Kanalcodierung

Leitprogramm

Peter Friedl und Patrizia Mottl

Inhalt:
Nach Abschluss des Leitprogrammes kennen die Studierenden die Struktur von Informationssystemen. Sie können Nachrichten mit Hilfe von unterschiedlichen Codetypen codieren und decodieren und gegebenenfalls Fehler erkennen und beheben..

Unterrichtsmethode: Leitprogramm

Fachliches Review:
Prof. Dr. M. Meyer, Leiter SG Elektrotechnik FH Nordwestschweiz

Fachdidaktisches Review:
Prof. Dr. A. Glattfelder, Institut für Automatik, ETH Zürich

Publiziert auf EducETH:
24. August 2009

Fach-Didaktik

Informationstechnologie und Elektrotechnik

Kanalcodierung

Leitprogramm

Fach	Kommunikationstechnik / Nachrichtentechnik
Schultyp	Technische Hochschule
Voraussetzungen	keine
Bearbeitungsdauer	7 - 8 Unterrichtslektionen (à 45 Min.)
Autoren	Peter Friedli, Patrizia Mottl
Betreuer	Prof. Martin Meyer, Windisch
Fassung	Oktober 06, (EDUCETH 2009)
Unterrichtserprobung	erprobt

Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Departement D-GESS
Institut für Verhaltenswissenschaften

Leitprogramm zum Thema Kanalcodierung

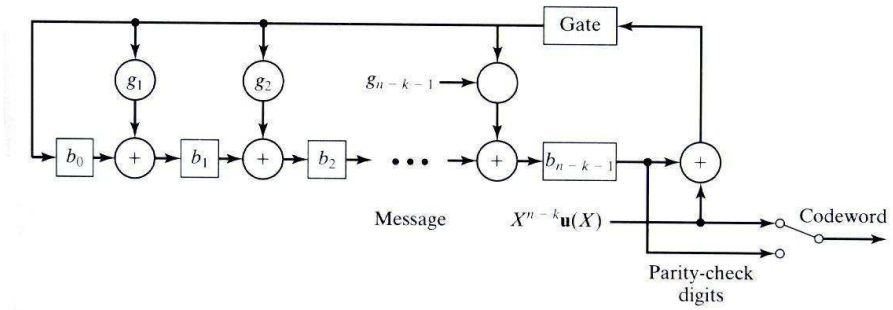

Abbildung 1: aus [Lin04, S. 148]

Fach:	Kommunikations-/Nachrichtentechnik
Adressaten:	FH-StudentInnen
Vorkenntnisse:	keine
Bearbeitungsdauer:	7-8 Unterrichtslektionen
Autoren:	Peter Friedli, 02-924-538, D-ITET, Rosenweg 1, 5300 Turgi Patrizia Mottl, 03-919-594, D-ITET, Döbeligut 5, 4665 Oftringen
Betreuer:	Prof. Dr. M. Meyer, Leiter SG Elektrotechnik FH Nordwestschweiz
Fassung vom:	1. Oktober 2006
Praxisstatus:	nicht erprobt

Einführung

In der heutigen Welt ist Kommunikation nicht mehr wegzudenken. Sei es nun Sprache via Mobiltelefon, Fernsehsignale via Satellit oder Finanztransaktionen via Internet, überall spielt die **Informationsübertragung** eine entscheidende Rolle.

Störungen sind jedoch allgegenwärtig. Sie führen in vielen Fällen zu Fehlern. Die Fehlerwahrscheinlichkeit ist vom Übertragunskanal und der verwendeten Übertragungstechnik abhängig. Treten beispielsweise „nur“ 10^{-7} Fehler/s auf, würde bei $10MBits/s$ (Übertragungsrate eines Ethernet Segments) jede Sekunde ein Bitfehler auftreten. Dies führte je nach Anwendung zu einer ungenügenden Qualität der Übertragung. Dieses Beispiel verdeutlicht, dass Fehlererkennung und Fehlerkorrektur wichtig für die heutige Gesellschaft sind.

Die Kanalcodierung fügt der ursprünglichen Nachricht zusätzliche Daten - sogenannte **Redundanz** - zu. Im Verlaufe des Leitprogrammes werden Möglichkeiten aufgezeigt, Fehler zu erkennen und gegebenfalls zu korrigieren.

Nach Abschluss des Leitprogrammes kennen Sie die Struktur von Informationssystemen. Sie können Nachrichten mit Hilfe von unterschiedlichen Codetypen codieren und decodieren und gegebenenfalls Fehler erkennen und beheben.

Zu Beginn erhalten Sie eine kurze Arbeitsanleitung. In **Kapitel 1 und 4** erhalten Sie darauf das mathematische Grundwissen. **Kapitel 2** gibt einen Überblick über die Kanalcodierung und hält einige Definitionen fest. In **Kapitel 3** beschäftigen Sie sich dann genauer mit zyklischen Codes - einer Teilmenge der linearen Codes. Zu guter Letzt erarbeiten Sie - falls Ihnen noch Zeit bleibt - in **Kapitel 5** einen Vertreter aus den zyklischen Codes: den BCH Code.

