten wir aufsteigend eine Generatormatrix in Form einer *Bandmatrix*:

$$G = \begin{pmatrix} g(x) \\ xg(x) \\ \vdots \\ x^{k-1}g(x) \end{pmatrix} = \begin{pmatrix} g_0 & \dots & g_{n-k} & 0 & \dots & \dots & 0 \\ 0 & g_0 & \dots & g_{n-k} & 0 & \dots & 0 \\ \vdots & & & & & & \vdots \\ 0 & \dots & \dots & 0 & g_0 & \dots & g_{n-k} \end{pmatrix}$$

Fallend erhalten wir eine Bandmatrix derselben Form, wenn wir die Faktoren x^i, $i < n-k$, in umgekehrter Reihenfolge durchlaufen und $\bar{g}_i = g_{n-1-i}$ setzen:

$$\bar{G} = \begin{pmatrix} x^{k-1}g(x) \\ \vdots \\ xg(x) \\ g(x) \end{pmatrix} = \begin{pmatrix} \bar{g}_0 & \dots & \bar{g}_{n-k} & 0 & \dots & \dots & 0 \\ 0 & \bar{g}_0 & \dots & \bar{g}_{n-k} & 0 & \dots & 0 \\ \vdots & & & & & & \vdots \\ 0 & \dots & \dots & 0 & \bar{g}_0 & \dots & \bar{g}_{n-k} \end{pmatrix}$$

6.2.04 Beispiele. (0) Paritätscode: Über jedem Körper ist $x-1$ das gradniedrigste normierte Polynom mit Komponentensumme 0. Etwa für $n = 5$ erhalten wir die Generatormatrizen

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

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

Die Frage, welche Polynome als Generatorpolynome auftreten können, hat eine überraschend einfache Antwort.

6.2.05 Satz. *Die (normierten) Generatorpolynome zyklischer Codes* $\mathcal{C} \leq F^n$ *sind genau die (normierten) Teiler von* $x^n - 1$, *modulo* $x^n - 1$.

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 die zyklische Vertauschung um k Positionen $w(x) = x^k g(x) - x^n + 1$. Weiterhin gelte $x^n - 1 = q(x)g(x) + s(x)$ mit $\operatorname{grd} s(x) < \operatorname{grd} g(x)$. Auflösen nach $s(x)$ liefert

$$\begin{aligned} s(x) &= -q(x)g(x) + x^n - 1 \\ &= -q(x)g(x) - w(x) + x^k g(x) \\ &= \Big(x^k - q(x)\Big)\, g(x) - w(x) \end{aligned}$$

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

($\Leftarrow$) $g(x)$ 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 $\mathcal{C}$ ist offensichtlich und ebenso, daß $g(x)$ unter den von $\boldsymbol{0}$ verschiedenen Elementen von $\mathcal{C}$ minimalen Grad hat. Nachzuweisen bleibt, daß $\mathcal{C}$ zyklisch ist. Da die Wörter $g(x)$, $xg(x)$, ..., $x^{k-1}g(x)$ eine Basis von $\mathcal{C}$ bilden, ist die Zyklizität von U äquivalent dazu, daß die zyklische Verschiebung $w(x) = x^k g(x) - x^n + 1$ um k Positionen wieder zu $\mathcal{C}$ gehört, denn die zyklische Verschiebung eines Codeworts ist eine Linearkombination der entsprechenden zyklischen Verschiebungen der Basiselemente. Nach Voraussetzung gilt $w(x) = g(x)\big(x^k - q(x)\big)$ und $\operatorname{grd}\big(x^k - q(x)\big) < k$, woraus $w(x) \in \mathcal{C}$ folgt. Für nicht normierte echte Teiler von $x^n - 1$ argumentiert man analog. □

6.2.06 Beispiel. Wir bestimmen alle zyklischen binären Codes der Länge 7, indem wir $x^7 - 1$ über $\mathbb{Z}_2$ vollständig faktorisieren:

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

$g(x)$	$\mathcal{C}$
1	$\mathbb{Z}_2^7$
$x^7 - 1 \cong 0 \bmod (x^7 - 1)$	$\{\boldsymbol{0}\}$
$x - 1$	Paritätscode
$x^6 + x^5 + x^4 + x^3 + x^2 + x^1 + 1$	Wiederholungscode
$x^3 + x + 1$	$(7,4)$-Hammingcode
$x^3 + x^2 + 1$	$(7,4)$-Hammingcode
$(x-1)\left(x^3 + x + 1\right)$	ausgedünnter Hammingcode
$(x-1)\left(x^3 + x^2 + 1\right)$	ausgedünnter Hammingcode

Man beachte, daß das Produkt teilerfremder Generatorpolynome das Generatorpolynom des Durchschnittscodes liefert; im allgemeinen Fall ist dafür das kleinste gemeinsame Vielfache kgV der Generatorpolynome zuständig. Speziell die Multiplikation mit $x - 1$ entspricht der Ausdünnung, d.h., dem Durchschnitt mit dem Code der Wörter mit Komponentensumme 0 (vergl. Beispiel 1.0.03 und Definition 1.2.09(5)).

Die beiden trivialen Faktoren 1 und $x^7-1 \cong 0 \bmod (x^7-1)$ sind natürlich für Codierungszwecke uninteressant.

Die Codierung zyklischer Codes kann nun genauso wie in Abschnitt 4.3 für konventionelle Reed-Solomon Codes erfolgen, unsystematisch durch Multiplikation des Klartextpolynoms mit dem Generatorpolynom, oder des Klartextvektors mit einer Generatormatrix, oder systematisch mittels Polynomdivision via $x^{n-k} \cdot u(x) - s(x)$ (fallend).

6.3 Syndrome und die Decodierung zyklischer Codes

Wie bei allen linearen Codes kann die Decodierung mittels Standardtabelle erfolgen, siehe Abschnitt 1.3.1, wie auch durch Syndrom-Decodierung, siehe Abschnitt 1.3.2. Aufgrund der Abgeschlossenheit unter zyklischer Vertauschung läßt sich dieses Verfahren allerdings effizienter gestalten. Anstelle einer vollständigen Tabelle, die alle zu korrigierenden Fehlermuster mit Syndromen in Beziehung setzt, kann man sich auf eine *reduzierte Syndromliste* beschränken, in der nur bestimmte Fehlermuster maximalen Grades vorkommen.

Zunächst wollen wir uns der Bestimmung von Syndromen empfangener Wörter $\boldsymbol{y}$ zuwenden. Bisher hatten wir Syndrome als Produkt von $\boldsymbol{y}$ mit einer Kontrollmatrix berechnet. Also stellt sich die Frage nach kanonischen Kontrollmatrizen für zyklische Codes. Weil das Generatorpolynom $g(x)$ sehr direkt zu einer Generatormatrix führt, bietet sich die Betrachtung des Quotienten $(x^n - 1)/g(x)$ an.

6.3.01 Definition. Ist $g(x)$ Generatorpolynom eines zyklischen $[n,k]$-Codes über F, so bezeichnen wir $h(x) := (x^n - 1)/g(x)$ als das zugehörige *Kontrollpolynom*.

Für $g(x) = \sum_{i \le n-k} g_i x^i$ und $h(x) = \sum_{j \le k} h_j x^j$ schreiben wir die definierenden Gleichung um:

$$-1 + x^n = g(x)h(x) = g_0 h_0 + \sum_{0<l<n} \left(\sum_{i+j=l} g_i h_j \right) x^l + g_{n-k} h_k x^n$$

Daraus schließen wir

$$g_0 h_0 = -1 \quad , \quad g_{n-k} h_k = 1 \quad \text{sowie} \quad \sum_{i+j=l} g_i h_j = 0 \quad \text{für alle } 0 < l < n$$

Fallende Polynomschreibweise vertauscht die Vorzeichen in den ersten beiden Bedingungen. Diese besagen, daß $g(x)$ genau dann normiert ist, wenn dies für $h(x)$ gilt, und daß 0 weder Wurzel von $g(x)$ noch von $h(x)$ ist. In beiden Fällen erlaubt uns die letzte Bedingung, die Koeffizienten von $h(x)$ in einer Spalten-Bandmatrix zu organisieren, deren Produkt von rechts mit der Generatormatrix die Nullmatrix liefert. Dabei ist zu beachten, daß die Indizes von g_i und h_j in entgegengesetzter Richtung laufen. Mit $\bar{h}_i := h_{k-i}$ erfüllen folgende Matrizen

$G \cdot \bar{H} = 0 = \bar{G} \cdot H$:

$$\bar{H} = \begin{pmatrix} \bar{h}_0 & 0 & \dots & 0 \\ \vdots & \bar{h}_0 & & \vdots \\ \bar{h}_k & \vdots & & 0 \\ 0 & \bar{h}_k & & \bar{h}_0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & \bar{h}_k \end{pmatrix} \qquad \text{bzw.} \qquad H = \begin{pmatrix} h_0 & 0 & \dots & 0 \\ \vdots & h_0 & & \vdots \\ h_k & \vdots & & 0 \\ 0 & h_k & & h_0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & h_k \end{pmatrix}$$

6.3.02 Corollar. *Der Dualcode $\mathcal{C}^\perp$ eines zyklischen $[n,k]$-Codes $\mathcal{C}$ der Länge n mit Kontrollpolynom $h(x) = \sum_{j \le k} h_j x^j$ hat das Generatorpolynom $g^\perp(x) = \bar{h}(x) = x^k \cdot h\left(x^{-1}\right)$. Zwecks Normierung ist durch $h_0 = h(0)$ zu dividieren.* □

Hinter den Generator- und Kontrollmatrizen linearer Codes $\mathcal{C} \le F^n$ verstecken sich lineare Abbildungen $F^k \xrightarrow{g} F^n \xrightarrow{h} F^{n-k}$ mit $\operatorname{img} g = \mathcal{C} = \ker h$, die aber nicht eindeutig bestimmt sind (sie können vorne bzw. hinten mit Isomorphismen verknüpft werden). Im vorliegenden Fall gibt es außer der durch die Kontrollmatrizen $\bar{H}$ bzw. H spezifizierten linearen "Kontrollabbildungen" $F^n \longrightarrow F^{n-k}$ noch eine weitere kanonische derartige Abbildung, nämlich $\_ \bmod g(x)$, siehe Satz 6.2.01.

6.3.03 Definition. Unter dem *Syndrompolynom* eines empfangenen Worts $\boldsymbol{y}$ bzgl. eines zyklischen Codes mit Generatorpolynom $g(x)$ verstehen wir den Rest $s(x)$ bei Division von $y(x)$ durch $g(x)$, bzw. den Rest $\bar{s}(x)$ bei Division von $\bar{y}(x)$ durch $g(x)$.

Zur Decodierung der systematischen Codierung mittels Polynomdivision durch $g(x)$ ist diese Form des Syndroms besonders nützlich.

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

Fehler	Syndrom		Fehler	Syndrom	
0	0	bzw. 000	x^3	$x+1$	bzw. 011
1	1	bzw. 001	x^4	x^2+x	bzw. 110
x	x	bzw. 010	x^5	x^2+x+1	bzw. 111
x^2	x^2	bzw. 100	x^6	x^2+1	bzw. 101

Empfangen wir etwa das Wort $\boldsymbol{y} = 1001101$, so gilt $\bar{y}(x) = x^6+x^3+x^2+1$. Der Divisionsalgorithmus

mittels der Schieberegistermaschine (2.3-14) liefert (mit der blauen Hilfslinie als Abbruchkriterium)

$$\begin{array}{rcccc|ccc}
 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\
\hline
 & 0 & 0 & 0 & & & & \\
-(& 1 & 0 & 1 & 1 &) & & \\
\hline
 & & 0 & 1 & 1 & & & \\
 & -(& 0 & 0 & 0 & 0 &) & \\
\hline
 & & & 1 & 1 & 0 & & \\
 & & -(& 1 & 0 & 1 & 1 &) \\
\hline
 & & & & 1 & 1 & 1 & \\
 & & & -(& 0 & 0 & 0 & 0 &) \\
\hline
 & & & & & 1 & 1 & 0
\end{array}$$

woraus wir $\boldsymbol{s} = 101 + 110 = 011$ ablesen. Korrektur des Koeffizienten von x^3 liefert $\boldsymbol{c} = 1000101$, was im systematischen Fall den Klartext $\boldsymbol{u} = 1000$ ergibt.

Das folgende Ergebnis liefert die Grundlage für eine weitere Vereinfachung der Standarddecodierung. Diese Verfahren werden ebenfalls auf Polynomdivision durch $g(x)$ beruhen, daher beschränken wir uns hier auf die fallende Polynomdarstellung.

6.3.05 Proposition. *Ist $g(x)$ ein echter Faktor von $x^n - 1$ und hat $\bar{y}(x) \in F[x]_{x^n-1}$ das Syndrompolynom $\bar{s}(x)$, so stimmen die Syndrompolynome der in $F[x]_{x^n-1}$ gebildeten Produkte $x \cdot \bar{y}(x)$ und $x \cdot \bar{s}(x)$ überein.*

Beweis: Wir wollen in $F[x]$ rechnen. Dort hat das Produkt $x \cdot \bar{y}(x) \bmod x^n - 1$ die Darstellung $x \cdot \bar{y}(x) - y_0\,(x^n - 1)$, während $x \cdot \bar{s}(x) \bmod x^n - 1$ wegen $\operatorname{grd} s(x) < \operatorname{grd} g(x) < n$ auch in $F[x]$ die Form $x \cdot s(x)$ hat. Mit $\bar{y}(x) = q(x)g(x) + \bar{s}(x)$ und $x^n - 1 = g(x)h(x)$ erhalten wir

$$\begin{aligned} x \cdot \bar{y}(x) - y_0\,(x^n - 1) &= x \cdot \bar{y}(x) - y_0 g(x)h(x) = x\big(\bar{s}(x) - q(x)g(x)\big) - y_0 \cdot g(x)h(x) \\ &= x \cdot \bar{s}(x) + g(x)\big(x \cdot q(x) - y_0 \cdot h(x)\big) \end{aligned}$$

Damit haben $x \cdot \bar{y}(x) - y_0\,(x^n - 1)$ und $x \cdot \bar{s}(x)$ bei der Division durch $g(x)$ denselben Rest. □

Läßt man in Beispiel 6.3.04 die Schieberegistermaschine einen Schritt weiterlaufen, so wird die Division durch $g(x)$ auf $x \cdot \bar{s}(x)$ angewendet, also erhalten wir das Syndrom von $x \cdot \bar{y}(x)$ was dem um eine Position zyklisch nach links verschobenen Wort $\boldsymbol{y}^{(1)}$ entspricht. Entsprechend können wir die Syndrome aller zyklischen Verschiebungen des Worts $\boldsymbol{y}$ berechnen. Insofern können wir bei der Korrektur eines Fehlers solange warten, bis er in der signifikantesten Position auftritt. Damit sind in der Syndromliste nur Fehlermuster von Interesse, deren (fallende) Polynome maximalen Grad haben. Diese Idee wird im folgenden Verfahren umgesetzt.

6.3.06 Algorithmus. *Meggitt-Decoder*: Aus der vollständigen Syndromliste aller Fehlermuster, deren Hamming-Gewicht durch $(d-1)/2$ beschränkt ist, und der dazugehörigen Syndrome benötigen wir nur diejenigen Einträge, bei denen das erste Fehlerbit von 0 verschieden ist (das zugehöriger Fehlerpolynom hat dann maximalen Grad). Wir sprechen dann von der *reduzierten*

Syndromliste. Anstelle des gesamten Fehlermusters genügt es, den führenden Koeffizienten zu speichern (über $\mathbb{Z}_2$ entfällt auch das). In Beispiel 6.3.04 verbleibt somit nur das Syndrompolynom x^2+1. Der Algorithmus besteht aus 4 Schritten:

(0) Die Schieberegistermaschine zur Polynomdivision durch $g(x)$ verarbeitet in k Takten die ersten k Komponenten des empfangenen Worts $\boldsymbol{y} \in F^n$; die Summe aus Schieberegisterinhalt und den letzten r Komponenten des Worts $\boldsymbol{y}$ liefert dann das Syndrompolynom $\bar{s}^{(0)}(x)$ des unverschobenen Worts $\boldsymbol{y} = \boldsymbol{y}^{(0)}$.

(1) In weiteren $n-1$ Takten, unter Eingabe der letzten $n-k$ Komponenten des Worts $\boldsymbol{y}$ und weiterer $k-1$ Nullen, erhalten wir nach entsprechender Summenbildung die Syndrompolynome $\bar{s}^{(i)}(x)$ der ggf. korrigierten Verschiebungen von $\boldsymbol{y}^{(i)}$, $0 < i < n$, und schließlich $\boldsymbol{y}^{(n)} := \boldsymbol{c}$.

