

# fehlerkorrigierende Codes

Thomas Irgang  
Katholische Universität Eichsätt

28. Februar 2007

# Inhaltsverzeichnis

<b>1</b>	<b>Was sind fehlerkorrigierende Codes?</b>	<b>3</b>
1.1	Was fehlerkorrigierende Codes nicht sind! . . . . .	3
1.2	Definition fehlerkorrigierender Codes . . . . .	3
1.3	Wozu braucht man fehlerkorrigierende Codes . . . . .	3
1.4	Modellvorstellung . . . . .	3
1.5	Problemstellung . . . . .	3
1.5.1	Die Lösung . . . . .	4
1.5.2	Beispiel . . . . .	4
1.6	Hammingabstand . . . . .	4
1.7	Aufgaben . . . . .	4
1.7.1	Lösung zu den Aufgaben . . . . .	5
1.8	Minimalabstand und Fehlereigenschaften . . . . .	5
<b>2</b>	<b>Mathematik und fehlerkorrigierende Codes</b>	<b>5</b>
2.1	Was hat das mit Mathe zu tun? . . . . .	5
2.2	Vektoren und Codes . . . . .	6
2.3	lineare Codes . . . . .	6
2.4	Matrizenmultiplikation . . . . .	6
2.5	ein Beispiel . . . . .	6
2.6	Minimalgewicht . . . . .	7
2.7	Aufgabe . . . . .	7
2.7.1	Lösung zu den Aufgaben . . . . .	8
2.8	Aber was ist mit decodieren? . . . . .	8
2.9	der duale Code . . . . .	8
2.10	die Kontrollmatrix . . . . .	8
2.11	Syndromdecodierung . . . . .	8
2.12	der Decodieralgorithmus . . . . .	9
<b>3</b>	<b>ein vollständiges Beispiel</b>	<b>9</b>
<b>4</b>	<b>Beweise und Ergänzungen</b>	<b>11</b>
4.1	Der Körper $\mathbb{Z}_2$ . . . . .	11
4.2	Vektorraum über $\mathbb{Z}_2$ . . . . .	12
4.3	$d(x) = w(x)$ . . . . .	12
4.4	Nebenklassen und Syndrome . . . . .	13
<b>5</b>	<b>Literatur</b>	<b>14</b>

# 1 Was sind fehlerkorrigierende Codes?

## 1.1 Was fehlerkorrigierende Codes nicht sind!

Fehlerkorrigierende Codes haben *nichts* mit Verschlüsselung zu tun! Ganz im Gegenteil. Es geht darum Datenverlust zu vermeiden. Die codierten Daten sollen trotz Fehlern bei der Übertragung/Speicherung lesbar bleiben.

## 1.2 Definition fehlerkorrigierender Codes

Fehlerkorrigierende Codes sind Codes, die es ermöglichen trotz Fehlern, die bei der Übertragung oder Speicherung von Daten auftreten, die Daten wieder herzustellen.

## 1.3 Wozu braucht man fehlerkorrigierende Codes

Auch, oder gerade im Zeitalter der digitalen Datenverarbeitung passieren ständig Fehler. Zum Beispiel:

- werden CDs verkratzt
- entstehen Fehler bei der Übertragung von Daten auf Leitungen
- kommen Funksignale “gestört” an

## 1.4 Modellvorstellung

Die Grundsituatuion sieht folgendermaßen aus:

Ein Sender will einem Empfänger eine Nachricht schicken. Dazu wird die Nachricht in Codewörtern codiert. Diese codierte Nachricht wird an den Empfänger übermittelt, wobei bei der Übertragung jedoch Fehler auftreten. Der Empfänger versucht die veränderte Nachricht beim decodieren wieder herzustellen.

## 1.5 Problemstellung

Wie kann man an Hand der übertragenen Nachricht einfach erkennen, ob ein Fehler beim Übertragen passiert ist?

Eine einfache Möglichkeit ist z.B. ein Prüfbit. Dabei wird vereinbart, dass die Anzahl der Einsen immer gerade/ungerade sein muss ( “gerade/ungerade Parität” ). Ein Bitumkehrfehler kann damit erkannt werden. Falls ein Fehler erkannt wird fordert der Empfänger die Nachricht nochmal an.