Inhaltsverzeichnis

Arbeitsanleitung

Allgemein

Dieser Text gibt Ihnen eine Übersicht, wie Sie das Leitprogramm bearbeiten sollen. Die einzelnen Kapitel sind gleich aufgebaut.

1. Zu Beginn erhalten Sie jeweils eine kurzen Überblick über das Thema des Kapitels.
2. Gleich anschliessend werden Ihnen in einem Rahmen die Lernziele beschrieben. Die Lernziele sind in drei Teile unterteilt. Die Leitidee gibt Ihnen Antwort auf die Frage: „Wozu wurde das Thema ausgewählt?“. Das Dispositionsziel sagt aus, was Sie nach dem Kapitel grundsätzlich können und das operationalisierte Lernziel gibt an, was Sie schlussendlich konkret ausführen können. Die Lernziele dienen Ihnen zur Orientierung. Danach beginnt der eigentliche Text. Lesen Sie ihn in Ruhe durch und lösen Sie die eingestreuten Aufgaben. Sie geben Ihnen gleich von Anfang an die Möglichkeit, das Gelesene zu üben. Zur Korrektur Ihrer Lösungen, finden Sie jeweils im Anschluss ans Kapitel Lösungsvorschläge.
3. Am Ende finden Sie einige Aufgaben, die mit Lernkontrolle überschrieben sind. Mit diesen Aufgaben können Sie selbstständig prüfen, ob Sie die Lernziele erreicht haben. Sie finden gleich im Anschluss daran Lösungen dazu.
4. Wenn Sie das Kapitel durchgearbeitet haben und das Gefühl haben, Sie hätten die wesentlichen Lernziele erreicht, legen Sie beim Tutor einen Kapiteltest ab. Haben Sie ihn erfolgreich absolviert, dann können Sie mit dem nächsten Kapitel beginnen.

Symbole

Wir haben am linken Seitenrand Symbole stehen, die Ihnen klar zeigen, was Ihnen geboten wird. Wir geben Ihnen hier eine kurze Übersicht, was die jeweilige Bedeutung ist:

Hinweis Einen Hinweis oder eine wichtige Definition werden wir mit dieser Glühbirne kennzeichnen.

Bsp. 1 Beispiele werden wir mit nebenstehendem Symbol darstellen. Sie zeigen Ihnen, wie das entsprechende Wissen in Aufgaben verwendet werden kann.

Aufg. 2 Für Aufgaben, die Sie selber lösen sollen, verwenden wir das „Baustellen-Männlein“.

Ganz zum Schluss der Arbeitsanleitung geben wir Ihnen noch eine kurze Übersicht, wie lange Sie ungefähr für die einzelnen Kapitel durchschnittlich benötigen:

- Kapitel 1: eine - zwei Lektionen
- Kapitel 2: zwei Lektionen
- Kapitel 3: eine - zwei Lektion
- Kapitel 4: zwei - drei Lektionen
- Kapitel 5: eine - zwei Lektionen

Nun wünschen wir Ihnen viel Spass mit dem Leitprogramm „Kanalcodierung“.

Kapitel 1

Mathematische Grundlagen

1.1 Einführung und Lernziele

In diesem Kapitel geben wir Ihnen einen kurzen Überblick über die grundlegenden mathematischen Fertigkeiten, die Sie für die folgenden Kapitel besitzen müssen. Wir gehen davon aus, dass Sie schon ein Vorwissen mitbringen, dass es Ihnen erlaubt, das erste Kapitel zügig zu absolvieren. Wir gehen folgendermassen vor:

- In einem ersten Schritt betrachten wir anhand kleinerer Aufgaben verschiedene Operationen mit Matrizen und Vektoren.
- Anschliessend beschäftigen Sie sich mit binären Zahlen und deren Rechenregeln.
- Als Drittes lernen Sie, Polynome durch andere Polynome zu dividieren.

In diesem „Training“ erarbeiten Sie sich also das Wissen für die folgenden Kapitel. Auf der nächsten Seite finden Sie die detaillierten Lernziele. Studieren Sie diese. Beginnen Sie anschliessend gleich mit dem ersten Thema, der Matrizenrechnung.

LERNZIELE

Leitidee

Um Kommunikationssysteme behandeln zu können, bedarf es einer Darstellung der realen Gegebenheiten mit Hilfe mathematischer Modelle. Deshalb muss für das Verständnis der folgenden Kapitel eine mathematische Grundlage geschaffen werden.

Dispositionsziele

Sie sind sich beim Studium der folgenden Kapitel der mathematischen Grundlagen bewusst und können diese anwenden. Der Ihnen teilweise vielleicht schon bekannte Stoff in diesem Kapitel soll vertieft werden.