(2) Sobald eines der aufgelisteten Syndrompolynome erscheint, ist das signifikanteste Bit (links) der aktuellen Verschiebung $\boldsymbol{y}^{(i)}$ des Worts $\boldsymbol{y}$ korrigieren, also den Wert y_i. Zu diesem Zweck subtrahieren wir im *nächsten* Schritt den gespeicherten führenden Koeffizienten des zugehörigen Fehlermusters im letzten(!) Schieberegister, wo das signifikanteste Bit von $\boldsymbol{y}^{(i)}$ nun auftaucht. Dann fahren mit dem so korrigierten Register in der Rechnung fort. Treten in den Schieberegistern nur Nullen auf, sind keine weiteren Korrekturen nötig.

(3) Eine letzte Linksverschiebung, ggf. nach Korrektur der führenden Komponente, liefert das korrigierte empfangene Wort $\boldsymbol{c} = \boldsymbol{y}^{(n)}\boldsymbol{y} - \boldsymbol{e}$. Bei systematischer Codierung erhalten wir durch Entfernen der letzten r Kontrollbits das gesuchte Infowort $\boldsymbol{u} \in F^k$, während bei direkter Codierung eine Polynomdivision durch $g(x)$ zu erfolgen hat.

6.3.07 Beispiel. Fortsetzung von Beispiel 6.3.04:

reduziere Syndromliste:	Fehlermuster	Syndrom
	x^6 bzw. 1000000	x^2+1 bzw. 101

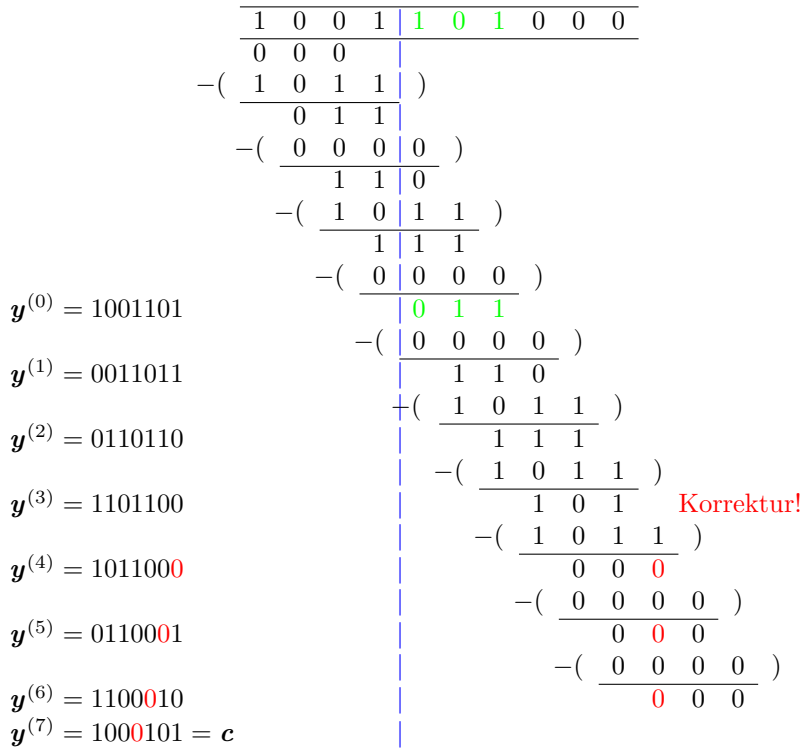
$$
\begin{array}{rl}
 & \begin{array}{cccc|cccccc} 1&0&0&1&1&0&1&0&0&0 \end{array} \\
 & \begin{array}{cccc|} 0&0&0& \end{array} \\
-(& \begin{array}{cccc|} 1&0&1&1 \end{array}) \\
 & \begin{array}{cccc|} &0&1&1 \end{array} \\
-(& \begin{array}{cccc|} 0&0&0&0 \end{array}) \\
 & \begin{array}{cccc|} &&1&1 \end{array} \\
-(& \begin{array}{cccc|} 1&0&1&1 \end{array}) \\
 & \begin{array}{cccc|} &&1&1 \end{array} \\
-(& \begin{array}{cccc|} 0&0&0&0 \end{array}) \\
\boldsymbol{y}^{(0)} = 1001101 & \begin{array}{cccc|} &&1&1 \end{array} \quad \Longrightarrow \bar{s}^{(0)}(x) = x+1 \\
-(& \begin{array}{cccc|} 0&0&0&0 \end{array}) \\
\boldsymbol{y}^{(1)} = 0011011 & \begin{array}{cccc|} &&1&0 \end{array} \quad \Longrightarrow \bar{s}^{(1)}(x) = x^2+x \\
+(& \begin{array}{cccc|} 1&0&1&1 \end{array}) \\
\boldsymbol{y}^{(2)} = 0110110 & \begin{array}{cccc|} &&0&1 \end{array} \quad \Longrightarrow \bar{s}^{(2)}(x) = x^2+x+1 \\
-(& \begin{array}{cccc|} 1&0&1&1 \end{array}) \\
\boldsymbol{y}^{(3)} = 1101100 & \begin{array}{cccc|} &&1&0 \end{array} \quad \Longrightarrow \bar{s}^{(3)}(x) = x^2+1, \text{ Korrektur!} \\
-(& \begin{array}{cccc|} 1&0&1&1 \end{array}) \\
\boldsymbol{y}^{(4)} = 1011000 & \begin{array}{cccc|} &&0&0 \end{array} \quad \Longrightarrow \bar{s}^{(4)}(x) = 0 \\
-(& \begin{array}{cccc|} 0&0&0&0 \end{array}) \\
\boldsymbol{y}^{(5)} = 0110001 & \begin{array}{cccc|} &&0&0 \end{array} \quad \Longrightarrow \bar{s}^{(5)}(x) = 0 \\
-(& \begin{array}{cccc|} 0&0&0&0 \end{array}) \\
\boldsymbol{y}^{(6)} = 1100010 & \begin{array}{cccc|} &&0&0 \end{array} \quad \Longrightarrow \bar{s}^{(6)}(x) = 0 \\
\boldsymbol{y}^{(7)} = 1000101 = \boldsymbol{c} &
\end{array}
$$

Die Syndrompolynome sind so nummeriert, daß die zu korrigierende Komponente direkt ablesbar ist. In diesem Fall ist es y_3, und wir decodieren $\boldsymbol{y} = 1001101$ systematisch zu $\boldsymbol{u} = 1000$.

Hier werden die letzten r Bits mehrfach verwendet: bei der Berechnung von $\bar{s}^{(0)}(x)$ sowie in den nächsten r Takten der SRM. Schreibt man aber $\bar{s}^{(0)}(x)$ direkt in die Schieberegister, kann man die letzten r Eingabe-Bits anschließend ignorieren. Dazu könnte man sämtliche Eingaben in r Pufferregistern verzögern, um nach r Takten Vorlauf und den ersten k Takten der Maschine die letzten r Eingabe-Bits aus den Pufferregistern simultan zu den Schieberegistern zu addieren. Oder man könnte nach den ersten k Takten in r eingeschobenen Takten die jeweilige Eingabe zum linken Schieberegister addieren und die Schieberegister zyklisch nach links verschieben.

6.3.08 Beispiel. Fortsetzung von Beispiel 6.3.07: Schreibt man das Syndrompolynom $\bar{s}^{(0)}(x)$

direkt in die Schieberegister (in grün), ergibt sich die folgende Modifikation der obigen Berechnung:



Wieder decodieren wir $\boldsymbol{y} = 1001101$ systematisch zu $\boldsymbol{u} = 1000$.

Der nächste Decoder dient vornehmlich der Korrektur von *Burst-Fehlern*. Diese treten lokalisiert in Gruppen auf, im Gegensatz zum statistisch gleichmäßig verteilten "weißen Rauschen". Die Methode kann neben Einzelfehlern in manchen Fällen auch Doppelfehler korrigieren.

6.3.09 Definition. $\boldsymbol{e} \in F^n$ mit $\omega(\boldsymbol{e}) \leq l$ heißt *zyklisches Fehlerbündel* der Länge l, falls die Komponenten $\neq 0$ innerhalb von l zyklisch aufeinanderfolgenden Positionen auftreten.

Einzelfehler sind immer Bündel der Länge 1. Treten t Fehler in einem Codewort der Länge n auf, so ist die Existenz von k fehlerfreien Positionen äquivalent dazu, daß alle Fehler in einem Bündel der Länge $r = n-k$ auftreten. Damit dies bei jeder Verteilung der t Fehler funktioniert, muß $k < n/t$ bzw. $t < n/k$ gelten. Dies ist die Voraussetzung für die Anwendbarkeit von Algorithmus 6.3.11 zur Burstfehlerkorrektur bei der Korrektur unkorrelierter Fehler.

6.3.10 Proposition. *Der zyklische $[n,k]$-Code $\mathcal{C}$ über F möge maximal t Fehler korrigieren. Falls im empfangenen Wort $\boldsymbol{y}$ höchstens t Fehler aufgetreten sind, so gilt:*

(0) *aus $\omega(\boldsymbol{s}) \leq t$ folgt $\boldsymbol{s} = \boldsymbol{e}$;*

(1) *ist der Fehler in einem Bündel der Länge $n-k$ lokalisiert, so existiert eine zyklische Vertauschung $\boldsymbol{y}^{(i)}$ des Worts $\boldsymbol{y}$ mit $\omega\big(\boldsymbol{s}^{(i)}\big) \leq t$ (d.h., der Fehler läßt sich "einfangen").*

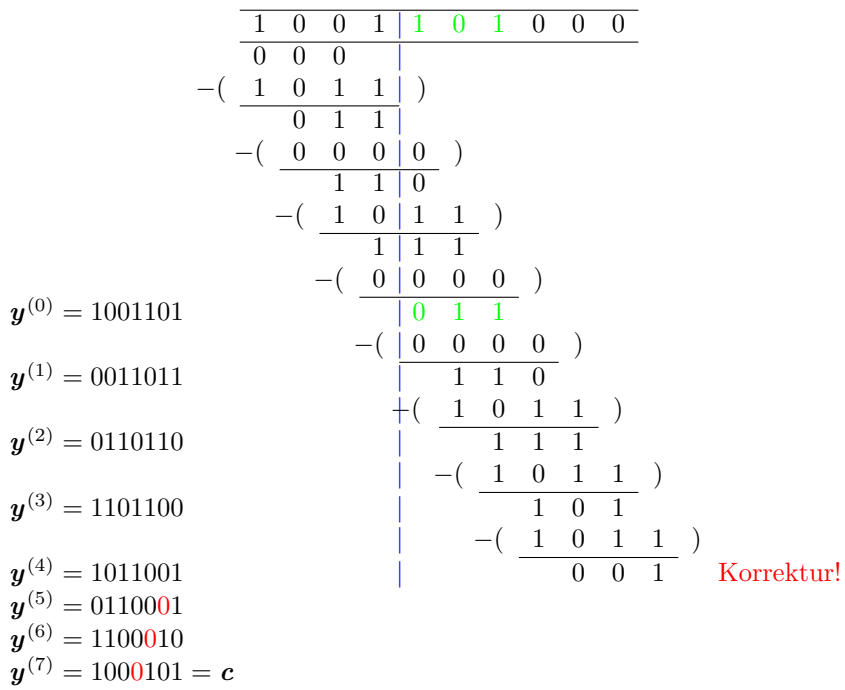
Beweis:

(0) Der Fehler $\boldsymbol{e} = \boldsymbol{y} - \boldsymbol{c}$ mit $\omega(e) \le t$ und $\boldsymbol{y}$ haben dasselbe Syndrom $\boldsymbol{s}$. Gilt $\omega(\boldsymbol{s}) \le t$, so folgt aus $\bar{e}(x) = q(x) \cdot g(x) + \bar{s}(x)$, daß das zu $q(x) \cdot g(x)$ gehörige Codewort höchstens ein Hamming-Gewicht von $2t$ haben kann und somit mit $\boldsymbol{0}$ übereinstimmt.

(1) Ist $\boldsymbol{e}$ ein Fehlerbündel der Länge $n-k$, können wir $\boldsymbol{y}$ zyklisch so verschieben, daß alle Fehler in den letzten $n-k$ Komponenten von $\boldsymbol{y}^{(i)}$ auftreten. Der entsprechende Fehler $\boldsymbol{e}^{(i)}$ hat dasselbe Syndrom wie $\boldsymbol{y}^{(i)}$, und zwar $\boldsymbol{e}^{(i)}$ wegen $\operatorname{grd} \bar{e}^{(i)}(x) < n-k$. Nach Voraussetzung folgt nun $\omega\big(\boldsymbol{s}^{(i)}\big) \le t$. □

6.3.11 Algorithmus. Decodierung durch Fehlereinfangen (ohne reduzierte Syndromliste):
Wie beim Meggitt-Decoder werden die Syndrome $\boldsymbol{s}^{(i)}$, $i < n$, des empfangenen Worts $\boldsymbol{y}$ und seiner zyklischen Linksvertauschungen bis $\boldsymbol{y}^{(n)}$ berechnet. Sobald $\omega\big(\boldsymbol{s}^{(i)}\big) \le t$ gilt, subtrahieren wir $\boldsymbol{s}^{(i)}$ rechtsbündig von $\boldsymbol{y}^{(i)}$ und haben damit alle Fehler gleichzeitig korrigiert. Die Division kann dann abgebrochen werden, aber $\boldsymbol{c} = \boldsymbol{y}^{(n)}$ ist noch zu bestimmen.

6.3.12 Beispiel. Wird in Beispiel 6.3.08 auf die Korrektur von $\boldsymbol{y}^{(3)}$ verzichtet, so taucht nach 8 Takten erstmals ein Syndrom mit einem Hamminggewicht ≤ 1 auf, nämlich $\boldsymbol{s}^{(4)} = 001$. Da $\boldsymbol{y}^{(4)}$ mit y_3 endet, wird diese Komponente durch rechtsbündige Subtraktion von 001 korrigiert.



Der wesentliche Unterschied im Ablauf beider Verfahren tritt aber erst zutage, wenn mehr als ein Fehler korrigierbar ist. Der Meggitt-Decoder korrigiert jeden Fehler einzeln, während beim Fehlereinfangen alle Fehler simultan korrigiert werden.

6.3.13 Beispiel. Das Generatorpolynom $g(x) = x^8 + x^7 + x^6 + x^4 + 1$ erzeugt einen zyklischen $[15, 7]$-Code $\mathcal{C}$ über $\mathrm{GF}(2)$, der Doppelfehler korrigieren kann.

Es gibt 15 bzw. $\binom{15}{2} = 105$ Muster für Einzel- bzw. Doppelfehler. Damit hat die Syndromliste 120 Einträge. Aber nur 15 dieser Fehlermuster beginnen mit einer 1.