Das Problem ist damit jedoch nicht gelöst! Was ist wenn die Übertragung so schlecht ist, dass bei fast jedem Datenpaket fehler auftreten? Was ist wenn die Nachricht nicht einfach nochmal geschickt werden kann, z.B. weil die Datenquelle eine verkratzte CD ist?

Man braucht also Verfahren, die die Daten so kodieren, dass eine begrenzte Anzahl von Fehlern korrigiert werden kann. Wie kann man das erreichen?

### 1.5.1 Die Lösung

Wenn die Codewörter sich paarweise in mehreren Bits unterscheiden kann man sich diesen Unterschied als "Abstand" der Codewörter vorstellen. Es gibt also "freie" Bereiche zwischen den Codewörtern, damit gibt es eine Umgebung um jedes Codewort in dem jedes Wort darin minimalen Abstand zu diesem Codewort hat. Wenn wir also ein Wort empfangen können wir es dem Codewort zuordnen von dem es den geringsten Abstand hat. An einem Beispiel wird das sofort klar.

### 1.5.2 Beispiel

Gegeben sind die Codewörter  $a, b \in C \subseteq \{0, 1\}^4$  (aus dem Code  $C$ , der aus 4-Tupeln über  $\{0, 1\}$  besteht),  $a := 0011$ ,  $b := 1110$

a und b unterscheiden sich in:  $\left. \begin{array}{l} 0011 \\ 1110 \end{array} \right\}$  drei Stellen.

Wenn wir jetzt ein verändertes Codewort  $x$  empfangen, z.B.  $x = 0110$ , können wir es mit  $a$  und  $b$  vergleichen

a und x:  $\left. \begin{array}{l} 0011 \\ 0110 \end{array} \right\}$  unterscheiden sich in zwei Stellen.

b und x:  $\left. \begin{array}{l} 1110 \\ 0110 \end{array} \right\}$  unterscheiden sich in einer Stelle.

Damit können wir unter der Annahme einer minimalen Fehlerzahl  $x$  als  $b$  decodieren.

## 1.6 Hammingabstand

Die Anzahl der Stellen in denen sich zwei Codewörter unterscheiden bezeichnet man als *Hammingabstand*. Mathematisch sieht das so aus:

$$u, v \in C \subseteq \{0, 1\}^n$$

$$d(u, v) = |\{i : u_i \neq v_i\}|$$

Der minimale Hammingabstand eines Codes ist definiert als:

$$d(C) := \min\{d(u, v) : u, v \in C\}$$

Also der minimale Abstand den zwei Codewörter haben.

## 1.7 Aufgaben

Definiere:  $c : A \rightarrow C$ ,  $A$  Alphabet,  $C$  Code, d.h.  $c(5)$  ist das Codewort von 5  
 $c(0)=00011$ ,  $c(1)=00110$ ,  $c(2)=01100$ ,  $c(3)=11000$

1. Bestimme den minimalen Hamming-Abstand der Codewörter  $c(0)$ ,  $c(1)$ ,  $c(2)$ ,  $c(3)$ !
2. Kann man das veränderte Codewort  $x=00010$  eindeutig decodieren?

- Beschreiben Sie den Zusammenhang zwischen dem minimalen Hamming-Abstand eines Codes und seiner Eigenschaften bezüglich des Erkennens und Korrigierens von Fehlern?

### 1.7.1 Lösung zu den Aufgaben

1.

	00011	00110	01100	11000
00011	–	2	4	4
00110	–	–	2	4
01100	–	–	–	2
11000	–	–	–	–

⇒ der Minimalabstand der Codewörter ist 2.

- Nein, das Wort hat z.B. Abstand 1 sowohl von 0 als auch von 1 und ist kein gültiges Codewort. Daraus folgt:
- Ein Minimalabstand von 2 reicht aus um einen Fehler zu erkennen (man muss mindestens 2 Bit ändern um wieder ein gültiges Codewort zu erhalten), das veränderte Codewort kann aber i. A. nicht eindeutig einem Codewort zugeordnet werden.