Operationalisierte Lernziele

Sie

- beherrschen grundlegende Operationen mit Matrizen und Vektoren
- können binäre Zahlen miteinander verrechnen
- können die Polynomdivision anwenden

1.2 Matrizenrechnung

1.2.1 Definitionen

Wir unterscheiden grundsätzlich zwischen folgenden zwei Formen:

- **Matrix**: Man nennt das folgende Gebilde eine Matrix der Dimension $m \times n$:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & & & & \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{pmatrix} \tag{1.1}$$

  Dabei ist a_{ij} das ij-te Element der Matrix.

- **Vektor**: Eine etwas schlankere Variante bezeichnet man als Vektor der Dimension m:

$$\begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{pmatrix} \tag{1.2}$$

Bsp. 1.1 Eine 2x3 Matrix A sei gegeben als

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 7 & 3 & 5 \end{pmatrix}$$

Das Element in der zweiten Zeile und der dritten Spalte bezeichnet man als a_{23}. Hier ist $a_{23} = 5$.

Bsp. 1.2 Ein Vektor der Dimension 3 ist folgender Vektor b:

$$b = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Ein Vektor der Dimension m ist also eine $m \times 1$ Matrix!

Rufen Sie sich nochmals in Erinnerung, wie man die Grössen von Matrizen bzw. Vektoren angibt. Merken Sie sich, welche Grösse (Anzahl Zeilen oder Anzahl Spalten) zuerst angeben wird. Sie ersparen sich so eine Menge an „Verwechslungspotential“!

1.2.2 Operationen

Da wir davon ausgehen, dass Sie die verschiedenen Rechnungsarten mit Matrizen beherrschen, geben wir Ihnen zu den jeweiligen Fällen nur je die Regel und ein Beispiel an:

Addition von Matrizen Die Matrizen werden elementweise zusammengezählt. Bei der Addition zweier Matrizen A und B addieren sich also a_{11} und b_{11}, etc.

$$\begin{pmatrix} 1 & 6 & 2 \\ 4 & 2 & 2 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 5 \\ 2 & 1 & 3 \end{pmatrix} = \begin{pmatrix} 1+0 & 6+0 & 2+5 \\ 4+2 & 2+1 & 2+3 \end{pmatrix} = \begin{pmatrix} 1 & 6 & 7 \\ 6 & 3 & 5 \end{pmatrix}$$

Skalarmultiplikation: Matrix mal Zahl: Jedes Element der Matrix wird mit der Zahl multipliziert.

$$4 \cdot \begin{pmatrix} 1 & 3 & 4 \\ 2 & 4 & 5 \end{pmatrix} = \begin{pmatrix} 4\cdot 1 & 4\cdot 3 & 4\cdot 4 \\ 4\cdot 2 & 4\cdot 4 & 4\cdot 5 \end{pmatrix} = \begin{pmatrix} 4 & 12 & 16 \\ 8 & 16 & 20 \end{pmatrix}$$

Matrixmultiplikation: Matrix mal Matrix: Hier müssen Sie jeweils Zeilen mit Spalten elementweise miteinander multiplizieren. Als Beispiel ergibt sich unten für $A \cdot B = C$ das Element c_{21} aus der elementweisen Multiplikation der zweiten Zeile aus A mit der ersten Spalte von B.

$$\begin{pmatrix} 1 & 3 & 2 \\ 8 & 6 & 4 \end{pmatrix} \cdot \begin{pmatrix} 2 & -3 \\ 4 & 5 \\ -1 & 2 \end{pmatrix} = \begin{pmatrix} 1\cdot 2+3\cdot 4+2\cdot(-1) & 1\cdot(-3)+3\cdot 5+2\cdot 2) \\ 8\cdot 2+6\cdot 4+4\cdot(-1) & 8\cdot(-3)+6\cdot 5+4\cdot 2) \end{pmatrix} = \begin{pmatrix} 12 & 16 \\ 36 & 14 \end{pmatrix}$$

Multiplikation: Matrix mal Vektor Entspricht in einer vereinfachten Form der obigen Regel.

$$\begin{pmatrix} 1 & 3 & 2 \\ 8 & 6 & 4 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 4 \\ -1 \end{pmatrix} = \begin{pmatrix} 1\cdot 2+3\cdot 4+2\cdot(-1) \\ 8\cdot 2+6\cdot 4+4\cdot(-1) \end{pmatrix} = \begin{pmatrix} 12 \\ 36 \end{pmatrix}$$

Transponieren einer Matrix Die Elemente der Matrix werden an der Diagonalen gespiegelt. Man bezeichnet diese Operation mit einem hochgestelltem ^{T}.

$$\begin{pmatrix} 1 & 3 & -2 \\ 8 & -6 & 4 \end{pmatrix}^T = \begin{pmatrix} 1 & 8 \\ 3 & -6 \\ -2 & 4 \end{pmatrix}$$

Wichtig ist bei all diesen Rechnungen, dass **die Grössen der Matrizen aufeinander passen**! Was heisst das für die Multiplikation zweier Matrizen? Genau, die Dimensionen der beiden Matrizen müssen gespiegelt sein. Im Beispiel hatte die erste Matrix die Dimension 2×3 und die zweite Matrix die Dimension 3×2!