Fehlermuster	Syndrom
100000000000000	11101000
110000000000000	10011100
101000000000000	11010010
100100000000000	11110101
100010000000000	00001110
100001000000000	10011011
100000100000000	00111001

Fehlermuster	Syndrom
100000010000000	01101000
100000001000000	10101000
100000000100000	11001000
100000000010000	11111000
100000000001000	11100000
100000000000100	11101100
100000000000010	11101010
100000000000001	11101001

Der Meggitt-Decoder korrigiert jeden Fehler einzeln:

```
 1 0 0 0 1 0 0 | 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 | 0
-( 1 1 1 0 1 0 0 | 0 1 )
    1 1 0 1 0 0 | 0 1
  -( 1 1 1 0 1 0 | 0 0 1 )
      0 1 1 1 0 | 0 1 1
    -( 0 0 0 0 0 | 0 0 0 0 )
        1 1 1 0 | 0 1 1 0
      -( 1 1 1 0 | 1 0 0 0 1 )
          0 0 0 | 1 1 1 0 1
        -( 1 1 1 | 0 1 0 0 0 1 )
            1 1 | 1 0 1 0 1 1
          -( 1 1 | 1 0 1 0 0 0 1 )
              0 | 0 0 0 0 1 1 1
            -( 0 | 0 0 0 0 0 0 0 0 )
               | 0 0 0 0 1 1 1 0   Korrektur von y_0
             -( | 0 0 0 0 0 0 0 0 0 )
               |   0 0 0 1 1 1 0 1
              +( 0 0 0 0 0 0 0 0 0 )
               |     0 0 1 1 1 0 1 0
               |   -( 0 0 0 0 0 0 0 0 0 )
               |       0 1 1 1 0 1 0 0
               |     -( 0 0 0 0 0 0 0 0 0 )
               |         1 1 1 0 1 0 0 0   Korrektur von y_4
               |       -( 1 1 1 0 1 0 0 0 1 )
               |           0 0 0 0 0 0 0 0   Rest OK
```

Wegen $2 < \lceil 15/7 \rceil = 3$ funktioniert auch die Korrektur durch Fehlereinfangen:

$$
\begin{array}{rcl}
1\;0\;0\;0\;1\;0\;0 & | & 0\;0\;0\;0\;0\;0\;0\;0 \\
\hline
0\;0\;0\;0\;0\;0\;0 & | & 0 \\
-(\;1\;1\;1\;0\;1\;0\;0 & | & 0\;1\;\;) \\
\hline
\quad 1\;1\;0\;1\;0\;0 & | & 0\;1 \\
-(\;1\;1\;1\;0\;1\;0 & | & 0\;0\;1\;\;) \\
\hline
\quad 0\;1\;1\;1\;0 & | & 0\;1\;1 \\
-(\;0\;0\;0\;0\;0 & | & 0\;0\;0\;0\;\;) \\
\hline
\quad 1\;1\;1\;0 & | & 0\;1\;1\;0 \\
-(\;1\;1\;1\;0 & | & 1\;0\;0\;0\;1\;\;) \\
\hline
\quad 0\;0\;0 & | & 1\;1\;1\;0\;1 \\
-(\;1\;1\;1 & | & 0\;1\;0\;0\;0\;1\;\;) \\
\hline
\quad 1\;1 & | & 1\;0\;1\;0\;1\;1 \\
-(\;1\;1 & | & 1\;0\;1\;0\;0\;0\;1\;\;) \\
\hline
\quad 0 & | & 0\;0\;0\;0\;1\;1\;1 \\
-(\;0 & | & 0\;0\;0\;0\;0\;0\;0\;0\;\;) \\
\hline
 & | & 0\;0\;0\;0\;1\;1\;1\;0 \qquad \omega(\bar{s}^{(0)}) = 3 \\
-(\; & | & 0\;0\;0\;0\;0\;0\;0\;0\;0\;\;) \\
\hline
 & | & \quad 0\;0\;0\;1\;1\;1\;0\;0 \qquad \omega(\bar{s}^{(1)}) = 3 \\
+(\; & | & 0\;0\;0\;0\;0\;0\;0\;0\;0\;\;) \\
\hline
 & | & \quad\;\; 0\;0\;1\;1\;1\;0\;0\;0 \qquad \omega(\bar{s}^{(2)}) = 3 \\
 & | & -(\;0\;0\;0\;0\;0\;0\;0\;0\;0\;\;) \\
\hline
 & | & \quad\;\;\; 0\;1\;1\;1\;0\;0\;0\;0 \qquad \omega(\bar{s}^{(3)}) = 3 \\
 & | & -(\;0\;0\;0\;0\;0\;0\;0\;0\;0\;\;) \\
\hline
 & | & \quad\;\;\;\; 1\;1\;1\;0\;0\;0\;0\;0 \qquad \omega(\bar{s}^{(4)}) = 3 \\
 & | & -(\;1\;1\;1\;0\;1\;0\;0\;0\;1\;\;) \\
\hline
 & | & \quad\;\;\;\;\; 0\;0\;0\;1\;0\;0\;0\;1 \qquad \omega(\bar{s}^{(5)}) = 2
\end{array}
$$

Zu korrigieren ist nun die aktuelle Verschiebung $\boldsymbol{y}^{(5)} = 000000000010001$ durch rechtsbündige Addition des Syndroms $\bar{s}^{(5)}$, was das Nullwort liefert.

6.3.14 Bemerkung. Abgesehen von der Tatsache, daß Syndrome für Doppelfehler berechnet, aber nur zur Korrektur des jeweils ersten dieser Fehler verwendet werden, fallen beim obigen Meggitt-Decoder weitere Besonderheiten auf.

- Die Fehlermuster 100000010000000 und 100000001000000 (wie auch andere) lassen sich zyklisch ineinander überführen. Insofern sollte eines der zugehörigen Syndrome überflüssig sein. Entfernt man aber $\boldsymbol{y} = 100000001000000$ mit seinem Syndrom $\boldsymbol{s} = 10101000$ aus der Liste, so findet man bei der Decodierung des Worts $\boldsymbol{y}$ erstmals bei $\boldsymbol{y}^{(8)}$ einen Listeneintrag, der zur Korrektur von $y_8 = 1$ führt. Werden nur die zyklischen Vertauschungen $\boldsymbol{y}^{(i)}$, $i < 15$, betrachtet, wird die Komponente y_0 nie korrigiert. Um das zu vermeiden, müssen wir 15 weitere zyklische Vertauschungen vornehmen, also i bis 29 laufen lassen, damit die Korrektur von y_0 in der zweiten Runde erfolgen kann. Der Preis für die weiter reduzierte Syndromliste (in diesem Fall mit 8 Einträgen) besteht darin, bei jeder Decodierung $t{\cdot}n$ zyklische Vertauschungen des empfangenen Worts betrachten zu müssen. Dieser trade-off zwischen Speicherplatz und Rechenzeit erscheint wenig sinnvoll zu sein.

- Mit dem Syndrom 11101000 für den Einzelfehler 100000000000000 kennen wir automatisch die Syndrome für alle Doppelfehler, deren erster Fehler in Komponente 0 und deren zweiter Fehler in den letzten $n-k = 15-7 = 8$ Komponenten vorkommt: man braucht den entsprechenden Fehler nur zu 11101000 zu addieren. All diese Fälle sind im zweiten Teil der Syndromliste zusammengefaßt. Zur Decodierung reicht schon der erste Teil der Liste mit 7 Einträgen: Bei jedem berechneten Syndrom testen wir, ob es bei einem Doppelfehlermuster auftritt, oder sich um maximal eine Position von einem Einzelfehlersyndrom unterscheidet. Auf diese Weise läßt sich tatsächlich Speicherplatz einsparen, ohne Rechenzeit zu opfern. Allerdings bleibt zu klären, wie groß die reduzierte Syndromliste für einen allgemeinen $[n,k,2t+1]$-Code über $\mathrm{GF}(q)$ sein muß.

6.3.15 Beispiel. Der nach Beispiel 3.2.05 erwähnte binäre Golay-Code ist ein zyklischer, perfekter, 3 Fehler korrigierender $[23,12,7]$-Binärcode mit Generatorpolynom

$$g(x) = x^{11}+x^{10}+x^6+x^5+x^4+x^2+1 \qquad \text{oder} \qquad g(x) = x^{11}+x^9+x^7+x^6+x^5+x+1$$

Treten in $\boldsymbol{y}$ bis zu drei Fehler in einem Bündel der Länge ≤ 11 auf, funktioniert die Decodierung durch Fehlereinfangen, d.h., eine zyklische Vertauschung des Worts $\boldsymbol{y}$ liefert ein Syndrom der Länge 11 mit Hamminggewicht ≤ 3, welches dem entsprechend vertauschten Fehlermuster entspricht (alle Fehler in den letzten 11 Positionen).

Allerdings sind nicht alle Dreifach- oder Doppelfehler auf Bündel der Länge 11 beschränkt. Die folgende Deocodierungsmethode basiert auf der Beobachtung, daß in diesem Fall immer eine zyklische Vertauschung möglich ist, so daß nur ein Fehler außerhalb der letzten 11 Bits liegt.

Sind 2 oder 3 Fehler aufgetreten, die sich nicht in einem Bündel der Länge ≤ 11 lokalisieren lassen, kann nach Proposition 6.3.10 keines der Syndrome der zyklischen Vertauschungen ein Hamminggewicht ≤ 3 haben. Nun ändern wir y_0 und suchen nach Syndromen mit Hamminggewicht ≤ 2!

War y_0 falsch, so enthält das modifizierte Wort nur noch ≤ 2 Fehler in einem Bündel der Länge ≤ 11. Dies führt garantiert zu einem Syndrom mit Hamminggewicht ≤ 2, die verbliebenen Fehler können also korrigiert werden.

War y_0 korrekt, so liegen nun 3 oder 4 Fehler in einem Bündel der Länge ≥ 12 vor. Wegen des Minimalabstandes von 7 kann es kein Codewort mit Hammingabstand 2 zum modifizierten Wort geben. Folglich haben alle systematischen Syndrome des modifizierten Wortes ein Hamminggewicht ≥ 3. Stellen wir dies fest, so setzen wir y_0 wieder auf den ursprünglichen Wert und verändern y_1. Spätestens die Änderung der Komponente y_{11} korrigiert einen Fehler, so daß die verbleibenden Fehler in einem Bündel der Länge ≤ 11 lokalisiert sind.

6.4 Faktorisierung des Generatorpolynoms

Ziel ist es nun, die Wurzeln des Generatorpolynoms eines zyklischen Codes nutzbringend einzusetzen, etwa zur schnellen Konstruktion einer Kontrollmatrix. Im folgenden Abschnitt 6.5 werden wir einen Zusammenhang mit den BCH-Codes aus Abschnitt 4.6 herstellen.

Die möglichen Generatorpolynome waren in Satz 6.2.05 als Teiler der Polynome $x^n - 1$ charakterisiert worden. Bei ihrem ersten Auftreten im Rahmen konventioneller Reed-Solomon Codes (4.2-29) waren Generatorpolynome aber als Produkte bestimmter Linearfaktoren eingeführt worden. Diesen Zusammenhang gilt es zunächst zu untersuchen.

Zuvor bedarf es aber einer Charakterisierung derjenigen Polynome $a(x) \in F[x]$, die in keiner algebraischen Erweiterung $\bar{F}$ des Körpers F mehrfache Wurzeln haben.

6.4.01 Lemma. *$a(x) \in F[x]$ hat genau dann in jeder AKE $\tilde{F}$ von $F = \mathrm{GF}(q)$ nur einfache Wurzeln, wenn $a(x)$ zu seiner formalen Ableitung (vergleiche Definition 5.4.05) teilerfremd ist, also $\mathrm{ggT}\,(a(x), a'(x)) = 1$ gilt.*

Beweis: Falls $a(x)$ in einer AKE $\tilde{F}$ von F die m-fache Wurzel β hat, $0 < m$, so können wir $a(x)$ in $\tilde{F}[x]$ umschreiben als

$$a(x) = (x - \beta)^m b(x) \tag{6.4-43}$$

wobei $b(\beta) \neq 0$ gilt. Die formale Ableitung ergibt sich nun zu

$$a'(x) = m(x - \beta)^{m-1} b(x) + (x - \beta)^m b'(x) \tag{6.4-44}$$

Folglich ist $r(x) := \mathrm{ggT}\,(a(x), a'(x))$ genau dann durch $x - \beta$ teilbar, wenn $m > 1$ gilt. Speziell impliziert $\mathrm{grd}\, r(x) = 0$ sofort $m = 1$.

Umgekehrt folgt aus $\mathrm{grd}\, r(x) > 0$ die Existenz einer Wurzel β von $r(x)$ in einer AKE $\tilde{F}$ von F. Dann teilt $x - \beta$ speziell $a(x)$, das wir gemäß (6.4-43) darstellen. Im Fall $m = 1$ müßte laut (6.4-44) gelten $a'(\beta) = m \cdot b(\beta) = 0$, was der Bedingung $b(\beta) \neq 0$ widerspricht. □

6.4.02 Definition. *$a(x)$ sei ein Polynom über $F = \mathrm{GF}(q)$.*

(a) Unter dem *Zerfällungskörper* von $a(x)$ verstehen wir die kleinste AKE $\tilde{F}$ von F, so daß $a(x)$ in $\tilde{F}[x]$ vollständig in Linearfaktoren zerfällt.

(b) Ist $a(x)$ zu x teilerfremd, d.h., $\mathrm{ggT}\,(a(x), x) = 1$, so bezeichnet der *Exponeten* $\exp a(x)$ die kleinste positive Zahl e, so daß $x^e - 1$ von $a(x)$ geteilt wird. Existiert keine solche Zahl, so setzen wir $\exp a(x) = \infty$.

Der Exponent von $a(x)$ stimmt mit der multiplikativen Ordnung von x im Faktorring $F[x]_{a(x)}$ überein, denn $a(x) \,|\, (x^e - 1)$ ist zu $(x^e - 1) \bmod a(x) = 0$ bzw. zu $x^e \bmod a(x) = 1$ äquivalent.

6.4.03 Proposition. *Für ein zu x teilerfremdes Polynom $a(x)$ über $F = \mathrm{GF}(q)$ gilt*

(0) *$a(x) \,|\, (x^\ell - 1)$ genau dann, wenn $\exp a(x) \,|\, \ell$.*

Ist darüberhinaus $e := \exp a(x)$ teilerfremd zu q, so

(1) *sind die Wurzeln von $a(x)$ in jeder AKE von F einfach, und*

(2) *der Zerfällungskörper von* $a(x)$ *ist* $\mathrm{GF}(q^m)$, *wobei* m *die kleinste positive Zahl mit* $e \,|\, (q^m - 1)$ *ist.*

Beweis:

(0) Beide Aussagen sind äquivalent zu $x^\ell \bmod a(x) = 1$.

(1) Da $a(x)$ Teiler von $x^e - 1$ ist, genügt es nachzuweisen, daß letzteres Polynom in jeder AKE von F nur einfache Wurzeln hat. Um Lemma 6.4.01 anwenden zu können, berechnen wir $(x^e - 1)' = e \cdot x^{e-1}$, was wegen $\mathrm{ggT}(e, q) = 1$ von 0 verschieden und offensichtlich zu $x^e - 1$ teilerfremd ist.

(2) $a(x)$ zerfällt über einer AKE $\mathrm{GF}(q^m)$ von F genau dann vollständig in verschiedene Linearfaktoren, wenn $a(x)$ Teiler des Produkts *aller* Linearfaktoren über $\mathrm{GF}(q^m)$ ist. Aber dieses Produkt ist genau $x\left(x^{q^m-1} - 1\right)$, denn die Ordnung $q^m - 1$ der multiplikativen Gruppe von $\mathrm{GF}(q^m)$ wird von der Ordnung jedes ihrer Elemente geteilt, das somit Wurzel von $x^{q^m-1} - 1$ ist; und 0 ist trivialerweise eine Wurzel. Da $a(x)$ und x nach Voraussetzung teilerfremd sind, ist obige Bedingung auch zu $a(x) \,|\, x^{q^m-1} - 1$ äquivalent, und gemäß Teil (0) auch zu $e \,|\, q^m - 1$. □

Als Teiler von $x^n - 1$ ist ein Generatorpolynom $g(x)$ eines zyklischen $[n, k]$-Codes $\mathcal{C}$ über $F = \mathrm{GF}(q)$ teilerfremd zu x, nach Proposition 6.4.03 muß also $e := \exp g(x)$ die Codelänge n teilen. Handelt es sich um einen echten Teiler, ist $x^{\exp g(x)} - 1$, bzw. der zugehörige Vektor aus F^n als Vielfaches von $g(x)$ ein Codewort mit Hamming-Gewicht 2, was $d \le 2$ impliziert und insofern wenig interessant ist. Daher setzen wir künftig $\exp g(x) = n$ voraus. Zudem wollen wir fordern, daß die Codelänge n zur Körperbasis q teilerfremd ist, d.h., $\mathrm{ggT}(n, q) = 1$. Insbesondere gilt dann auch $\mathrm{ggT}(e, q) = 1$, weshalb Proposition 6.4.03 greift: der Zerfällungskörper $\mathrm{GF}(q^m)$ von $g(x)$ kann bestimmt werden sowie die dortige Zerlegung von $g(x)$ in Linearfaktoren.