Allgemein folgt daraus:

## 1.8 Minimalabstand und Fehlereigenschaften

Ein Code  $C \subseteq \{0,1\}^n$  heißt ein *t-fehlerkorrigierender* Code falls der Minimalabstand zwischen zwei Codewörtern  $\geq 2t + 1$  ist. Ein solcher Code kann  $2t$  Fehler erkennen. Es gibt also um jedes Codewort eine Umgebung mit Radius  $t$  in der in der jedes Wort dem Codewort zugeordnet werden kann.

## 2 Mathematik und fehlerkorrigierende Codes

### 2.1 Was hat das mit Mathe zu tun?

Was hat das ganze denn nun überhaupt mit Mathe zu tun?

Wir müssen irgendwie "effektiv" mit dem Code umgehen können! Es wäre sehr mühselig bzw. rechenaufwendig jedes empfangene Codewort mit jedem gültigen Codewort zu vergleichen, bzw. den Mindestabstand mit jedem Codewort zu bestimmen nur um es zu decodieren. An dieser Stelle kommt die Mathematik, bzw. viel mehr die lineare Algebra, ins Spiel.

## 2.2 Vektoren und Codes

Ihr kennt Vektoren aus dem Matheunterricht. Man kann aber nicht nur Vektoren über  $\mathbb{R}$  betrachten, sondern über jedem Körper. Die beiden Elemente 0 und 1 bilden auch einen Körper (Beweis siehe 4.1) und unsere Codewörter sind Vektoren über diesem Körper (Beweis siehe 4.2).

Wenn die Codewörter alle in einem Unterraum des Vektorraums liegen - ein solcher Code heißt linearer Code - bringt das eine Reihe von Vorteilen mit sich.

## 2.3 lineare Codes

Ein linearer Code kann:

- mit Hilfe einer Basis erzeugt werden und
- mit Hilfe von Matrizen “verarbeitet” werden.

Verarbeitet bedeutet in diesem Zusammenhang dass Daten codiert, decodiert und Fehler korrigiert werden.

## 2.4 Matrizenmultiplikation

Im folgenden brauchen wir die Multiplikation von Vektoren mit Matrizen. Diese Multiplikation ist allgemein definiert durch:

$$\begin{pmatrix} a_{11} & \dots & a_{1n} \\ a_{21} & \dots & a_{2n} \\ \dots & \dots & \dots \\ a_{m1} & \dots & a_{mn} \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ v_2 \\ \dots \\ v_n \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^n a_{1i} \cdot v_i \\ \sum_{i=1}^n a_{2i} \cdot v_i \\ \dots \\ \sum_{i=1}^n a_{mi} \cdot v_i \end{pmatrix}$$

(also: Zeile  $\cdot$  Spalte)

Ein Beispiel dazu:

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \cdot 1 + 0 \cdot 1 + 0 \cdot 0 \\ 1 \cdot 1 + 1 \cdot 1 + 0 \cdot 0 \\ 0 \cdot 1 + 1 \cdot 1 + 1 \cdot 0 \\ 0 \cdot 1 + 0 \cdot 1 + 0 \cdot 0 \\ 0 \cdot 1 + 0 \cdot 1 + 0 \cdot 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

## 2.5 ein Beispiel

Die Matrix  $G := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$  (G für Generatormatrix) erzeugt einen linearen 1-

fehlerkorrigierenden Code. Die Spalten der Matrix spannen dabei einen linearen Unter-

raum auf in dem die Codewörter liegen. Wenn jetzt z.B. das Datenwort  $x := \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$

codiert werden soll rechnet man einfach  $G \cdot x = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$ . Dadurch muss man statt 16

(=  $2^4$ ) Codewörtern nur 4 speichern. Damit können wir Daten effektiv codieren und Codes effektiv speichern. Um auch effektiv decodieren zu können brauchen wir noch etwas mehr Mathematik.