Falls Sie sich bei einem dieser Beispiele unsicher fühlen, schlagen Sie in Ihrem Mathematikbuch oder auf [WikiM06] nach, oder fragen Sie Ihren Tutor!

Lösen Sie nun zur Festigung Ihrer Kenntnisse die folgende Aufgabe:

A1.1 Gegeben sind die folgenden Matrizen

$$A = \begin{pmatrix} 3 & 2 & 3 & 0 \\ 0 & -1 & 0 & -1 \\ 4 & 0 & -2 & 0 \end{pmatrix} \quad B = \begin{pmatrix} 1 & 0 \\ 1 & -2 \\ 1 & -3 \\ 0 & -1 \end{pmatrix} \quad C = \begin{pmatrix} 1 \\ -1 \\ 1 \\ 0 \end{pmatrix}$$

$$D = \begin{pmatrix} 1 & 0 \\ 0 & 3 \end{pmatrix} \quad E = \begin{pmatrix} 2 & 1 \\ 1 & 3 \end{pmatrix}$$

Berechnen Sie:

a) AB

b) $3C^T B$

c) $B \cdot (D + E)$

1.3 Polynomdivision

Bei der Beschreibung von Codewörtern und Codierungssystemen werden wir Polynome verwenden.

Definition: Polynome sind Konstrukte der Art

$$p(x) = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \ldots + a_2 \cdot x^2 + a_1 \cdot x + a_0 \tag{1.3}$$

wobei a_i der Koeffizient zur Potenz x^i ist. Die Koeffizienten sind in unserem Fall reelle Zahlen.

Als **Ordnung** bzw. **Grad** eines Polynom bezeichnet man dessen grösste Potenz. Das Polynom in (1.3) ist also vom Grad n.

Wie bei der Division von reellen Zahlen definiert man die **Division zweier Polynome** $p(x)$ und $q(x)$ als

$$d(x) = \frac{p(x)}{q(x)} \tag{1.4}$$

Diese Rechnung bezeichnet man sinnigerweise als **Polynomdivision**. Dabei müssen $p(x)$ und $q(x)$ nicht den gleichen Grad haben.

Machen wir dazu doch gleich ein Beispiel! Sie finden anschliessend daran die allgemeine Vorgehensweise. Versuchen Sie, die dort angegebenenen Schritte im Beispiel nachzuvollziehen.

Bsp. 1.3 Gegeben sind die beiden Polynome

$$p(x) = 3/4x^3 + 19/4x^2 + 4x + 1 \qquad q(x) = 3/4x^2 + 4x$$

Gesucht ist

$$\frac{p(x)}{q(x)}$$

Die Lösung ergibt sich wie folgt:

```
(3/4x^3  + 19/4x^2  + 4x  + 1) : (3/4x^2 + 4x)  =  x + 1   Rest  1
3/4x^3  +    4x^2
 ____________________________
            3/4x^2  + 4x  + 1
            3/4x^2  + 4x
            ___________________
                             1
```

Um die Division von Hand durchzuführen, gehen Sie allgemein wie folgt vor:

- Sie ordnen sowohl $p(x)$ als auch $q(x)$ in absteigender Reihenfolge der Grösse der Potenzen.

$$p(x) = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \ldots + a_2 \cdot x^2 + a_1 \cdot x + a_0$$

$$q(x) = b_m \cdot x^m + b_{m-1} \cdot x^{m-1} + \ldots + b_2 \cdot x^2 + b_1 \cdot x + b_0$$

- Sie gehen nun von links nach rechts durch $p(x)$, und teilen jeweils immer durch das erste Element von $q(x)$. Sie arbeiten sich also gewissermassen durch $p(x)$ durch...

  Zu Beginn teilen Sie das erste Element von $p(x)$ durch das erste Element von $q(x)$.

$$\frac{a_n \cdot x^n}{b_m \cdot x^m} = \frac{a_n}{b_m} x^{n-m} = e_1(x)$$

- Nun rechnen Sie zurück, d.h. Sie multiplizieren Ihr Ergebnis mit dem ganzen (!) Polynom

$q(x)$ und schreiben dieses Resultat unter $p(x)$.

$$e_1(x) \cdot q(x)$$

- Jetzt bilden Sie die Differenz von $p(x)$ mit dem gerade eben berechneten Produkt, und ergänzen diese wieder ganz unten.