Wie in Proposition 2.4.10(1) gezeigt, zerfällt die Wurzelmenge von $g(x)$ in einige $\sim_F$-Klassen, etwa

$$\{\, \beta_i \,|\, i < n - k \,\} = \sum_{j<t} C_j$$

Wähle Repräsentanten $\gamma_j \in C_j$. Nach Lemma 2.4.09(2) ist der Grad des Minimalpolynoms $M_{\gamma_j}(x)$, $j < t$, durch m beschränkt. Damit gilt

$$n - k = \mathrm{grd}\, g(x) = \sum_{i<t} \mathrm{grd}\, M_{\gamma_j}(x) \le t \cdot m \tag{6.4-45}$$

Mittels der gewählten Repräsentanten erhalten wir nun in Analogie zu Abschnitt 4.2 eine $n \times t$-Kontrollmatrix für $\mathcal{C}$ über $\tilde{F}$, vergl. 4.2-28:

$$\boldsymbol{c} \in \mathcal{C} \iff \boldsymbol{c} \cdot \bar{H} = \boldsymbol{0} \iff (\, \bar{c}\,(\gamma_j) : j < t \,) = \boldsymbol{0} \tag{6.4-46}$$

wobei wir die fallende Polynomschreibweise bevorzugen, also

$$\bar{H} = \begin{pmatrix} \gamma_0^{n-1} & \gamma_1^{n-1} & \dots & \gamma_{t-1}^{n-1} \\ \vdots & \vdots & \vdots & \vdots \\ \gamma_0^2 & \gamma_1^2 & \dots & \gamma_{t-1}^2 \\ \gamma_0 & \gamma_1 & \dots & \gamma_{t-1} \\ 1 & 1 & \dots & 1 \end{pmatrix}$$

Dies liefert eine $n \times t \cdot m$-Kontrollmatrix über $F = \mathrm{GF}(q)$, wenn man die Einträge als Zeilenvektoren aus $\mathrm{GF}(q^m)$ repräsentiert.

6.4.04 Beispiel. Über $F = \mathrm{GF}(2)$ betrachten wir einen zyklischen Code $\mathcal{C}$ der Länge $n = 7$ mit Generatorpolynom $g(x) = x^4 + x^3 + x^2 + 1 = (x-1)(x^3 + x + 1)$, einem Teiler von $x^7 - 1$ (vergl. Beispiel 6.2.06). $\exp g(x)$ stimmt als Teiler von 7 mit 7 überein. Somit ist der Zerfällungskörper von $g(x)$ durch $\mathrm{GF}(2^3)$ gegeben, denn 3 ist die kleinste positive Zahl m mit $7 \mid 2^m - 1$.

Wie in Beispiel 2.4.05(1) gesehen, kann $\mathrm{GF}(2^3)$ mit Hilfe der irreduziblen Polynome $x^3 + x + 1$ sowie $x^3 + x^2 + 1$ realisiert werden, mittels formaler Wurzeln α bzw. β. Offenbar gilt dann $M_\alpha(x) = x^3 + x + 1$ und $M_\beta(x) = x^3 + x^2 + 1$.

Im ersten Fall zerfällt die Wurzelmenge von $g(x)$ in zwei Konjugationsklassen

$$\{1\} + \{\alpha, \alpha^2, \alpha^4\}$$

Die Repräsentanten 1 und α liefern folgende Kontrollmatrizen über $\mathrm{GF}(2^3)$ bzw. $\mathrm{GF}(2)$:

$$\begin{pmatrix} 1 & \alpha^6 \\ 1 & \alpha^5 \\ 1 & \alpha^4 \\ 1 & \alpha^3 \\ 1 & \alpha^2 \\ 1 & \alpha \\ 1 & 1 \end{pmatrix} \quad \text{bzw.} \quad \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Die ersten beiden Spalten der binären Kontrollmatrix sind offenbar irrelevant. Entfernt man sie, bleibt eine Matrix vom Spaltenrang $4 = n - k = \mathrm{grd}\, g(x)$ übrig. Die Wahl eines anderen Repräsentanten für die zweite Konjugationsklasse, etwa α^2, resultiert in einer bestimmten Vertauschung der Zeilen(!) der Kontrollmatrizen

$$\begin{pmatrix} 1 & \left(\alpha^2\right)^6 \\ 1 & \left(\alpha^2\right)^5 \\ 1 & \left(\alpha^2\right)^4 \\ 1 & \left(\alpha^2\right)^3 \\ 1 & \left(\alpha^2\right)^2 \\ 1 & \alpha^2 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & \alpha^5 \\ 1 & \alpha^3 \\ 1 & \alpha \\ 1 & \alpha^6 \\ 1 & \alpha^4 \\ 1 & \alpha^2 \\ 1 & 1 \end{pmatrix} \quad \text{bzw.} \quad \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

Da die binären Matrizen Kontrollmatrizen desselben Codes sind, müssen sie durch elementare Spaltenoperationen auseinander hervorgehen.

Die dritte Möglichkeit der Representantenwahl liefert schließlich

$$\begin{pmatrix} 1 & (\alpha^4)^6 \\ 1 & (\alpha^4)^5 \\ 1 & (\alpha^4)^4 \\ 1 & (\alpha^4)^3 \\ 1 & (\alpha^4)^2 \\ 1 & \alpha^4 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & \alpha^3 \\ 1 & \alpha^6 \\ 1 & \alpha^2 \\ 1 & \alpha^5 \\ 1 & \alpha \\ 1 & \alpha^4 \\ 1 & 1 \end{pmatrix} \quad \text{bzw.} \quad \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

Wir können aber auch die zweite Darstellung von $\mathrm{GF}(2^3)$ mit der formalen Wurzel β von x^3+x^2+1 verwenden. Im diesem Fall ist β^3 eine Wurzel von $g(x)$, die Konjugiertheits-Partition ist also gegeben durch

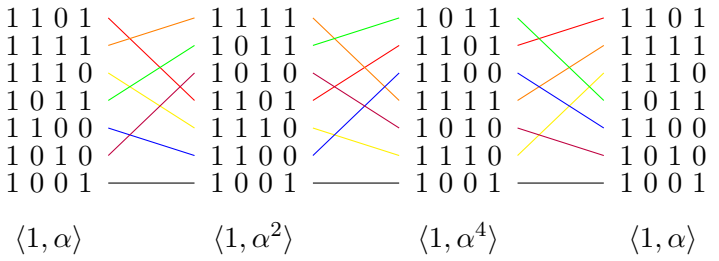
$$\{1\} + \{\beta^3, \beta^6, \beta^5\}$$

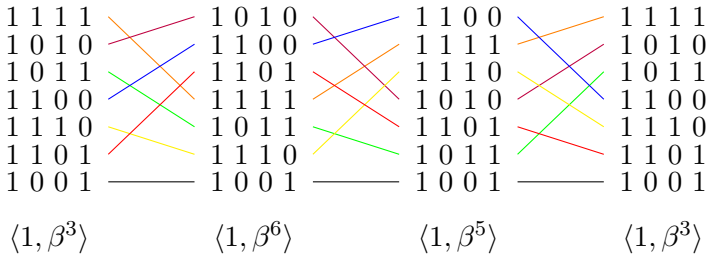
was drei weitere Kontrollmatrizen über $\mathrm{GF}(2^3)$ bzw. $\mathrm{GF}(2)$ liefert:

$$\begin{pmatrix} 1 & (\beta^3)^6 \\ 1 & (\beta^3)^5 \\ 1 & (\beta^3)^4 \\ 1 & (\beta^3)^3 \\ 1 & (\beta^3)^2 \\ 1 & \beta^3 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & \beta^4 \\ 1 & \beta \\ 1 & \beta^5 \\ 1 & \beta^2 \\ 1 & \beta^6 \\ 1 & \beta^3 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & (\beta^5)^6 \\ 1 & (\beta^5)^5 \\ 1 & (\beta^5)^4 \\ 1 & (\beta^5)^3 \\ 1 & (\beta^5)^2 \\ 1 & \beta^5 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & \beta^4 \\ 1 & \beta \\ 1 & \beta^5 \\ 1 & \beta^2 \\ 1 & \beta^6 \\ 1 & \beta^3 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & (\beta^6)^6 \\ 1 & (\beta^6)^5 \\ 1 & (\beta^6)^4 \\ 1 & (\beta^6)^3 \\ 1 & (\beta^6)^2 \\ 1 & \beta^6 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & \beta \\ 1 & \beta^2 \\ 1 & \beta^3 \\ 1 & \beta^4 \\ 1 & \beta^5 \\ 1 & \beta^6 \\ 1 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix} \quad \text{bzw.} \quad \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix} \quad \text{bzw.} \quad \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

Abschließend betrachten wir die Permutationen der Matrixzeilen bei den Ersetzungen der rewils zweiten Repräsentanten $\alpha \longrightarrow \alpha^2 \longrightarrow \alpha^4 \longrightarrow \alpha$ bzw. $\beta^3 \longrightarrow \beta^6 \longrightarrow \beta^5 \longrightarrow \beta^3$





Aus kombinatorischer Sicht stimmen diese beiden 3'er Zyklen überein. Kann es noch weitere Kontrollmatrizen geben, wenn man auf der fallenden Tupeldarstellung von α bzw. β beharrt, also die Permutation von Spalten ausschließt?

6.5 BCH-Codes aufgefaßt als zyklische Codes

Per Definition in Abschnitt 4.6 entstehen BCH-Codes der Länge n als Durchschnitte konventioneller Reed-Solomon-Codes der Länge n über einer AKE von F mit F^n. Unter der Voraussetzung $\mathrm{ggT}(n,q) = 1$ wollen wir speziell als AKE den Zerfällungskörper $\tilde{F} = \mathrm{GF}(q^m)$ von $x^n - 1$ betrachten. Dort wählen wir ein Element β der Ordnung n, Zahlen $D, b < n$, und eine Folge aufeinanderfolgender Wurzeln β^{b+l}, $l < D-1$. Diese bestimmen einen konventionellen Reed-Solomon-Code $\tilde{\mathcal{C}}$ mit dem Generatorpolynom $\tilde{g}(x) = \prod_{l<D-1}(x - \beta^{b+l}) \in \tilde{F}[x]$, der nach Beispiel 6.1.02 (c) zyklisch ist, wie auch der BCH-Code $\mathcal{C} = \tilde{\mathcal{C}} \cap F^n$. Dessen Generatorpolynom $g(x) \in F[x]$ hat mindestens die Wurzeln β^{b+l}, $l < D-1$, ist also in $\tilde{F}[x]$ durch $\tilde{g}(x)$ teilbar.

Das gradniedrigste derartige Polynom in $F[x]$ ist offenbar das Minimalpolynom $m(x)$ der Wurzelmenge β^{b+l}, $l < D-1$, das entsteht, indem $\tilde{g}(x)$ mit allen Linearfaktoren der Form $(x-\gamma)$ multipliziert wird, für die $\gamma \in \tilde{F}$ zu einer der Wurzeln von $\tilde{\mathcal{C}}$ konjugiert ist. Als Teiler von $g(x)$ ist $m(x)$ auch ein Teiler von $x^n - 1$, erzeugt also einen zyklischen Code, der nach Konstruktion in $\tilde{\mathcal{C}}$ enthalten ist. Also folgt $g(x) = m(x)$.

[Frage: Läßt sich jeder BCH-Code mit $\mathrm{ggT}(n,q) = 1$ auf diese Weise rekonstruieren?]

Die zu den fortlaufenden Potenzen β^{b+l}, $l < D-1$, konjugierten Wurzeln, die *nicht* zu dieser Folge gehören, heißen *überzählige Wurzeln* des Generatorpolynoms $g(x)$. Welchen Einfluß üben diese auf den Minimalabstand d aus? Je geringer ihre Anzahl, desto größer ist die Dimension k des BCH-Codes, genauer

$$\mathrm{grd}\, g(x) - (D-1) = (n-k) - (D-1) = (n-D+1) - k$$

Da die fortlaufende Wurzelfolge höchstens $D-1$ Konjugationsklassen nichtleer schneidet, erhalten wir

$$n - k \leq (D-1)m$$

denn die Größe einer Konjugationsklasse ist nach Lemma 2.4.09 durch m beschränkt,

6.5.01 Beispiel. Wir betrachten einen binären BCH-Code im engeren Sinne mit ungeradem designierten Minimalabstand, d.h., $q = 2$, $b = 1$ und $D = 2t + 1$. In diesem Fall verteilt sich die fortlaufende Wurzelfolge auf die Konjugationsklassen der ungeraden Potenzen von β und die obige Abschätzung verbessert sich zu

$$n - k \leq t \cdot m = \frac{D-1}{2} m$$

6.5.02 Beispiel. Im Falle eines binären BCH-Codes im engeren Sinne mit geradem designierten Minimalabstand $D = 2t$ werden ebenfalls die Konjugationsklassen der positiven geraden Potenzen von β von denen der ungeraden Potenzen abgedeckt; es verbleibt noch die einelementige Klasse $\{1\}$. Dies liefert die Abschätzung

$$n - k \leq 1 + (t-1)m = 1 + \left(\frac{D}{2} - 1\right) m$$

Wir haben auf diese Weise für BCH-Codes die Schranken aus den Beispielen 4.5.03 und 4.5.04 in Abschnitt 4.5 über allgemeine alternant Codes wiedergefunden, dabei aber nur die Zyklizität verwendet.

6.5.03 Beispiel. Über $F = \mathrm{GF}(2)$ geben wir die Parameter $n = 15$, $b = 1$ und $D = 7$ vor. Die kleinste Zahl m, so daß $2^m - 1$ durch $n = 15$ teilbar ist, berechnen wir zu $m = 4$. Der Zerfällungskörper von $x^{15} - 1$ ist somit $\tilde{F} = \mathrm{GF}(2^4)$. Wir können ihn bekanntermaßen mittels des irreduziblen Polynoms $x^4 + x + 1$ realisieren, vergl. Beispiel 4.5.02.

In diesem Fall hat die formale Wurzel ξ von x^4+x+1 tatsächlich die nötige Ordnung 15. Die fortlaufende Wurzelfolge $\xi, \xi^2, \xi^3, \xi^4, \xi^5, \xi^6$ verteilt sich wie folgt auf die Konjugationsklassen aller Potenzen von ξ, deren Minimalpolynome wir gleich mit angeben:

$$\begin{aligned}
C_\xi &= \{\underline{\xi}, \underline{\xi^2}, \underline{\xi^4}, \xi^8\} & M_\xi(x) &= x^4 + x + 1 \\
C_{\xi^3} &= \{\underline{\xi^3}, \underline{\xi^6}, \xi^{12}, \xi^9\} & M_{\xi^3}(x) &= x^4 + x^3 + x^2 + x + 1 \\
C_{\xi^5} &= \{\underline{\xi^5}, \xi^{10}\} & M_{\xi^5}(x) &= x^2 + x + 1 \\
C_{\xi^7} &= \{\xi^7, \xi^{14}, \xi^{13}, \xi^{11}\} & M_{\xi^7}(x) &= x^4 + x^3 + 1
\end{aligned}$$

Daraus ergibt sich das Generatorpolynom $g(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1$ des Grades 10 und des Hamming-Gewichts 7. Wegen $n - k = \operatorname{grd} g(x)$ erhalten wir somit einen $[15, 5, 7]$ BCH-Code $\mathcal{C}$. Die Schranke $\frac{D-1}{2} m = 12$ wird in diesem Fall nicht angenommen, dafür aber die Kugelpackungsschranke:

$$2^{n-k} \geq \binom{15}{0} + \binom{15}{1} + \binom{15}{2} + \binom{15}{2} = 576$$

erfordert $n - k \geq 10$.

Die übliche (15×3)-Kontrollmatrix über $\mathrm{GF}(2^4)$ liefert nach Ersetzung der Einträge durch Tupel in F^4 eine (15×12)-Kontrollmatrix über $F = \mathrm{GF}(2)$ mit Spaltenrang 10.

Insbesondere spannen die letzten 4 Spalten, die von ξ^5 herrühren, nur einen Raum der Dimension ≤ 2 auf.

In diesem Fall erhalten wir denselben Code, wenn wir mit der Wurzelfolge $\xi, \xi^2, \xi^3, \xi^4, \xi^5$ starten, also $D = 6$ anstelle von $D = 7$ verwenden. Das zeigt, daß der konventionelle Reed-Solomon-Code, der einen gegebenen BCH-Code erzeugt, nicht eindeutig bestimmt ist.