## 2.6 Minimalgewicht

Als *Gewicht* eines Vektors wird seine Anzahl der von 0 verschiedenen Stellen bezeichnet. Formal sieht die Definition so aus:

$$w(c) := |\{i \mid c_i \neq 0\}|, c \in C$$

Das *Minimalgewicht*  $w(C)$  des Codes  $C$  definiert man dann als:

$$w(C) := \min\{w(c) \mid c \in C, c \neq 0\}$$

Weiter ergibt sich, dass  $d(C) = w(C)$ . (Beweis siehe 4.3). Man muss also maximal  $|C|$  Codewörter betrachten anstatt  $|C|$  Codewörter paarweise miteinander zu vergleichen.

*Achtung:* Dies vereinfacht das erkennen von Fehlerkorrektoreigenschaften erheblich, man muss aber dennoch alle Codewörter betrachten, denn  $a := 011100$  und  $b := 111000$  haben beide Gewicht 3, der lineare Code für den diese eine Basis sind enthält aber auch  $a + b$ , also  $\left. \begin{array}{r} 111000 \\ + 011100 \end{array} \right\} = 100100$ , hat damit ein Minimalgewicht  $\leq 2$  und kann somit keine Fehler korrigieren.

## 2.7 Aufgabe

Wie viele Fehler kann der folgende lineare Code erkennen/korrigieren?

0000000	0001110	0010111	0011001
0100101	0101011	0110010	0111100
1111111	1110001	1101000	1100110
1011010	1010100	1001101	1000011

### 2.7.1 Lösung zu den Aufgaben

Der Code hat Minimalgewicht 3  $\Rightarrow$  er kann einen Fehler korrigieren und 3 Fehler erkennen. Dies ist der Code der von der obigen Generatormatrix erzeugt wird.

## 2.8 Aber was ist mit decodieren?

Wir können jetzt also einfach Datenwörter codieren, bzw. alle  $2^k$  Codewörter aus  $k$  Basisvektoren erzeugen und einen Code schnell einschätzen. Aber wie können wir einfach fehlerkorrigieren und die Codewörter decodieren? Dazu brauchen wir, wie soll es auch anders sein, mehr Theorie.

## 2.9 der duale Code

Um einen linearen Code effektiv zu decodieren brauchen wir den dualen Code. Sei  $C$  ein Code, der zu  $C$  *duale Code*  $C^\perp$  ist definiert durch:

$$C^\perp := \{v \in V \mid c \cdot v = 0 \forall c \in C\}$$

dabei bedeutet  $c \cdot v = \sum_{i=1}^n c_i \cdot v_i$ .

*Achtung:* Bei dieser Definition gibt es ein paar merkwürdige Phänomene!  $C^\perp$  und  $C$  können teilweise übereinstimmen. Dies ist eine Folge des endlichen Körpers  $\mathbb{Z}_2$  und spielt hier im weiteren keine Rolle, soll aber niemanden verwirren. Trotz allem gilt aber:

$$C^{\perp\perp} = C$$

## 2.10 die Kontrollmatrix

Wenn wir den dualen Code haben, haben wir auch die Kontrollmatrix und sind damit wieder einen Schritt näher am Ziel. Eine Matrix deren Zeilen eine Basis des dualen Codes sind nennt man *Kontrollmatrix* des linearen Codes, das Produkt  $H \cdot s$  nennt man das *Syndrom*. Mit Hilfe des Syndroms können wir nun endlich einen linearen Code effektiv decodieren. Da die Basis des dualen Codes Teilmenge des Codes ist gilt natürlich  $s(c) = 0 \forall c \in C$ .

## 2.11 Syndromdecodierung

Weiterhin gilt  $s(v) = s(w) \Leftrightarrow v + C = w + C$  (siehe 4.4)

Das bedeutet dass das Syndrom eines Vektors nur von der Nebenklasse abhängt in der der Vektor liegt.