$$d_1(x) = p(x) - e_1(x) \cdot q(x)$$

- Nun dividieren Sie wieder das erste Element dieser Differenz durch das erste Element von $q(x)$

$$\frac{\text{1. Element von } (d_1(x))}{b_m \cdot x^m} = e_2(x)$$

- Sie rechnen wieder zurück, d.h. Sie multiplizieren $e_2(x)$ mit $q(x)$ und schreiben dies unter $d_1(x)$.

$$e_2(x) \cdot q(x)$$

- Sie berechnen die Differenz von $d_1(x)$ mit dem berechneten Produkt, und ergänzen diese wieder ganz unten.

$$d_2(x) = d_1(x) - e_2(x) \cdot q(x)$$

- Sie merken: Der Vorgang beginnt wieder von neuem. Sie wiederholen diese Schritte solange, bis die die jeweils neu entstehende Differenz von einer kleineren Ordnung als $q(x)$ ist. Falls die letzte Differenz nicht verschwindet (gleich null wird), bleibt sie als Rest zurück.

Falls Sie das Prinzip verstanden haben, versuchen Sie sich an den Aufgaben. Sonst schlagen Sie auf [Bruenner06] nach, wo Sie auch weitere Beispiele ausprobieren können.

A1.2 Berechnen Sie

a) $\dfrac{x^2+3x+2}{x+1}$

b) $\dfrac{8x^3+16x^2+16x+10}{x^2+4x+4}$

Tipp: Bei Teilaufgabe b) wird die Division einen Rest ergeben!

1.4 Modulo-2-Arithmetik

Bei Übertragungssystemen wird nicht wie oben mit dem uns besser bekannten dezimalen Zahlensystem gerechnet. Man beschränkt sich auf das aus dem Computerbereich bekannte binäre Zahlensystem, d.h. die Ziffern 0 und 1. Es ist deshalb notwendig, die Rechenvorschriften neu zu definieren, da wir uns ja in einer neuen Basis befinden.
Wir geben als erstes die Wahrheitstabellen für Addition und Subtraktion an (aus [Meyer02, S. 297]):

- Addition

$$\begin{aligned} 0 + 0 &= 0 \\ 0 + 1 &= 1 \\ 1 + 0 &= 1 \\ 1 + 1 &= 0 \end{aligned} \tag{1.5}$$

- Multiplikation

$$\begin{aligned} 0 \cdot 0 &= 0 \\ 0 \cdot 1 &= 0 \\ 1 \cdot 0 &= 0 \\ 1 \cdot 1 &= 1 \end{aligned} \tag{1.6}$$

Diese Wahrheitstafeln entstehen, indem man alle Elemente (hier: 0 und 1) miteinander kombiniert.

A1.3 Vergleichen Sie die beiden Wahrheitstafeln mit digitalen Schaltungselementen, die Sie kennen.

Sie erkennen, dass $1 + 1 = 0$. Wir rechnen also ohne Überträge! Deshalb bezeichnet man dieses Rechnen als **Modulo-2-Arithmetik**. Die Modulo-Rechnung müssen Sie sich wie auf einem Kreis vorstellen: Wenn man ein Resultat erhält, das die zur Verfügung stehenden Ziffern überschreitet, so beginnt man einfach wieder von vorne mit zählen.

Bsp. 1.4 Nehmen wir mal die Modulo-6-Arithmetik: Hier ergibt zum Beispiel $3 + 5$ nicht etwa 8, da 8 nicht mehr im zulässigen Bereich liegt. Man gibt deshalb einfach den Rest an, der bei

der Division von 8 durch 6 entstehen würde. In unserem Fall ist das 2. Also ist $3+5=2$ in der Modulo-6-Arithmetik. In der Modulo-2-Arithmetik berechnet man entsprechend den Rest bei der Division durch 2.

Ausserdem sind Addition und Subtraktion gleichwertig, d.h. Gleichung (1.5) gilt ebenfalls für die Subtraktion. Wir verwenden für die Modulo-2-Addition das Symbol $\oplus$ und für die Modulo-2-Multiplikation das Symbol $\otimes$.

Achtung: Diese Operationen sind immer ziffernweise auszuführen, wie Sie am nächsten Beispiel erkennen werden. Es gibt also keine Überträge! Die Ziffern bezeichnet man als **Bits**.

Bsp. 1.5 Gegeben sind die beiden binären Zahlen $(A) = 1010111$ und $(B) = 0011101$. Die Addition (bwz. Subtraktion) ergibt sich zu

$$A+B = (1\oplus 0, 0\oplus 0, 1\oplus 1, 0\oplus 1, 1\oplus 1, 1\oplus 0, 1\oplus 1) = 1001010$$

Die Division (das Entgegengesetzte zur Multiplikation) zweier Zahlen wollen wir uns etwas genauer ansehen. Wir dividieren 100111 durch 101. Das Resultat ergibt sich zu 1011. Wie geht man dabei vor? Wir verwenden das selbe Schema wie bei der Polynomdivision:

```
1 0 0 1 1 1  /  1 0 1  =  1 0 1 1
1 0 1 --------------------+ | | |
-----                       | | |
  0 1 1 1 1                 | | |
  0 0 0     -----(*)--------+ | |
  -----                       | |
    1 1 1 1                   | |
    1 0 1   ------------------+ |
    -----                       |
      1 0 1                     |
      1 0 1 --------------------+
      -----
          0 --> Rest 0
```

A1.4 Was fällt Ihnen bei der mit einem (*) markierten Zeile auf?
Sie sehen ausserdem: Das Vorgehen ist bis auf eine Ausnahme analog zur Polynom-Division mit reellen Zahlen. Aber welcher Unterschied besteht?