Gemäß Beispiel 4.5.02 (sieh auch Lemma 2.4.09) haben genau die Elemente der Konjugationsklassen $C_\xi = \{\xi, \xi^2, \xi^4, \xi^8\}$ und $C_{\xi^7} = \{\xi^7, \xi^{14}, \xi^{13}, \xi^{11}\}$ die Ordnung 15. Die zugehörigen Minimalpolynome sind insofern Spiegelbilder voneinander, als sich $x^4 + x^4 + 1$ als Spiegelbild von $x^1 1(x^4 + x + 1)$ ergibt. Insofern ist es nicht überraschend, daß die Verteilung der Wurzeln von $g(x)$ (blau markiert) in der Potenzenfolge der Elemente der Ordnung 15 innerhalb der Konjugationsklassen gleich und zwischen den Konjugationsklassen spiegelbildlich ist.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
ξ	1	ξ	ξ^2	ξ^3	ξ^4	ξ^5	ξ^6	ξ^7	ξ^8	ξ^9	ξ^{10}	ξ^{11}	ξ^{12}	ξ^{13}	ξ^{14}
ξ^2	1	ξ^2	ξ^4	ξ^6	ξ^8	ξ^{10}	ξ^{12}	ξ^{14}	ξ	ξ^3	ξ^5	ξ^7	ξ^9	ξ^{11}	ξ^{13}
ξ^4	1	ξ^4	ξ^8	ξ^{12}	ξ	ξ^5	ξ^9	ξ^{13}	ξ^2	ξ^6	ξ^{10}	ξ^{14}	ξ^3	ξ^7	ξ^{11}
ξ^8	1	ξ^8	ξ	ξ^9	ξ^2	ξ^{10}	ξ^3	ξ^{11}	ξ^4	ξ^{12}	ξ^5	ξ^{13}	ξ^6	ξ^{14}	ξ^7
ξ^7	1	ξ^7	ξ^{14}	ξ^6	ξ^{13}	ξ^5	ξ^{12}	ξ^4	ξ^{11}	ξ^3	ξ^{10}	ξ^2	ξ^9	ξ	ξ^8
ξ^{11}	1	ξ^{11}	ξ^7	ξ^3	ξ^{14}	ξ^{10}	ξ^6	ξ^2	ξ^{13}	ξ^9	ξ^5	ξ	ξ^{12}	ξ^8	ξ^4
ξ^{13}	1	ξ^{13}	ξ^{11}	ξ^9	ξ^7	ξ^5	ξ^3	ξ	ξ^{14}	ξ^{12}	ξ^{10}	ξ^8	ξ^4	ξ^2	ξ
ξ^{14}	1	ξ^{14}	ξ^{13}	ξ^{12}	ξ^{11}	ξ^{10}	ξ^9	ξ^8	ξ^7	ξ^6	ξ^5	ξ^4	ξ^3	ξ^2	ξ

Für jede Wahl eines Elements der Ordnung 15 liefert somit die maximale Folge aufeinanderfolgender Potenzen, die Wurzeln von $g(x)$ sind, einen konventionellen Reed-Solomon-Code, dessen Durchschnitt mit F^n den BCH-Code $\mathcal{C}$ erzeugt. [Frage: ist das immer so?]

6.5.04 Satz. (Die BCH-Schranke) *$\mathcal{C}$ sei ein zyklischer $[n, k, d]_q$-Code mit $\mathrm{ggT}(n, q) = 1$. Ist β ein Element der Ordnung n im Zerfällungskörper von $x^n - 1$ und ist β^{b+l}, $l < D - 1$, eine fortlaufende Wurzelfolge von $\mathcal{C}$ für Zahlen $b, D < n$ und $2 \leq D$, dann gilt $D \leq d$.*

Beweis: Da $\mathcal{C}$ mindestens die Wurzeln des durch die fortlaufende Wurzelfolge bestimmten BCH-Codes hat, ist $\mathcal{C}$ ein Unterraum des letzteren, und dessen Minimalabstand hat mindestens den Wert D. □

6.5.05 Beispiel. Wir betrachten einen binären zyklischen Code $\mathcal{C}$ mit den Parametern $n = 2^m - 1$ und $k = 2^m - m - 1$. Weil das Generatorpolynom $g(x)$ Teiler von $x^{2^m-1} - 1$ ist, sind seine Wurzeln auch Wurzeln von $x^{2^m-1} - 1$ und ihre Ordnung ist folglich ein Teiler von $n = 2^m - 1$.

Setzt man $\exp g(x) = 2^m - 1$ voraus, so haben alle Wurzeln von $g(x)$ die Ordnung $n = 2^m - 1$, sind also primitive Wurzeln der AKE $\mathrm{GF}(2^m)$. Ist $g(x)$ irreduzibel, so besteht

die Wurzelmenge aus genau einer Konjugationsklasse, und somit ist $g(x)$ das Minimalpolynom all seiner Wurzeln. Beide Bedingungen zusammen identifizieren $g(x)$ also als Minimalpolynom einer primitiven Wurzel, etwa α, von $\mathrm{GF}(2^m)$, (derartige Polynome heißen ebenfalls *primitiv*).

In der Aufzählung $\langle \alpha^r : r < 2^m - 1 \rangle$ ist die längste fortlaufende Wurzelfolge modulo $2^m - 1$ durch α, α^2 gegeben, denn α^{2^m-2} ist nicht konjugiert zu α (an dieser Stelle geht die Voraussetzung $q = 2$ ein). Damit ergeben sich für die BCH-Schranke die Parameter $b = 1$ und $D = 3$, woraus $d \geq 3$ folgt. In der Tat handelt es sich bei $\mathcal{C}$ um einen Hamming-Code mit $d = 3$.

6.5.06 Beispiele. Wie in Abschnitt 3.2 erwähnt, sind die beiden Golay-Codes die einzigen perfekten linearen Codes außer den Hamming-Codes und den binären Wiederholungscodes ungerader Länge.

(a) Im binären Fall, vergl. Beispiel 6.3.15, handelt es sich um einen $[23, 12, 7]$-Code. Die kleinste Zahl m mit $23 \mid 2^m - 1$ ist $m = 11$. Für ein Element α der Ordnung 23 in $\mathrm{GF}(2^{11})$ zerfallen die Potenzen wie folgt in Konjugationsklassen

$$\begin{aligned} C_\alpha &= \{\alpha, \alpha^2, \alpha^4, \alpha^8, \alpha^{16}, \alpha^9, \alpha^{18}, \alpha^{13}, \alpha^3, \alpha^6, \alpha^{12}\} \\ C_{\alpha^5} &= \{\alpha^5, \alpha^{10}, \alpha^{20}, \alpha^{17}, \alpha^{11}, \alpha^{22}, \alpha^{21}, \alpha^{19}, \alpha^{15}, \alpha^7, \alpha^{14}\} \\ C_1 &= \{1\} \end{aligned}$$

mit den Minimalpolynomen

$$\begin{aligned} M_\alpha(x) &= x^{11} + x^{10} + x^6 + x^5 + x^4 + x^2 + 1 \\ M_{\alpha^5}(x) &= x^{11} + x^9 + x^7 + x^6 + x^5 + x + 1 \\ M_1(x) &= x + 1 \end{aligned}$$

Die beiden Varianten des binären Golay-Codes, die von $M_\alpha(x)$ bzw. $M_{\alpha^5}(x)$ erzeugt werden, sind spiegelinvers zueinander, denn $M_{\alpha^5}(x) = x^{23}\left(x^{-12}M_\alpha\left(x^{-1}\right)\right) = \left(x^{11}M_\alpha\left(x^{-1}\right)\right.$.

Die längsten fortlaufenden Wurzelfolgen in C_α und C_{α^5} sind gegeben durch $\alpha, \alpha^2, \alpha^3, \alpha^4$ bzw. $\alpha^{19}, \alpha^{20}, \alpha^{21}, \alpha^{22}$. Dies liefert in beiden Fällen $D = 5$ und folglich $d \geq 5$.

(b) Im ternären Fall handelt es sich um einen $[11, 6, 5]$-Code über $\mathrm{GF}(3)$. Die kleinste Zahl m mit $11 \mid 3^m - 1$ bestimmen wir zu $m = 5$. Für ein Element α der Ordnung 11 in $\mathrm{GF}(3^6)$ zerfallen die Potenzen wie folgt in Konjugationsklassen

$$\begin{aligned} C_\alpha &= \{\alpha, \alpha^3, \alpha^9, \alpha^5, \alpha^4\} & M_\alpha(x) &= x^5 + x^4 - x^3 + x^2 - 1 \\ C_{\alpha^2} &= \{\alpha^2, \alpha^6, \alpha^7, \alpha^{10}, \alpha^8\} & M_{\alpha^2}(x) &= x^5 - x^3 + x^2 - x - 1 \\ C_1 &= \{1\} & M_1(x) &= x - 1 \end{aligned}$$

Wie zuvor sind auch die beiden Varianten des ternären Golay-Codes, die von $M_\alpha(x)$ bzw. $M_{\alpha^2}(x)$ erzeugt werden, spiegelinvers zueinander, da $M_{\alpha^2}(x) = -x^{11}\left(x^{-6}M_\alpha\left(x^{-1}\right)\right) = -x^5 M_\alpha\left(x^{-1}\right)$.

Die längsten fortaufenden Wurzelfolgen $\alpha^3, \alpha^4, \alpha^5$ in C_α und $\alpha^6, \alpha^7, \alpha^8$ in C_{α^2} haben in diesem Fall die Länge 3, was $D = 4$ und somit $d \geq 4$ impliziert.

7 Trellis- und Faltungscodes

In diesem Kapitel wird die Codierung als Abbildung stärker im Vordergrund stehen als ihre Bildmenge, der Code.

Bisher hatten wir uns mit Block-Codes beschäftigt, die der Codierung einzelner Klartextblöcke dienten. Die entsprechende Abbildung $F^k \longrightarrow F^n$ läßt sich immer kanonisch zu einem Monoid-Homomorphismus $(F^k)^* \longrightarrow (F^n)^*$ erweitern, mit dessen Hilfe auch längere Klartexte codierbar sind, sie müssen halt nur in mehrere Klartextblöcke aufgeteilt werden, die dann unabhängig voneinander, also *kontextfrei*, zu behandeln sind.

Diese Vorgehensweise hat den konzeptionellen Nachteil, daß für ein q-elementiges Alphabet F jeder Klartextblock $\boldsymbol{u} \in F^k$ der Länge $k \in \mathbb{N}$ immer gleich codiert wird, unabhängig davon, in welcher Position der Blockfolge er auftaucht. Das kann einem Angreifer ermöglichen, gezielt Blöcke in bestimmten Positionen zu manipulieren. Im Rahmen der Cryptographie begegnet man diesem Problem mit Hilfe sogenannter *Betriebsmodi*, bei denen bereits verschlüsselte Blöcke die Verschlüsselung späterer Blöcke beeinflussen. Etwas Ähnliches streben wir hier auch an.

Gelänge es uns, kanonische Isomorphismen zwischen $(F^m)^*$ und $(F^*)^m$ zu finden, $m \in \mathbb{N}$, so ergäbe sich sofort die Frage nach weiteren sinnvollen Codierungs-Abbildungen $(F^*)^k \longrightarrow (F^*)^n$ neben dem obigen Monoid-Homomorphismus, die evtl. ein besseres Fehlerkorrekturverhalten aufweisen könnten. Leider ist für unstrukturierte Mengen F das freie Monoid $(F^m)^*$ im Allgemeinen *nicht* zur Menge $(F^*)^m$ isomorph, denn letztere enthält auch m-Tupel von Wörtern unterschiedlicher Länge, und diese können mangels eines ausgezeichneten Elements in F nicht bis zur maximalen auftretenden Länge aufgefüllt werden.

Sofern es sich bei F aber um einen Körper handelt (ein Monoid mit neutralem Element 0 genügt bereits), können wir einerseits $(F^m)^*$ mit dem Polynomring $(F^m)[x]$ identifizieren, und andererseits in $(F^*)^m$ Längenunterschiede von Wörtern über F mit Hilfe von Nullen ausgleichen, ohne die Grade der entsprechenden Polynome zu verändern. Dies liefert einen kanonischen Isomorphismus $(F^m)[x] \cong (F[x])^m$. Nun kann man z.B. die Theorie der Block-Codierung auf die Vektorräume $(F[x])^k$ bzw. $(F[x])^n$ übertragen, wobei $F[x]$ die Rolle des Alphabets spielt. Im Allgemeinen wird dann die Codierung einzelner k-Blöcke wie auch die Decodierung einzelner n-Blöcke *kontextsensitiv* sein.

Es wird sich herausstellen, daß endliche Wörter bzw. Polynome über F^m nicht immer ausreichen, stattdessen werden wir die Menge $(F^m)^\omega$ der durch natürliche Zahlen indizierten unendlichen Blockfolgen oder *Ströme* $\langle\, \boldsymbol{u}_t \in F^m : t \in \mathbb{N} \,\rangle$, betrachten, bzw. im Fall, daß F ein Körper ist, die Menge $(F^m)[\![x]\!]$ der *formalen Potenzreihen* über F^m. Hier gibt es ebenfalls einen kanonischen Isomorphismus $(F^m)[\![x]\!] \cong (F[\![x]\!])^m$, der den obigen Isomorphismus erweitert.

Als Basis für die Codierung von Strömen werden wir endliche *Graphen* mit Kantenlabeln verwenden, in diesem Fall spricht man von einer *Trellis-Codierung*. Die Idee dabei ist, einen

Strom von Klartextwörtern als unendlichen Pfad mit spezifiziertem Anfangsknoten darzustellen, und die Kantenlabel zur Codierung zu nutzen.

Graphen können im Allgemeinen durch zwei Abbildungen $\mathcal{G} = (E \xrightarrow{\partial_0, \partial_1} V)$ modelliert werden. Dabei ist V die Menge der *Knoten* oder *Zustände*, E die Menge der *Kanten* oder *Übergänge* und die Funktionen ∂_0 und ∂_1, genannt *Domain* und *Codomain*, ordnen den Kanten den jeweiligen Start- und Zielknoten zu. Graphen sind also zunächst einmal *gerichtet* und können beliebige *Loops* wie auch Mehrfachkanten aufweisen.

Im Hinblick auf ihre Nutzbarkeit für die Codierung sind weitere Eigenschaften zu fordern:

- $\mathcal{G}$ soll *irreduzibel* sein in dem Sinne, daß je zwei Knoten auf einem gerichteten Kreis liegen (mit anderen Worten, der Graph hat nur eine *starke Komponente* im Sinne der Vorlesung "Theoretische Informatik 2");
- es soll eine Zahl $M \in \mathbb{N}$ geben, so daß jeden Knoten von $\mathcal{G}$ genau M Kanten verlassen.

Die zweite Bedingung dient dazu, Klartext-Mengen der Form F^k aus $M = q^k$ Wörtern sinnvoll bearbeiten zu können: für jeden Knoten des Graphen soll jedes Klartext-Wort eindeutig eine"Richtung" bestimmen, entlang der wir zu einem weiteren Knoten fortschreiten. Wir können sogar E mit dem cartesischen Produkt $V \times F^k$ identifizieren und auf diese Weise die Kanten selbst durch Angabe des Startpunkts und einer Richtung spezifizieren. Wählt man einen Anfangsknoten $i \in V$, so entsprechen Wörter aus $(F^k)^*$ (bzw. Ströme aus $(F^k)^\omega$ eindeutig (unendlichen) Pfaden durch den Graphen, die bei i beginnen. Die Forderung nach Irreduzibilität stellt dann sicher, daß alle Knoten des Graphen durch Pfade unbeschränkt großer Länge erreichbar sind. Man vermeidet damit Knoten, die gar nicht oder nur entlang "kurzer" Pfade erreichbar sind. Es handelt sich also um eine Effizienzbedingung.

Zur Codierung braucht man darüberhinaus noch ein Label-Funktion von der Kantenmenge E des Graphen in eine geeignete Potenz des Zielalphabets, das hier von der Form F^n sein wird. Die Abbildung $E \xrightarrow{L} F^n$ muß eine weitere Bedingung erfüllen:

- starten verschiedene endliche Pfade π und ρ am Knoten $u \in V$, so müssen sich entweder ihre Endpunkte unterscheiden, oder die Wörter $L^*(\pi), L^*(\rho) \in (F^n)^*$, die durch Konkatenation der Label entlang der entsprechenden Kanten entstehen.

Diese Bedingung verschärft eine Forderung, der man häufig im Bereich der *markierten Transitionssysteme* ("labeled transition systems"), oder kurz *LTS*, begegnet: nicht nur parallele Kanten sondern auch parallele endliche Pfade sollen verschiedene Label tragen. Ein derartiges LTS heißt auch *verlustfrei* ("lossless"). Zwar kann dann immer noch verschiedenen Klartextwörtern dasselbe Codewort zugeordnet werden, aber zumindest enden die entsprechenden Pfade in verschiedenen Knoten. Genau besehen gehört also die Angabe des Endknotens also mit zum Codewort. Diese Komplikation entfällt, wenn wir nur Wörter codieren, die einen zyklischen Pfad erzeugen, zurück in den Anfangszustand i. Genau das wird im Folgenden die Strategie sein.