Als *Anführer* einer Nebenklasse wird der Vektor mit dem kleinsten Gewicht bezeichnet. Diese Nebenklassenführer sind eindeutig, d.h. jede Nebenklasse hat genau einen Anführer. (siehe 4.4)

## 2.12 der Decodieralgorithmus

Damit kann man, mit Hilfe einer Liste der Nebenklassenanhänger folgendermaßen vorgehen:

1. Syndrom vom empfangen Wort  $x$  berechnen
2. Nebenklassenanhänger  $e$  feststellen
3. Codewort  $c$  durch  $c = x + e$  berechnen

## 3 ein vollständiges Beispiel

Diesen Algorithmus können wir jetzt mal an einem Beispiel durchspielen. Wir benutzen

wieder die Generatormatrix  $G :=$

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

Als erstes brauchen wir wieder eine Kontrollmatrix, also brauchen wir den dualen Code. Wir brauchen also drei linear unabhängige Vektoren die das Gleichungssystem

$$\begin{aligned} (1) \quad c_1 + c_6 + c_7 &= 0 \\ (2) \quad c_2 + c_5 + c_7 &= 0 \\ (3) \quad c_3 + c_5 + c_6 + c_7 &= 0 \\ (4) \quad c_4 + c_5 + c_6 &= 0 \end{aligned}$$

erfüllen. Dies ist z.B. bei den Vektoren  $\begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}$ ,  $\begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}$ ,  $\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}$  der Fall, die Matrix

$$K := \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

ist also eine Kontrollmatrix zu  $G$ . Jetzt können wir die Nebenklassenanhänger berechnen. Dies erreichen wir in dem wir indem wir die Syndrome

der Einheitsvektoren berechnen (siehe 4.4). Dadurch erhalten wir:

```

0000000 000
0000001 111
0000010 011
0000100 101
0001000 110
0010000 001
0100000 010
1000000 100

```

Jetzt müssen wir nur noch die die Codewörter ihren Datenwörtern zuordnen. Das ist aber einfach, denn wenn wir die Generatormatrix betrachten sehen wir dass die ersten 4 Bit des Codeworts das Datenwort sind. Es reicht also die letzten drei Bit abzuschneiden. Jetzt haben wir alles um den Algorithmus anzuwenden. Nehmen wir uns ein Datenwort z.B.  $x = 0110$ . Als erstes codieren wir das Datenwort durch Multiplikation mit der Matrix.

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

Wenn jetzt bei der Übertragung ein Fehler passiert und aus dem Codewort z.B.

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

wird ergibt die Multiplikation mit der Kontrollmatrix

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1+0+0+0+0+0+0 \\ 0+1+0+0+0+1+0 \\ 0+0+1+0+0+1+0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

Also ist der Fehlervektor  $\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$  und  $\begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$ , das Datenwort ist demnach  $\begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$  was korrekt ist. Die Korrektur funktioniert also.

## 4 Beweise und Ergänzungen

### 4.1 Der Körper $\mathbb{Z}_2$

Um zu zeigen dass die Elemente  $\{0, 1\}$  einen Körper bilden muss man zeigen dass die beiden Elemente die Körperaxiome erfüllen. Dazu müssen wir zuerst die Addition und die Multiplikation definieren. Dies erfolgt mit Hilfe der Modulo-Rechnung. Was ist Modulo-Rechnung?

Einfach ausgedrückt ist  $a \bmod b$  der Rest der ganzzahligen Division, d.h.  $15 : 6 = 2 \text{ Rest } 3$ , dann ist  $15 \bmod 6 = 3$  (in vielen Programmiersprachen der `%`-Operator).

Im  $\mathbb{Z}_2$  ist die Addition durch

$$a +_2 b := (a + b) \bmod 2$$

definiert und die Multiplikation durch

$$a \cdot_2 b := (a \cdot b) \bmod 2$$

definiert. Mit diesem Vorwissen sind die Körperaxiome einfach zu zeigen.