Damit haben wir die Grundoperationen Modulo-2 eingeführt. Lösen Sie nun die folgenden Aufgaben.

A1.5 Berechnen Sie:
a) 11000101 + 1001011011
b) 11110111 / 10011

Gratulation! Sie sind jetzt am Schluss dieses Kapitels angelangt. Mit der Lernkontrolle auf Seite 14 können Sie nun schauen, ob Sie die drei Themen verstanden haben. Wenn ja, dann melden Sie sich beim Tutor für den Test.
Diese Vorgehensweise wird - wie ganz zu Beginn in der Arbeitsanleitung beschrieben - in den folgenden Kapitel genau gleich sein.

Lösungen der Aufgaben

Aufgabe 1.1

a) $\begin{pmatrix} 8 & -13 \\ -1 & 3 \\ 2 & 6 \end{pmatrix}$

b) $\begin{pmatrix} 3 & -3 \end{pmatrix}$

c) $\begin{pmatrix} 3 & 1 \\ 1 & -11 \\ 0 & -17 \\ -1 & -6 \end{pmatrix}$

Aufgabe 1.2

a) Das Resultat ist $x+2$.

Man erhält dieses wie folgt:

```
(x^2  + 3x  + 2) : (x + 1)  =  x + 2
 x^2  +  x
 --------------
        2x  + 2
        2x  + 2
        -------
              0
```

b) Das Resultat ist: $8x-16$ Rest $48x+74$.

Man kann dies auch schreiben als $8x-16+\frac{48x+74}{x^2+4x+4}$.

Die Rechnung lautet wie folgt (nächste Seite):

```
(8x^3  + 16x^2  + 16x  + 10) : (x^2 + 4x + 4)  =  8x - 16   Rest  48x + 74
 8x^3  + 32x^2  + 32x
 -------------------------
       - 16x^2  - 16x  + 10
       - 16x^2  - 64x  - 64
       --------------------
                  48x  + 74
```

Aufgabe 1.3

Die Modulo-2-Addition entspricht einer EXOR-Verknüpfung (Exklusiv-Oder), die Modulo-2-Multiplikation einer AND-Verknüpfung.

Aufgabe 1.4

Bei der mit einem (*) markierten Zeile müssen wir 011 durch 101 dividieren. Das ergibt natürlich 0, weil das erste Element aus 011 ja 0 ist, und wir dieses durch das erste Element von 101 dividieren müssen!
Unterschied: Die Bildung der Differenzen erfolgt natürlich ebenfalls nach der Modulo-2-Arithmetik. So ist zum Beispiel $0-1=0+1=1$ (Subtraktion und Addition binärer Zahlen ist ja identisch) und nicht etwa wie im dezimalen System $0-1=-1$!

Aufgabe 1.5

a) 1010011110

b) 1101

Lernkontrolle

Diese Lernkontrolle gibt Ihnen zu den drei behandelten Themen je eine Möglichkeit, ihr Wissen zu testen. Falls Sie auch nach mehrmaligen Versuchen nicht zur korrekten Lösung kommen, schauen Sie sich das entsprechende Unterkapitel nochmals an. Lösen Sie die dort stehenden Aufgaben erneut, und/oder investieren Sie etwas Zeit bei den angegebenen weiterführenden Links.

L.1

Sei $A = \begin{pmatrix} -1 & 5 & 5 \\ 2 & -1 & 0 \\ 0 & 2 & -1 \end{pmatrix}$, $B = \begin{pmatrix} 1 & 1 & 0 \\ 1 & -1 & 0 \\ 2 & 1 & -2 \end{pmatrix}$ und $C = \begin{pmatrix} 2 & 1 \\ 0 & 0 \\ 1 & -2 \end{pmatrix}$.

Berechnen Sie $A(B^T \cdot 2C)$

L.2

Berechnen Sie $\dfrac{4x^4 + 7x^3 + 3x^2 + 32/9x - 1/3}{3x^3 + 3x^2 + 8/3}$.

L.3

Subtrahieren Sie die zweite von der ersten binären Zahl. Gebrauchen Sie dabei Modulo-2-Arithmetik.

a) 10111, 100

b) 110001,101100

L.4

Dividieren Sie 1001000 durch 1011.