7.1 Trellis-Graphen

Da wir potentiell unendliche Pfade in unseren Graphen betrachten wollen, bietet es sich der Übersichtlichkeit halber an, die Graphen entlang der diskreten Zeitachse "abzurollen". Speziell die Decodierung mittels des Viterbi-Algorithmus in Abschnitt 7.3 wird sich in diesem abgeleiteten Graphen übersichtlicher darstellen lassen.

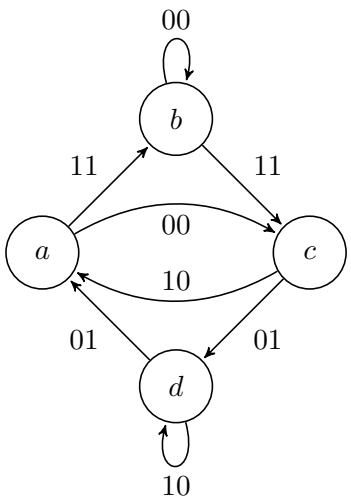
7.1.01 Definition. Zu einem gegebenen Graphen $\mathcal{G} = (E \xrightarrow{\partial_0, \partial_1} V)$ definieren wir den *Trellis-Graphen* $T(G) = \big(T(E) \xrightarrow{\partial_0, \partial_1} T(V)\big)$ wie folgt:

$$T(V) := V \times \mathbb{N} \quad , \quad T(E) := E \times \mathbb{N} \quad , \quad \partial_0\langle e, n\rangle := \langle \partial_0 e, n\rangle \quad \text{und} \quad \partial_1\langle e, n\rangle := \langle \partial_1 e, n+1\rangle$$

Es handelt sich bei $T(G)$ um das Produkt von $\mathcal{G}$ mit dem durch die Nachfolgerfunktion $\mathbb{N} \xrightarrow{s} \mathbb{N}$ bestimmten *irreflexiven* Graphen $\mathbb{N}_s$, der nur Kanten der Form $n \longrightarrow s(n) := n+1$ besitzt und diskrete Zeit modelliert. Insbesondere sind Eigenschaften wie Irreduzibilität und M-Regularität von $\mathcal{G}$ unerheblich für diese Konstruktion.

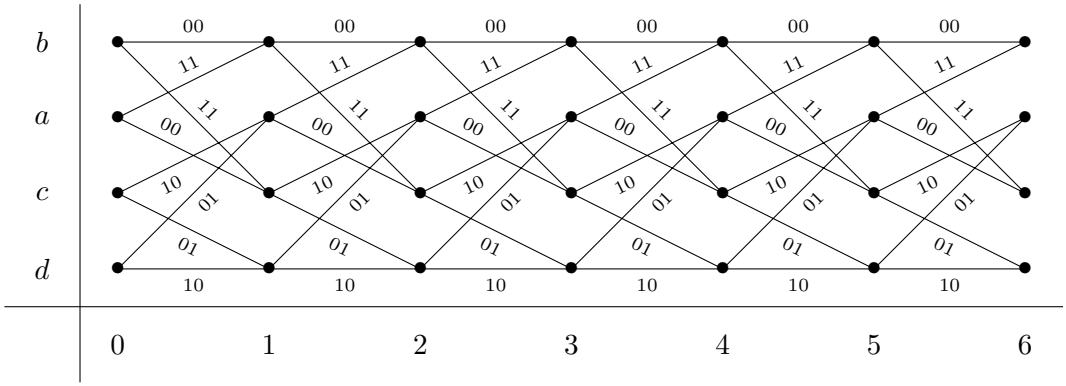
Eine Label-Funktion $E \xrightarrow{L} F^n$ für $\mathcal{G}$ liefert eine kanonische Label-Funktion für $T(G)$ durch Verknüpfung mit der Projektion $E \times \mathbb{N} \longrightarrow E$. Und jede Wahl eines Anfangszustands $i \in V$ führt zu einem Anfangszustand $\langle i, 0\rangle$ im Trellis-Graphen.

7.1.02 Beispiel. Wir betrachten das folgende markierte Transitionssystem über $\mathrm{GF}(2)^2$:



(7.1-47)

Der Beginn des zugehörigen Trellis-Systems ist gegeben durch



(7.1-48)

7.1.03 Definition. Für ein markiertes Transitionssystem mit einem irreduziblen M-regulären Graphen $\mathcal{G} = (E \xrightarrow{\partial_0, \partial_1} V)$ mit gewähltem Anfangszustand $i \in V$ und einer Labelfunktion $E \xrightarrow{L} F^n$ verstehen wir unter dem *Trellis-Code* die Menge $\mathcal{C}(\mathcal{G}, i, L) \subseteq (F^n)^\omega$ aller L-induzierten unendlichen Wörter entlang unendlicher Pfade in $\mathcal{G}$, oder äquivalent $T(G)$, die bei i bzw. $\langle i, 0\rangle$ starten. Die *Coderate* von $\mathcal{C}(\mathcal{G}, i, L)$ definieren wir als

$$R := \frac{\log_{|F|} M}{n}$$

Für $l \in \mathbb{N}$ bezeichen $\mathcal{C}_l(\mathcal{G}, i, L) \subseteq (F^n)^l$ und $\mathcal{C}_l^\circ(\mathcal{G}, i, L) \subseteq (F^n)^l$ die Mengen L-induzierter Wörter der Länge l über F^n entlang beliebiger, bzw. zyklischer Pfade der Länge l mit Startzustand $i \in V$ bzw. $\langle i, 0\rangle \in V \times \mathbb{N}$.

Ohne Beweis stellen wir fest, daß die Coderaten der Block-Codes die Rate des Trellis-Codes approximieren. Mit $F = \mathrm{GF}(q)$ gilt

7.1.04 Proposition.

$$\lim_{l\to\infty} \frac{\log_{q^n} |\mathcal{C}_l(\mathcal{G}, i, L)|}{l} = \limsup_{l\to\infty} \frac{\log_{q^n} |\mathcal{C}_l^\circ(\mathcal{G}, i, L)|}{l} = \frac{\log_q M}{n}$$

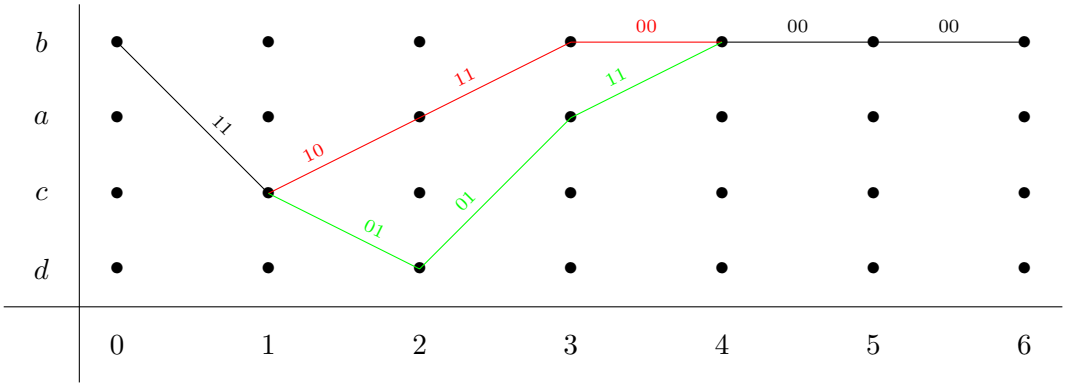
Der Limes Superior im mittleren Term ist nötig, weil der Anfangszustand i nicht auf zyklischen Pfaden beliebiger Länge l liegen muß; $\mathcal{C}_l^\circ(\mathcal{G}, i, L)$ kann auch leer sein. Für welche Werte l dies nicht der Fall ist, wird in Proposition 7.2.01 geklärt.

Wörter der Länge l über F^n sind natürlich Wörter der Länge $n \cdot l$ über F. Während der Hamming-Abstand Δ von l-Tupeln in $(F^n)^l$ nur die Gleichheit bzw. Verschiedenheit der Komponenten aus F^n berücksichtigt, stellt es sich heraus, daß der *Hamming-Abstand* Δ_F *über* F, der die $n \cdot l$ Komponenten aus F betrachtet, sinnvollere Ergebnisse liefert. In Analogie zum Minimalabstand bei Block-Codes definiert man für Trellis-Codes den *freien Abstand* d_{free}:

7.1.05 Definition. Für $C = \mathcal{C}(\mathcal{G}, i, L)$ setzen wir

$$d_{\text{free}} := \min \left\{ \sum_{t \in \mathbb{N}} \Delta_{\text{H}}(x_t, y_t) : \langle \boldsymbol{x}, \boldsymbol{y} \rangle \in C^2 \wedge \boldsymbol{x} \neq \boldsymbol{y} \right\}$$

7.1.06 Beispiel. In Beispiel 7.1.02 wählen wir b als Anfangszustand und betrachten zwei Pfade der Länge 6:



Aus $d_F\langle \boldsymbol{x}, \boldsymbol{y} \rangle = 2 + 1 + 2 = 5$ folgern wir $d_{\text{free}}(C) \leq 5$, denn für die Bestimmung des freien Abstandes benötigen wir minimale "inselförmige" Abweichungen von Pfaden, die ansonsten übereinstimmen. In diesem Fall gilt sogar $d_{\text{free}} = 5$.

Offenbar ist für jedes $l > 0$ der Minimalabstand $d_F(\mathcal{C}_l^\circ(\mathcal{G}, i, L))$ eine obere Schranke für d_{free}: jede Abweichung obiger Form zwischen zwei Pfaden in den ersten l Zeiteinheiten wird bei der Minimierung in der Definition von d_{free} berücksichtigt. Andererseits können bei der Berechnung von d_{free} Abweichungen auftreten, die nicht innerhalb eines Blocks der Länge l liegen.

7.2 Trellis-Codierung

Wie zu Beginn des Kapitels angedeutet, wollen wir zur Trellis-Codierung nur zyklische Pfade im Graphen $\mathcal{G}$ verwenden; genauer, wir wollen Codewörter aus Elementen von $\mathcal{C}_l^\circ(\mathcal{G}, i, L)$ für geeignetes l zusammensetzen. Zumindest der Anfangszustand $i \in V$ sollte auf einem Kreis der Länge l liegen. Und damit wir den Anfangszustand bei Bedarf auch ändern können, sollten auch alle übrigen Zustände dieser Bedingung genügen. Zum Abschätzen von l definieren wir die *Periode* $p(\mathcal{G})$ als den größten gemeinsamen Teiler der Längen aller einfachen Kreise in $\mathcal{G}$. Für welche Vielfachen von $p(\mathcal{G})$ können wir für jeden Zustand garantieren, daß er auf einem Kreis der entsprechenden Länge liegt?

7.2.01 Proposition. *Ist p der* ggT *einer Menge $\mathcal{L}$ positiver Zahlen, so existiert $t_0 \in \mathbb{N}$ derart, da für jedes $t \geq t_0$ das Produkt $t \cdot p$ als $\mathbb{N}$-Linearkombination der Elemente von $\mathcal{L}$ darstellbar ist.*

Beweis: Offenbar genügt es, sich auf den Fall $p = 1$ zu beschränken.

Aufgrund des Euklidischen Algorithmus ist klar, daß der ggT 1 als $\mathbb{Z}$-Linearkombination der Elemente von $\mathfrak{L}$ darstellbar ist, etwa

$$1 = \sum\{\, b_l l : l \in \mathfrak{L} \,\}$$

Wir separieren die positiven und die negativen Beiträge

$$\alpha := \sum\{\, [b_l > 0] b_l l : l \in \mathfrak{L}\} \quad \text{und} \quad \beta := -\sum\{\, [b_l < 0] b_l l : l \in \mathfrak{L}\}$$

und setzen $t_0 := \beta(\beta - 1)$. Jede Zahl $t \geq t_0$ können wir nun als $t = q \cdot \beta + s$ darstellen, wobei $q \geq \beta - 1$ und $s < \beta$ gilt. Wegen $\alpha - \beta = 1$ erhalten wir nun

$$t = s \cdot \alpha + (q - s) \cdot \beta$$

was eine positive Linearkombination der Zahlen aus $\mathfrak{L}$ ist. □

Folglich kommen ab einem bestimmten Faktor t_0 alle größeren Vielfachen der Periode $p(\mathcal{G})$ als sinnvolle Werte für l in Frage.

Natürlich liefern nicht alle Wörter aus $(F^n)^l$ von Anfangszustand i ausgehend zyklische Pfade zurück nach i. Gelänge es, eine nur vom Graphen $\mathcal{G}$ abhängige Zahl Ξ zu finden, die sogenannte "Backlength", so daß für jedes hinreichend großen Vielfache l von $p(\mathcal{G})$ jeder Pfad der Länge $h = l - \Xi$ zu einem zyklischen Pfad der Länge l vervollständigt werden kann, so könnten zumindest Wörter aus $(F^n)^h$ mit dem oben angedeuteten Verfahren codiert werden. Bevor wir uns der Backlength direkt zuwenden können, bedarf es einiger Vorbereitung.

7.2.02 Definition. Für einen Graphen $\mathcal{G}$ besteht die Relation $\sim_{p(\mathcal{G})} \subseteq V^2$ aus genau den Knotenpaaren $\langle u, v \rangle$, so daß v von u aus entlang eines gerichteten Pfades erreichbar ist, dessen Länge durch $p(\mathcal{G})$ geteilt wird.

7.2.03 Lemma. *$\sim_{p(\mathcal{G})}$ ist eine Äquivalenzrelation.*

Beweis: Die Reflexivität und die Transitivität sind trivial. Zum Nachweis der Symmetrie wählen wir für $u \sim_{p(\mathcal{G})} v$ einen Pfad $u \xrightarrow{\pi} v$ mit $p(\mathcal{G}) \,|\, |\pi|$. Aber es gibt auch einen Pfad $v \xrightarrow{\rho} u$, der π zu einem Kreis ergänzt. Dieser Kreis läßt sich aus einfachen Kreisen zusammensetzen, seine Länge ist also ein Vielfaches von $p(\mathcal{G})$. Folglich muß auch $|\rho|$ durch $p(\mathcal{G})$ teilbar sein. □

7.2.04 Satz. *Für jeden Graphen $\mathcal{G}$ existiert eine Zahl $\Xi > 0$, so daß aus $u \sim_{p(\mathcal{G})} v$ die Existenz eines Pfades von v nach u der Länge Ξ folgt. Insbesondere ist Ξ Vielfaches von $p(\mathcal{G})$.*

Beweis: Wir wählen einen Knoten $x \in V$ und zu jedem $u \in V$ einen nichtleeren Kreis K_u minimaler Länge, der x und u enthält. K_u bestimmt Pfade $u \xrightarrow{\pi_u} x$ und $x \xrightarrow{\rho_u} u$.

Für $\langle u, v\rangle \in \sim_{p(\mathcal{G})}$ betrachten wir einen Pfad $u \xrightarrow{\sigma_{u,v}} v$, dessen Länge durch $p(\mathcal{G})$ teilbar ist. Dies gilt folglich auch für den aus einfachen Kreisen zusammensetzbaren Kreis $\sigma_{u,v}\pi_v\rho_u$, und somit auch für den Pfad $\pi_v\rho_u$. Wir setzen

$$t_2 := \max\{\, |\pi_v| + |\rho_u| : u \sim_{p(\mathcal{G})} v \,\}$$

Da die Kreise K_u, $u \in V$, aus einfachen Kreisen zusammengesetzt sind, ist der ggT P ihrer Längen zumindest ein Vielfaches von $p(G)$.

Nach Proposition 7.2.01 existiert eine Zahl t_1, so daß für jedes $t \geq t_1$ das Produkt $t \cdot P$ als positive Linearkombination der Kreislängen $|K_u|$, $u \in V$, darstellbar ist. Nun ist $\Xi := t_1 \cdot p(\mathcal{G}) + t_2$ ein Vielfaches von $p(G)$.

Für jedes Paar $\langle u, v\rangle \in \sim_{p(\mathcal{G})}$ wählen wir einen Kreis $C_{v,u}$ der Länge $t_1 \cdot p(\mathcal{G}) + t_2 - (|\pi_v| + |\rho_u|)$, der sich aus Kreisen K_u, $u \in V$, zusammensetzt. Der Pfad $\pi_v C_{v,u} \rho_u$ von v nach u hat nun die Länge Ξ. □

7.2.05 Bemerkung. Die Zahl Ξ ist sicherlich nicht eindeutig bestimmt; je kleiner Ξ desto effizienter wird die Codierung sein. Aus diesem Grund erscheint es wünschenswert, mit einer Menge $\mathcal{R}$ nicht notwendig einfacher Kreise zu arbeiten die alle den Knoten x enthalten, und $p(\mathcal{G})$ als ggT ihrer Längen haben. Falls die Kreise K_u, $u \in V$, diese Bedingung nicht erfüllen, vergrößern wir hinreichende viele ausgewählte Exemplare durch einfache Kreise geeigneter Länge, bis der gewünschte ggT erreicht ist.