**Kommutativgesetz**  $a + b = (a + b) \bmod 2 = (b + a) \bmod 2 = b + a$   
 $a \cdot b = (a \cdot b) \bmod 2 = (b \cdot a) \bmod 2 = b \cdot a$

**Assoziativgesetz**  $(a + b) + c = ((a + b) \bmod 2) + c \bmod 2 = (a + b + c) \bmod 2 =$   
 $(a + ((b + c) \bmod 2)) \bmod 2 = a + (b + c)$   
 $(a \cdot b) \cdot c = ((a \cdot b) \bmod 2) \cdot c \bmod 2 = (a \cdot b \cdot c) \bmod 2 = (a \cdot ((b \cdot c) \bmod 2)) \bmod 2 = a \cdot (b \cdot c)$

**neutrales Element der Addition**  $a +_2 0 = (a + 0) \bmod 2 = a \bmod 2 = a$ , d.h. die "übliche" 0 ist auch das Null-Element in  $\mathbb{Z}_2$ .

**neutrales Element der Multiplikation** Analog zur Addition folgt:  $a \cdot_2 1 = (a \cdot 1) \bmod 2 = a \bmod 2 = a$ , d.h. die "übliche" 1 ist auch das Eins-Element in  $\mathbb{Z}_2$ .

**eindeutige Lösbarkeit der Gleichung  $a \cdot x = 1$  für  $a \neq 0$**  Im  $\text{mathds}Z_2$  gibt es genau ein Element  $\neq 0$ , die 1, und  $1 \cdot 1 = 1$ , d.h. die 1 ist ihr eigenes inverses Element.

**Distributivgesetz**  $a \cdot_2 (b +_2 c) = a \cdot_2 ((b + c) \bmod 2) = (a \cdot (b + c)) \bmod 2 = (a \cdot b + a \cdot c) \bmod 2 = (((a \cdot b) \bmod 2) + ((a \cdot c) \bmod 2)) \bmod 2 = a \cdot_2 b +_2 a \cdot_2 c$

Damit ist gezeigt dass  $Z_2$  ein Körper ist.

## 4.2 Vektorraum über $Z_2$

Es reicht zu zeigen, dass  $Z_2^n$  bezüglich der Addition und der Multiplikation mit einem Körperelement abgeschlossen ist, d.h. wenn  $a, b \in Z_2^n$  dann ist auch  $\alpha \cdot a + \beta \cdot b \in Z_2^n$ .

$a, b \in Z_2^n$ , also  $a = \begin{pmatrix} a_1 \\ a_2 \\ \dots \\ a_n \end{pmatrix}$ , dann ist  $a + b$  definiert als  $\begin{pmatrix} a_1 + b_1 \\ a_2 + b_2 \\ \dots \\ a_n + b_n \end{pmatrix}$ . Da  $Z_2$  ein Körper

ist und  $a_i, b_i \in Z_2$ , dass  $a_i + b_i \in Z_2$ . Damit folgt  $a + b \in Z_2^n$ .  $Z_2^n$  ist also bezüglich der Addition abgeschlossen. Man muss noch zeigen dass  $Z_2^n$  bezüglich der Multiplikation abgeschlossen ist. Das funktioniert ähnlich:

$\alpha \cdot \begin{pmatrix} a_1 \\ a_2 \\ \dots \\ a_n \end{pmatrix} = \begin{pmatrix} \alpha \cdot a_1 \\ \alpha \cdot a_2 \\ \dots \\ \alpha \cdot a_n \end{pmatrix}$ . Da  $Z_2$  ein Körper ist folgt wieder  $\alpha \cdot a \in Z_2^n$ .

Damit ist eigentlich nur gezeigt dass  $Z_2^n$  ein Unterraum ist, dies ist in diesem Fall aber ausreichend.

## 4.3 $d(x) = w(x)$

Im folgenden seien  $u, v, w \in C$ .

Im folgenden braucht man dass  $d(\cdot, \cdot)$  eine Metrik ist, d.h.

1.  $d(u, v) > 0$  für  $u \neq v$  und  $d(u, u) = 0$
2.  $d(u, v) = d(v, u)$
3.  $d(u, w) \leq d(u, v) + d(v, w)$

Dann mal los.

1. ist klar
2. ist auch klar
3. Wir können annehmen dass sich  $u$  und  $w$  genau an den ersten  $t$  Stellen unterscheiden.