Lösungen der Lernkontrolle

L.1

$$\begin{pmatrix} 2 & 36 \\ 10 & -10 \\ 16 & -12 \end{pmatrix}$$

L.2

Das Resultat ist $4/3x+1$ Rest -3.

Die Rechnung ist wie folgt:

```
(4x^4  + 7x^3  + 3x^2  + 32/9x  - 1/3) : (3x^3 + 3x^2 + 8/3)  =  4/3x + 1
                                                                 Rest  -3


 4x^4  + 4x^3          + 32/9x
 -----------------------------------
         3x^3  + 3x^2           - 1/3
         3x^3  + 3x^2           + 8/3
         ---------------------------
                                  - 3
```

L.3

a) 10011

b) 11101

L.4

Das Resultat ist 1010 Rest 110.

Die Lösung ergibt sich wie folgt:

```
              ..

1 0 0 1 0 0 0  /  1 0 1 1  =  1 0 1 0

1 0 1 1 ----------------------+ | | |

-------                         | | |

  0 1 0 0 0 0                   | | |

  0 0 0 0     ------------------+ | |

  -------                         | |

    1 0 0 0 0                     | |

    1 0 1 1   --------------------+ |

    -------                         |

      0 1 1 0                       |

      0 0 0 0 -----------------------+

      -------

        1 1 0      Rest = 110
```

Kapitel 2

Kanalcodierung

2.1 Einführung und Lernziele

Zu Beginn dieses Kapitels führen wir ein **Übertragungsmodell** ein und stellen Ihnen einige grundlegende Verfahren zur Fehlererkennung und -korrektur vor. Zu Beginn kommen wir noch ohne mathematischen Formeln und Gleichungen aus. Im zweiten Teil werden jedoch Gleichungen vorgestellt, mit denen die **Anzahl korrigierbarer Fehler** bestimmt werden kann und es werden mit Matrizen Codes berechnet.
Folgendes sollten Sie nach Abschluss des Kapitels erreicht haben:

- Sie haben einen groben Überblick über das Thema Kanalcodierung.
- Sie kennen einige Definitionen, mit denen Sie Schranken für die Fehlerkorrigierbarkeit des Codes berechnen können.
- Sie wissen, was ein linearer Code ist und wie Sie diesen berechen können.

Die nötigen Vorkenntnisse für dieses Kapitel haben Sie sich in Kapitel 1 angeeignet.

LERNZIELE

Leitidee

Um die einzelnen Codierungsarten zu verstehen, benötigt es ein Grundverständnis des zugrundeliegenden Systems und eine Einführung in die Eigenschaften von unterschiedlichen Codes.

Dispositionsziele

Sie entwickeln ein Interesse für Übertragungssysteme und kennen das Grundprinzip der Codierungsmechanismen. Sie sind in der Lage zu entscheiden, ob es sinnvoll ist, den Mechanismus so zu konstruieren, dass er auftretende Fehler korrigieren kann oder ob eine reine Fehlerkontrolle ausreicht.

Operationalisierte Lernziele

Sie können

- das verwendete Übertragungsmodell erklären.
- einen linearen Blockcode berechnen.
- die Anzahl Fehler bestimmen, die ein linearer Code erkennen oder korrigieren kann.

2.2 Allgemeines Übertragungssystem

In der Einleitung des Leitprogrammes haben wir das erste Mal von Redundanz gesprochen. Erinnern Sie sich noch daran, was darunter verstanden wird? Dort haben wir gesagt, dass man unter Hinzufügen von Redundanz das Hinzufügen von zusätzlichen Daten versteht. Im Verlaufe des Leitprogrammes werden wir sehen, wie Redundanz gezielt genutzt werden kann, so dass Fehler gefunden und korrigiert werden können.

Als Erstes betrachten wir jedoch ein Blockdiagramm eines digitalen Kommunikationssystems:

A2.1 Versuchen Sie mit Hilfe des folgenden Schemas das verwendete Modell eines digitalen Kommunikationssystems zu verstehen.