Eine Trellis-Codierung über einem Alphabet A mit $|A| = M$ und einem Körper $F = \mathrm{GF}(q)$ läßt sich nun wie folgt realisieren:

- wähle einen M-regulären irreduziblen Graphen $\mathcal{G}$, zweckmäßigerweise mit der Kantenmenge $E = V \times A$;
- berechne $p(\mathcal{G})$ und bestimme einen möglichst kleinen Wert für die Backlength Ξ;
- wähle eine Labelfunktion $E \xrightarrow{L} F^n$, so daß das resultierende LTS lossless ist;
- wähle einen Anfangszustand $i \in V$ und für jeden Zustand $u \in [i]_{p(\mathcal{G})}$ einen festen Pfad von u nach i der Länge Ξ;
- wähle l als hinreichend großes Vielfaches von $p(\mathcal{G})$ mit $l > \Xi$.

Die Codierungsfunktion $A^{l-\Xi} \xrightarrow{\mathcal{E}_l^\circ} (F^n)^l$ operiert nun in drei Stufen:

- Zunächst wird $a \in A^{l-\Xi}$ der entsprechende Pfad in $\mathcal{G}$ von i aus zugeordnet; sein Endpunkt gehört nach Konstruktion zu $[i]_{p(\mathcal{G})}$.
- Dieser Pfad wird durch den vorher fest gewählten Pfad der Länge Ξ zu einem zyklischen Pfad von i nach i ergänzt.

- Die Konkatenation der Label entlang dieses verlängerten Pfades bestimmen das Codewort.

Angesichts des großen Spielraums beim Design einer Trellis-Codierung erscheint es sinnvoll, die Konstruktion des LTS etwas einzuschränken, siehe Abschnitt 7.4. Zuvor wollen wir uns aber der Decodierung mittels des Virterbi-Algorithmus zuwenden.

7.3 Decodierung mit dem Viterbi-Algorithmus

Wird ein Codewort $\boldsymbol{c} = \boldsymbol{c}_0\boldsymbol{c}_1\ldots\boldsymbol{c}_{l-1} \in C_l^\circ(\mathcal{G}, i, L) \subseteq (F^n)^l$ durch einen probabilistischen Kanal $S = \langle F, \Phi, P\rangle$ geschickt, so erhalten wir eine Wortfolge $\boldsymbol{y} = \boldsymbol{y}_0\boldsymbol{y}_1\ldots\boldsymbol{y}_{l-1} \in (\Phi^n)^l$. Ein MLD bezüglich S sollte dieser Folge ein Codewort $\hat{\boldsymbol{c}} = \hat{\boldsymbol{c}}_0\hat{\boldsymbol{c}}_1\ldots\hat{\boldsymbol{c}}_{l-1} \in C_l^\circ(\mathcal{G}, i, L)$ zuordnen, das

$$P(\,\boldsymbol{y}\ \text{empfangen}\,|\,\hat{\boldsymbol{c}}\ \text{gesendet}\,)$$

maximiert. Unter der Annahme, daß es sich um einen gedächtnislosen Kanal handelt, gilt

$$P(\,\boldsymbol{y}\ \text{empfangen}\,|\,\hat{\boldsymbol{c}}\ \text{gesendet}\,) = \prod_{i<l} P(\,\boldsymbol{y}_i\ \text{empfangen}\,|\,\hat{\boldsymbol{c}}_i\ \text{gesendet}\,)$$

Die Idee ist nun, dieses Maximierungs- in ein (Kosten-)Minimierungsproblem umzuwandeln. Das gelingt durch Logarithmieren zu einer beliebigen Basis > 1 und einen Vorzeichenwechsel:

$$\sum_{i<l} -\log P(\,\boldsymbol{y}_i\ \text{empfangen}\,|\,\hat{\boldsymbol{c}}_i\ \text{gesendet}\,)$$

Abkürzend schreiben wir

$$\chi\langle\boldsymbol{y}_i, \hat{\boldsymbol{c}}_i\rangle := -\log P(\,\boldsymbol{y}_i\ \text{empfangen}\,|\,\hat{\boldsymbol{c}}_i\ \text{gesendet}\,)$$

und bezeichnen diese Größe als die *Kosten* von $\boldsymbol{c}_i$ bzgl. $\boldsymbol{y}_i$, $i < l$.

7.3.01 Beispiel. $S = \langle F, F, P\rangle$ sei ein gedächtnisloser q-närer Kanal mit Umwandlungswahrscheinlichkeit $p < 1 - 1/q = (q-1)/q$. Für $i < l$ gilt nun

$$\begin{aligned} P(\boldsymbol{y}_i\ \text{empfangen}\,|\,\hat{\boldsymbol{c}}_i\ \text{gesendet}\,) &= \Big(\frac{p}{q-1}\Big)^{\Delta\langle\boldsymbol{y}_i,\hat{\boldsymbol{c}}_i\rangle}(1-p)^{l-\Delta\langle\boldsymbol{y}_i,\hat{\boldsymbol{c}}_i\rangle} \\ &= (1-p)^l\left(\frac{p}{(1-p)(q-1)}\right)^{\Delta_H\langle\boldsymbol{y}_i,\hat{\boldsymbol{c}}_i\rangle} \end{aligned}$$

Die Kosten berechnen sich nun zu

$$\chi\langle\boldsymbol{y}_i, \hat{\boldsymbol{c}}_i\rangle = -l\cdot\log(1-p) - \log\left(\frac{p}{(1-p)(q-1)}\right)\Delta_H\langle\boldsymbol{y}_i, \hat{\boldsymbol{c}}_i\rangle$$

Wegen $p < (q-1)/q$ und $1-p > 1/q$ folgern wir

$$\frac{p}{(1-p)(q-1)} < \frac{1}{(1-p)q} < 1 \quad\text{also}\quad \log\left(\frac{p}{(1-p)(q-1)}\right) < 0$$

Folglich liefert in diesem Fall die MLD ein Codewort, das den Hamming-Abstand zum empfangenen Wort minimiert, d.h.,

$$\sum_{i<l} \Delta_H\langle \boldsymbol{y}_i, \hat{\boldsymbol{c}}_i\rangle = \Delta_F\langle \boldsymbol{y}_0\boldsymbol{y}_1\ldots\boldsymbol{y}_{l-1}, \hat{\boldsymbol{c}}_0\hat{\boldsymbol{c}}_1\ldots\hat{\boldsymbol{c}}_{l-1}\rangle \quad \text{minimal}$$

Der Viterbi-Algorithmus sucht einen Pfad $\langle\, e_i : i < l\,\rangle$ von $\langle i,0\rangle$ nach $\langle i,l\rangle$ in $T(\mathcal{G})$, so daß die Kostensumme

$$\sum_{i<l} \chi\langle \boldsymbol{y}_i, L(e_i)\rangle$$

minimiert wird. Aufgrund des obigen Beispiels ist es im Falle eines gedächtnislosen q-nären Kanals rechentechnisch natürlich viel einfacher, stattdessen den Hamming-Abstand

$$\sum_{i<l} \Delta\langle \boldsymbol{y}_i, L(e_i)\rangle = \Delta_F\langle \boldsymbol{y}_0\boldsymbol{y}_1\ldots\boldsymbol{y}_{l-1}, L^*(e_0e_1\ldots e_{l-1})\rangle$$

zu minimieren. Zu diesem Zweck ordnen wir jedem Knoten $\langle s,t\rangle \in V \times \{\, t \in \mathbb{N} : t \le l\,\}$ die minimalen "Reisekosten" von $\langle i,0\rangle$ nach $\langle s,t\rangle$ zu:

$$\chi_t(s) := \min\{\, \chi(\pi) : \langle i,0\rangle \xrightarrow{\pi} \langle s,t\rangle\,\}$$

Existiert kein Pfad von $\langle i,0\rangle$ nach $\langle s,t\rangle$, so setzen wir $\chi_t(s) := \infty$, während natürlich $\chi_0(i) = 0$ gilt.

Die Berechnung der Knotenkosten $\langle\, \chi_t(s) : s \in V\,\rangle$, $t \le l$, erfolgt iterativ, zusammen mit Pfaden $\langle i,0\rangle \xrightarrow{\pi} \langle s,t\rangle$, die diese Kosten realisieren. Diese sind nicht notwendig eindeutig bestimmt, insofern ist der Algorithmus nichtdeterministisch. Die Knotenkosten genügen folgender Rekursionsformel

$$\chi_{t+1}(s) = \min\{\, \chi_t(e) + \chi_t(\partial_0(e)) : e \in E \wedge \partial_1(e) = s\,\}$$

7.3.02 Algorithmus. (Viterbi Algorithmus)
Eingabe: Kantenkosten $\chi_t(e) := \Delta_H\langle \boldsymbol{y}_t, L(e)\rangle$ für $e \in E$ und $t < l$.
Ausgabe: Pfad $\langle i,0\rangle \xrightarrow{\pi_{\min}} \langle i,l\rangle$

(0) Initialisierung: für jedes $s \in V$ setze

$$\chi_0(s) := \begin{cases} 0 & \text{falls } s = i \\ \infty & \text{sonst} \end{cases}$$

(1) Doppelschleife: Für $t < l$ und für $s \in V$

(a) finde und markiere alle Kanten e mit $\partial_1(e) = s$ und $\chi_t(e) + \chi_t(\partial_0(e))$ minimal.

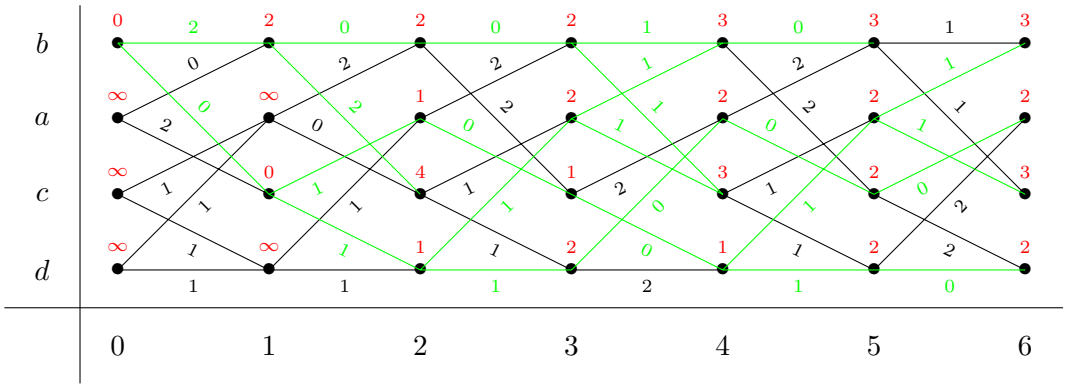
(b) definiere $\chi_{t+1}(s) := \chi_t(e) + \chi_t(\partial_0(e))$ für eine derartige Kante e.

(2) Bestimme $\langle i,0\rangle \xrightarrow{\pi_{\min}} \langle i,l\rangle$ durch Backtracking, indem ausgehend von $\langle i,l\rangle$ markierte Kanten rückwärts durchlaufen werden; hier kann es Verzweigungen und folglich Nichtdeterminismus geben, aber jeder solche rückwärtsgerichtete Pfad endet bei $\langle i,0\rangle$.

7.3.03 Beispiel. Für den Graphen in Beispiel 7.1.02 mit Anfangszustand b betrachten wir die Folge empfangener Wörter

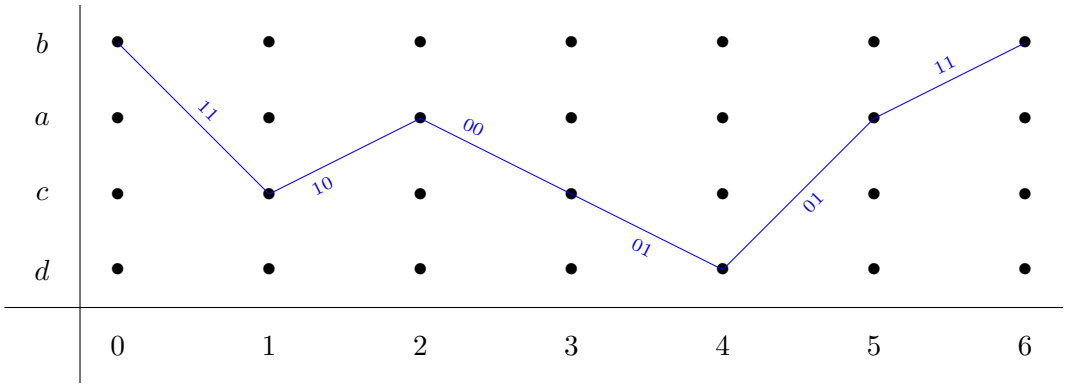
$$\boldsymbol{y}_0\boldsymbol{y}_1\boldsymbol{y}_2\boldsymbol{y}_3\boldsymbol{y}_4\boldsymbol{y}_5 = 11\,00\,00\,01\,00\,10$$

Im Trellies-Diagramm 7.1-48 ersetzen wir die zunächst die Kantenlabel durch ihre Hammingabstände zu den jeweiligen empfangenen Wörtern und die Knoten zur Zeit $t = 0$ durch die initialen Werte. Anschließend werden die Knotenkosten (rot) und die entsprechenden minimierenden Pfade (grün) für die Zeitpunkte $0 < t \leq l$ mittels des Viterbi-Algorithmus bestimmt:



(7.3-49)

Der (eindeutig bestimmte) Minimalpfad $\pi_{\min}$ wird durch Backtracking von $\langle b,6\rangle$ aus bestimmt und in Diagramm 7.1-48 übertragen:



(7.3-50)

was schließlich folgendes Codewort liefert:

$$\hat{\boldsymbol{c}}_0\hat{\boldsymbol{c}}_1\hat{\boldsymbol{c}}_2\hat{\boldsymbol{c}}_3\hat{\boldsymbol{c}}_4\hat{\boldsymbol{c}}_5 = 11\,10\,00\,01\,01\,11$$

Der Klartext ergibt sich nun aus dem Codewort als Folge der "Richtungen" entlang des spezifizierten Pfades.

7.4 LFSM'n und Faltungscodes

Für $F = \mathrm{GF}(q)$ bilden lineare $[n,k]_q$-Block-Codes Klartextwörter $\boldsymbol{u} \in F^k$ auf Codewörter $\boldsymbol{c} = \boldsymbol{u} \cdot G \in F^n$ ab, wobei $G \in \boldsymbol{Mat}_F(k,n)$ eine Generatormatrix ist. Eine lineare Variante der Trellis-Codierung würde G durch eine $k \times n$-Matrix mit Komponenten aus $F[\![x]\!]$ ersetzen. Dies erscheint zunächst wenig praktikabel, da solch eine Matrix nur selten endlich spezifiziert werden kann.

Man kann die Linearität aber auch an anderer Stelle einfließen lassen, nämlich bei der Konstruktion des Graphen: sogenannte "linear finite state machines" (LFSMs) sind besonders leicht handhabbar und führen letztendlich zu endlich spezifizierbaren Generatormatrizen mit rationalen Funktionen als Komponenten (also "Brüchen" von Polynomen), anstelle von Potenzreihen. Die resultierenden Trellis-Codes lassen sich direkt mit Hilfe von Schieberegistermaschinen realisieren und sind als *Faltungs-Codes* bekannt.

Als Knoten-, Richtungs- und Labelmengen für LFSMs verwenden wir Vektorräume über F. Das erlaubt uns, den durch Start-Knoten und Richtung spezifizierten Kanten in linearer Weise Ziel-Knoten bzw. Label zuordnen, indem wir dies linear für jede Komponente getrennt tun, und die Ergebnisse addieren.