Unter den ersten  $t$  Stellen gibt es dann  $r$  Stellen an denen sich  $u$  und  $v$  unterscheiden ( $r \in \{0, \dots, t\}$ ), das heißt aber  $v$  und  $w$  unterscheiden sich in  $t - r$  Stellen. Weiter unterscheiden sich  $u$  und  $v$  in den letzten  $n - t$  Stellen in  $s$  Stellen. Dann unterscheiden sich  $v$  und  $w$  ebenfalls in  $s$  Stellen. Insgesamt folgt dann:  $d(u, v) = r + s$  und  $d(w, v) = t - r + s$ , also  $d(u, v) + d(w, v) = r + s + t - r + s = t + 2s$  und damit gilt

$$d(u, w) \leq d(u, v) + d(w, v)$$

da  $s \geq 0$ .

Damit folgt dann aber

$$d(C) = \min\{d(c, c') \mid c, c' \in C, c \neq c'\} \leq \min\{d(c, 0) \mid c \in C, c \neq 0\} = w(C)$$

Also der Minimalabstand ist kleiner als das Minimalgewicht des Codes. Wenn wir jetzt noch zeigen dass es ein Codewort vom Gewicht  $d(C)$  gibt ist die Gleichheit gezeigt. Seien also  $c, c' \in C$  mit  $d(c, c') = d(C)$ . Es gilt  $w(c - c') = d(c - c', 0) = d(c, c') = d(C)$ . Da  $C$  ein linearer Code ist, ist  $c - c' \in C$ . Damit haben wir die Gleichheit gezeigt. Für lineare Codes (und nur für lineare Codes) gilt also  $d(C) = w(C)$

## 4.4 Nebenklassen und Syndrome

Vorweg:  $v + C := \{v + c \mid c \in C\}$ . Weiter ist  $s(\cdot)$  eine lineare Abbildung, d.h.  $s(v + c) = s(v) + s(c)$

Schauen wir uns jetzt die Aussage an:

$$s(v) = s(w), \quad v, w \in \mathbb{Z}_2^n$$

$$\Leftrightarrow s(v) - s(w) = 0$$

$$\Leftrightarrow s(v - w) = 0$$

$$\Leftrightarrow v - w \in C$$

$$\Leftrightarrow v + C = w + C$$

Nun zur Eindeutigkeit.

*Satz über die Eindeutigkeit der Nebenklassenanhänger*

Sei  $C$  ein linearer  $t$ -fehlerkorrigierender Code. Dann gilt:

1. Jeder Vektor von  $V$  vom Gewicht  $\leq t$  ist ein Nebenklassenanhänger.
2. Die Anhänger von Nebenklassen die einen Vektor vom Gewicht  $\leq t$  enthalten sind eindeutig bestimmt.

Beweis:

Sei  $v$  ein Vektor vom Gewicht  $\leq t$ . Betrachten wir nun einen beliebigen Vektor  $v' \in v + C \Leftrightarrow \exists c \in C$  mit  $v' = v + c$ . Sei weiter  $v' \neq v$ , d.h.  $c \neq 0$ . Wir müssen jetzt zeigen, dass  $w(v') \geq t + 1$  ist. Wenn wir das gezeigt haben wissen wir auch, da  $v'$  ein beliebiger Vektor aus dieser Nebenklasse war, dass alle Vektoren dieser Nebenklasse  $\neq v$  ein Gewicht von  $\geq t + 1$  haben.

Da  $v$  und  $v'$  in der selben Nebenklasse sind ist  $v - v' \in C$ , also  $v - v'$  ist ein Codewort.

Aber  $v \neq v'$  und damit  $v - v' \neq 0$ . Deshalb gilt  $w(v - v') \geq 2t + 1$ , da  $C$  ein  $t$ -fehlerkorrigierender Code ist. Damit folgt:

$$2t + 1 \leq w(v - v') = d(v - v', 0) = d(v, v') \leq d(v, 0) + d(v', 0) = w(v) + w(v') \leq t + w(v')$$

also haben wir  $2t + 1 \leq t + w(v') \Leftrightarrow t + 1 \leq w(v')$  und damit ist der Satz gezeigt.

## 5 Literatur

Albrecht Beutelspacher: Lineare Algebra, Vieweg, 6.Auflage, Wiesbaden 2003