7.4.01 Definition. Über einem Körper $F = \mathrm{GF}(q)$ wird eine $(k:n)$-LFSM mit k Inputs, n Outputs und q^m Zuständen als LTS wie folgt bestimmt: Neben den Knoten- und Kantenmengen

$$V := F^m \quad \text{bzw.} \quad E := V \times F^k = F^m \times F^k$$

und der Projektion $F^m \times F^k \longrightarrow F^m$ auf die erste Komponente als Domain-Funktionen ∂_0, sind die Codomain-Funktion ∂_1 sowie die Labelfunktion L mit Hilfe von vier Matrizen

$$P \in \boldsymbol{Mat}_F(m,m) \quad , \quad B \in \boldsymbol{Mat}_F(k,m) \quad , \quad Q \in \boldsymbol{Mat}_F(m,n) \quad \text{sowie} \quad D \in \boldsymbol{Mat}_F(k,n)$$

gegeben durch

$$\partial_1\langle s,u\rangle := sP + uB \quad \text{und} \quad L\langle s,u\rangle := sQ + uD$$

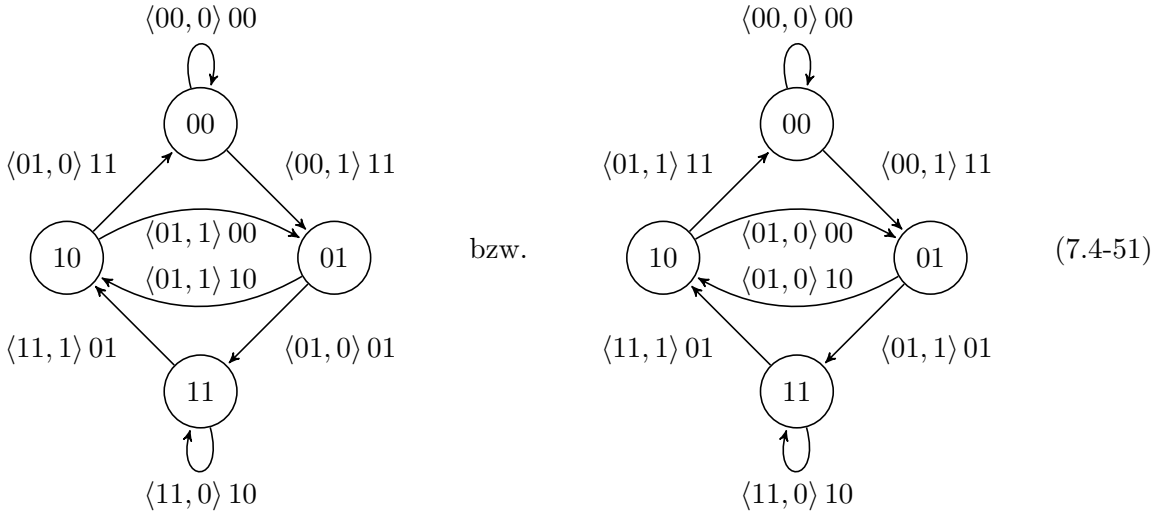
7.4.02 Beispiel. Für die Werte $m = 2$, $k = 1$ und $n = 2$ wählen wir über $F = \mathrm{GF}(2)$ einerseits die Matrizen

$$P = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \quad , \quad B = \begin{pmatrix} 0 & 1 \end{pmatrix} \quad , \quad Q = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \quad \text{sowie} \quad D = \begin{pmatrix} 1 & 1 \end{pmatrix}$$

und zum anderen die Matrizen

$$\tilde{P} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \quad , \quad \tilde{B} = \begin{pmatrix} 0 & 1 \end{pmatrix} \quad , \quad \tilde{Q} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \quad \text{sowie} \quad \tilde{D} = \begin{pmatrix} 1 & 1 \end{pmatrix}$$

Diese liefern folgende Varianten des aus 7.1-47 bekannten Graphen, dessen Kanten wir mit ihrem Namen aus $F^2 \times F$ ("Anfang und Richtung") und dem Label in F^2 versehen haben:



(7.4-51)

die sich nur in den Kanten-Richtungen in den Knoten 01 und 10 unterscheiden.

Ausgehend von einem Strom von Klartextwörtern $\boldsymbol{u}_t \in F^k$, $t \in \mathbb{N}$, und einem Anfangszustand $i = \boldsymbol{s}_0$ erhalten wir einen Strom von Zuständen und einen Strom von Codewörtern

$$\boldsymbol{s}_{t+1} := \partial_1\langle \boldsymbol{s}_t, \boldsymbol{u}_t\rangle = \boldsymbol{s}_t P + \boldsymbol{u}_t B \quad \text{bzw.} \quad \boldsymbol{c}_t := L\langle \boldsymbol{s}_t, \boldsymbol{u}_t\rangle = \boldsymbol{s}_t Q + \boldsymbol{u}_t D \tag{7.4-52}$$

Diese lassen sich als Potenzreihen über F^k bzw. F^n zusammenfassen, oder als entsprechende Tupel von Potenzreihen, wobei zwingend steigende Potenzen von x zu verwenden sind:

$$\begin{aligned} u(x) &:= \sum_{t\in\mathbb{N}} \boldsymbol{u}_t x^t \in (F^k)[\![x]\!] \cong (F[\![x]\!])^k \\ s(x) &:= \sum_{t\in\mathbb{N}} \boldsymbol{s}_t x^t \in (F^m)[\![x]\!] \cong (F[\![x]\!])^m \\ c(x) &:= \sum_{t\in\mathbb{N}} \boldsymbol{c}_t x^t \in (F^n)[\![x]\!] \cong (F[\![x]\!])^n \end{aligned} \tag{7.4-53}$$

Unser Ziel ist es, $c(x)$ in Abhängigkeit von $u(x)$ und dem Startzustand $i = \boldsymbol{s}_0$ auszudrücken. Dafür ist zunächst $s(x)$ auf diese Weise zu behandeln.

Der Vorteil der Darstellung als Tupel von Potenzreihen ist es, daß die Matrix-Gleichungen 7.4-52 zu Matrix-Gleichungen zwischen derartigen Tupeln geliftet werden können: zunächst multiplizieren wir die linke Gleichung in 7.4-52 mit x^{t+1} und summieren dann über t:

$$s(x) - s_0 = s(x) \cdot xP + u(x) \cdot xB$$

Dabei ist s_0 als m-Tupel $(s_0\, 0\, 0 \ldots, 0\rangle$ aufzufassen, während xP und xB als Matrizen mit Komponenten aus $F[x] \subseteq F[\![x]\!]$ zu interpretieren sind. Auflösen nach $s(x)$ liefert zunächst

$$s(x)(I_m - xP) = u(x) \cdot xB + \boldsymbol{s}_0$$

wobei I_m die $(m \times m)$-Einheitsmatrix bezeichnet. Im entsprechenden Matrixring über $F[\![x]\!]$ ist $(I - xP)$ in der Tat invertierbar, denn offenbar gilt

$$(I - xP) \sum_{t \in \mathbb{N}} x^t P^t = \sum_{t \in \mathbb{N}} x^t P^t - \sum_{t \in \mathbb{N}} x^{t+1} P^{t+1} = x^0 P^0 = I_m$$

Dies ermöglicht die explizite Bestimmung von $s(x)$ zu

$$s(x) = \big(u(x) \cdot xB + s_0\big)(I_m - xP)^{-1} \tag{7.4-54}$$

Einsetzen in die offensichtliche Gleichung $c(x) = s(x) \cdot Q + u(x) \cdot D$ liefert nun

$$\begin{aligned} c(x) &= \big(u(x) \cdot xB + s_0\big)(I_m - xP)^{-1} \cdot Q + u(x) \cdot D \\ &= u(x)\Big(xB \cdot (I - xP)^{-1} \cdot Q + D\Big) + \boldsymbol{s}_0 (I - xP)^{-1} \cdot Q \end{aligned}$$

Als *Response-Matrix* bezeichnen wir abkürzend

$$G(x) := xB \cdot (I_m - xP)^{-1} \cdot Q + D \in \boldsymbol{Mat}_{F[\![x]\!]}\langle k, n\rangle$$

Wie sich gleich zeigen wird, ist diese Matrix endlich spezifizierbar.

Eine besonders einfache Abhängigkeit des Codewort-Stroms $\boldsymbol{c}_t$ vom Strom der Informationswörter $\boldsymbol{u}_t$, $t \in \mathbb{N}$, ergibt sich bei Wahl des Nullvektors als Anfangszustand, d.h., $\boldsymbol{s}_0 = 0$. Wegen der Linearität ist der Pfad in Richtung $\boldsymbol{0} \in F^k$ von $\boldsymbol{0} \in F^m$ aus eine Schleife mit Label $\boldsymbol{0} \in F^n$. Daher liefern Eingabeströme aus $(F^k)[\![x]\!]$ als Ausgabe Codewortströme in $(F^n)[\![x]\!]$.

Nicht jede Auswahl von Matrizen P, B, Q und D liefert allerdings ein LTS, das lossless ist. Diese Eigenschaft von $\langle \mathcal{G}, L\rangle$ garantiert unterschiedliche Codewortströme bei verschiedenen Klartextströmen. Insofern ist das folgende Ergebnis nicht sehr überraschend:

7.4.03 Proposition. *$\langle \mathcal{G}, L\rangle$ ist genau dann lossless wenn die Response-Matrix $G(x)$ einer injektiven Abbildung entspricht, also Rang k hat.* □

Um eine endliche Beschreibung der Response-Matrix zu finden, erinnern wir daran, wie die Invertierung einer regulären Matrix U mit Hilfe der Determinante und der adjungierten Matrix formuliert werden kann:

$$U^{-1} = \frac{1}{\det U} \operatorname{Adj}(U)$$

Nun ist $\det(I_m - xP)$ ein Polynom des Grades $\leq m$ mit Wert 1 an der Stelle 0, während die Komponenten von $\operatorname{Adj}(I_m - xP)$ als Determinanten bestimmter $(m-1)\times(m-1)$-Teil-Matrizen Polynome des Grades $\leq m-1$ sind. Unter Berücksichtigung des Faktors xB lassen sich die Komponenten von $G(x)$ nun alternativ als *rationale Funktionen* über F darstellen:

$$G_{i,j}(x) = \frac{\omega_{i,j}(x)}{\sigma_{i,j}(x)} \quad \text{mit} \quad \omega_{i,j}(x), \sigma_{i,j}(x) \in F_{m+1}[x] \quad \text{und} \quad \sigma_{i,j}(0) = 1$$

wobei wir ohne Beschränkung der Allgemeinheit die Teilerfremdheit von Zähler und Nenner annehmen können; $\sigma_{i,j}(0) = 1$ kann wegen $\sigma_{i,j}(x) \mid \det(I_m - xP)$ und $\det(I_m - xP)(0) = 1$ mittels geeigneter skalarer Erweiterung realisiert werden.

Wir erinnern bei dieser Gelegenheit daran, daß die Potenzreihen über F nur einen Ring $F[\![x]\!]$, aber keinen Körper bilden, denn z.B. ist x nicht invertierbar. Andererseits bilden die rationalen Funktionen über F, d.h., die formalen Brüche aus Polynomen über F, deren Nenner von 0 verschieden sind, tatsächlich einen Körper $F(x)$. Um $F[\![x]\!]$ zu einem Körper zu erweitern, der $F(x)$ umfaßt, können wir zum Körper $F(\!(x)\!)$ der *Laurent-Reihen* übergehen, die ähnlich wie formale Potenzreihen aufgebaut sind, bei denen die Summation aber bei einer beliebigen Zahl aus $\mathbb{Z}$ beginnen darf, statt nur bei einer Zahl aus $\mathbb{N}$. Somit können endlich viele Summanden mit negativen Potenzen von x auftreten.

Jede rationale Funktion $\omega(x)/\sigma(x)$ ist als Quotient zweier Potenzreihen eine Laurentreihe, und wenn der Nenner, wie oben gefordert, die Bedingung $\sigma_0 = \sigma(0) = 1$ erfüllt, so ist der Quotient sogar eine Potenzreihe. Umgekehrt ist aber nicht jede Potenzreihe als rationale Funktion $\omega(x)/\sigma(x)$ darstellbar, deren Nenner $\sigma_0 = 1$ erfüllt, denn mittels der endlich vielen Koeffizienten können nur periodische Potenzreihen realisiert werden.

7.4.04 Beispiel. Die Matrizen P, B, Q und D aus Beispiel 7.4.02 liefern

$$I_2 - xP = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & x \\ x & x \end{pmatrix} = \begin{pmatrix} 1 & -x \\ -x & 1-x \end{pmatrix} = \begin{pmatrix} 1 & x \\ x & 1+x \end{pmatrix}$$

$$(I_2 - xP)^{-1} = \frac{1}{\det(I_2 - xP)} \operatorname{Adj}(I_2 - xP) = \frac{1}{1+x+x^2} \begin{pmatrix} 1+x & x \\ x & 1 \end{pmatrix}$$

Damit erhalten wir folgende Response-Matrix

$$\begin{aligned} G(x) &= xB \cdot (I_2 - xP)^{-1} \cdot Q + D \\ &= \frac{1}{1+x+x^2} \begin{pmatrix} 0 & x \end{pmatrix} \begin{pmatrix} 1+x & x \\ x & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & \frac{1+x^2}{1+x+x^2} \end{pmatrix} \\ &= \begin{pmatrix} 1 & \frac{(1+x)^2}{1+x+x^2} \end{pmatrix} = \begin{pmatrix} 1 & \frac{(1+x)^3}{1+x^3} \end{pmatrix} = \begin{pmatrix} 1 & (1+x)^3 \sum_{i \in \mathbb{N}} x^{3i} \end{pmatrix} \end{aligned}$$

Analog liefern die Matrizen $\tilde{P}$, $\tilde{B}$, $\tilde{Q}$ und $\tilde{D}$ die Response-Matrix

$$\tilde{G}(x) = \begin{pmatrix} 1+x+x^2 & 1+x^2 \end{pmatrix}$$

Der praktische Vorteil, die Komponenten von $G(x)$ als bestimmte rationale Funktionen aufzufassen, besteht darin, daß die Multiplikation einer Potenzreihe $u(x) \in F[\![x]\!]$ mit $G_{i,j}(x)$ in diesem Fall mittels einer Schieberegistermaschine realisiert werden kann, deren Komponenten praktisch direkt aus den Polynomen $\omega_{i,j}(x)$ bzw. $\sigma_{i,j}(x)$ ablesbar sind, vergl. Abschnitt 2.3.3. $k \times n$ dieser Maschinen werden dann als Matrix verschaltet, die k Eingabe-Ströme jeweils in n Ausgabe-Ströme umwandelt, die dann in n-Tupeln zusammengefaßt und addiert werden.

Man kann die Komplexität senken, indem man in jeder Zeie $i < k$ die rationalen Funktionen $\langle \omega_{i,j}(x), \sigma_{i,j}(x) \rangle$, mittels eines gemeinsamen co-normierten Nenners $\sigma_i(x)$ ausdrückt. Dann kommt man pro Zeile mit einer Maschine aus, deren unterer Teil die Division durch $\sigma_i(x)$ realisiert, während im oberen Teil die Multiplikationen mit geeigneten Vielfachen von $\omega_{i,j}(x)$, $j < n$, parallel erfolgen.

Da $G(x)$ wie eine Generatormatrix funktioniert, und zwar sogar als injektive lineare Abbildung von $F(\!(x)\!)^k$ nach $F(\!(x)\!)^n$, ist es für den erzeugten *Code* irrelevant, ob wir jede Zeile mit $\sigma_i(x)$ multiplizieren und somit eine Matrix aus $\boldsymbol{Mat}_{F[x]}\langle k, n\rangle$ erhalten. Auf der Seite der Eingaben entspricht dies einer Division der Komponente i durch $\sigma_i(x)$, $i < k$.

Literatur

[1] Ferret, S., and Storme, L. Minihypers and linear codes meeting the griesmer bound: Improvements to results of hamada, helleseth and maekawa. *Designs, Codes and Cryptography 25*, 2 (Feb 2002), 143–162.

[2] Griesmer, H. J. A bound for error-correcting codes. *IBM Journal of Res. and Dev 4*, 5 (1960), 532–542.

[3] MacKay, D. J. C. *Information Theory, Inference and Learning Algorighms.* Cambridge University Press, 2003.

[4] Plotkin, M. Binary codes with specified minimum distance. *IRE Transactions on Information Theory*, 6 (1960), 445–450.

[5] Roth, R. M. *Introduction to Coding Theory.* Cambridge University Press, 2006.

[6] Singleton, R. C. Maximum distance q-nary codes. *IEEE Transactions on Information Theory 10*, 2 (apr 1964), 116–118.

Index

Jürgen Koslowski (koslowj at iti dot cs dot tu dash bs dot de)
Institut für Theoretische Informatik
TU Braunschweig
Mühlenpfordstr. 23
D-38106 Braunschweig
Germany