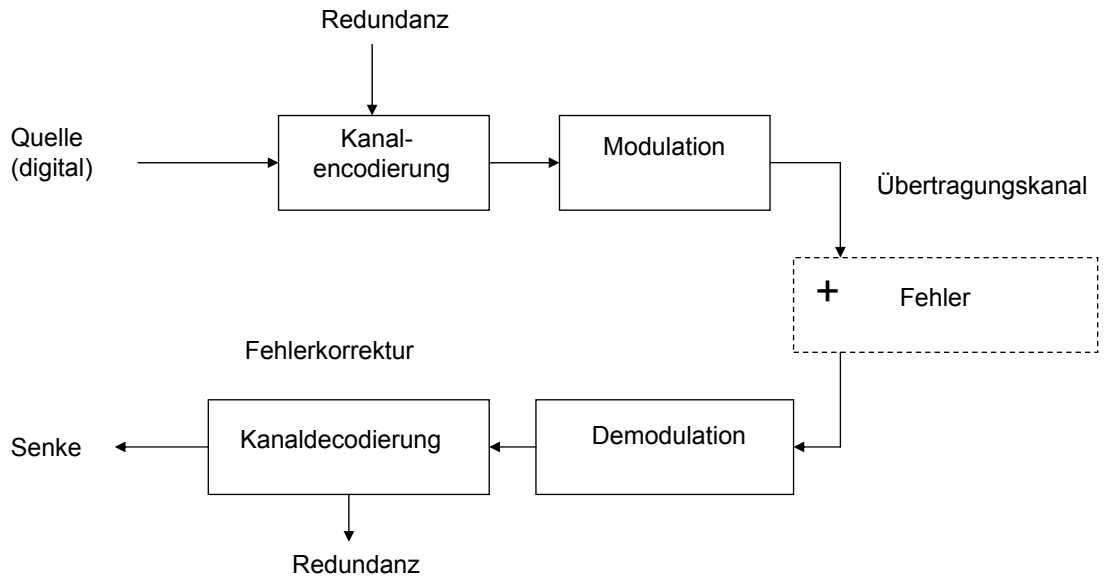

Abbildung 2.1: Modell aus [WikiK06]

In unserem Leitprogramm sprechen wir von „Kanalcodierung“. In der Grafik finden sie den Begriff „Kanalencodierung“. Die beiden Begriffe können gleichbedeutend verwendet werden.

Es sollte Ihnen klar geworden sein, dass die Funktion der Kanalcodierung ist, dem zu sendenden Wort Redundanz hinzuzufügen. Können Sie sich eine einfache Variante vorstellen wie dies möglich ist?

Wir werden Ihnen hier zu Beginn zwei einfache Varianten vorstellen:

2.2.1 Repetitionscode

Als erstes der Repetitionscode (m,1). Jedes Bit wird m-Mal übertragen und am Empfänger entscheidet man sich für das Symbol, welches am meisten aufgetreten ist. Beachten Sie, dass nur ungerade m Sinn machen. Nehmen wir an, wir möchten das Codewort 101 der Länge k=3 übertragen. Wir wählen m=3, das heisst jedes Bit wird dreimal übertragen. Das übertragene Codewort wäre somit 111000111.

Mit Matrizen lässt sich dies folgendermassen darstellen:

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

Die Matrix G bezeichnet man als Generatormatrix. Sie hat die Dimension $n \times k$, wobei $n = m \cdot k = 3 \cdot 3 = 9$ die Blocklänge und $k = 3$ die Anzahl Nachrichtenbits ist.

2.2.2 Parity Check

Eine weitere Möglichkeit wird in der Literatur als Parity Check bezeichnet. Hier wird jedem Datenwort ein zusätzliches Bit so zugefügt, dass die Anzahl der Einsen *gerade* wird.

Die allgemeine Generatormatrix für ein Datenwort ist folgendermassen definiert:

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

Multiplikation mit der letzten Zeile von G erzeugt somit das *Parity Bit*.

Auch hier ein Beispiel : Nehmen wir nun an, Sie möchten die Nachricht „100101“ übertragen. Die Anzahl der Einsen ist, da $3 mod 2 = 1$, ungerade. Somit sollte das erste Bit auf 1 gesetzt werden. Wenn Sie mit der Matrix nachrechnen, erkennen Sie, dass dies der Fall ist und erhalten folgendes Codewort: 1100101

A2.2 Bestimmen Sie den Repetitionscode (3,1) und den Parity Check des folgenden Nachrichtenwortes: 110101

Wir werden im Folgenden noch einige raffiniertere Verfahren behandeln, doch lassen Sie uns zuerst einige Grundlagen erarbeiten.

2.3 Verfahren zur Fehlerkorrektur und -erkennung

Es gibt zwei verschiedene Varianten zur Behandlung von auftretenden Fehlern. Die erste Variante ist so konstruiert, dass der Empfänger den Sender benachrichtigt, falls Daten fehlerhaft übertragen wurden. Solche Systeme werden als **automatic repeat request (ARQ)- Systeme** bezeichnet. Für ein solches Verfahren muss der zugrundeliegende Code nur Fehler erkennen können. Die zweite Möglichkeit zielt darauf ab, dass der Empfänger Fehler erkennt und gleich selbst korrigieren kann. Eine solche Fehlerkorrekturstrategie wird als **forward error correction (FEC)** bezeichnet.

A2.3 Vergleichen Sie FEC- und ARQ-Verfahren anhand von Vor- und Nachteilen der einzelnen Verfahren.

2.4 Klassierung von Codes

Im folgendenden Diagramm können Sie erkennen, dass sich zwei unterschiedliche, mathematisch etwas anspruchsvollere Klassen von Codes unterscheiden lassen: Blockcodes und Faltungscodes.

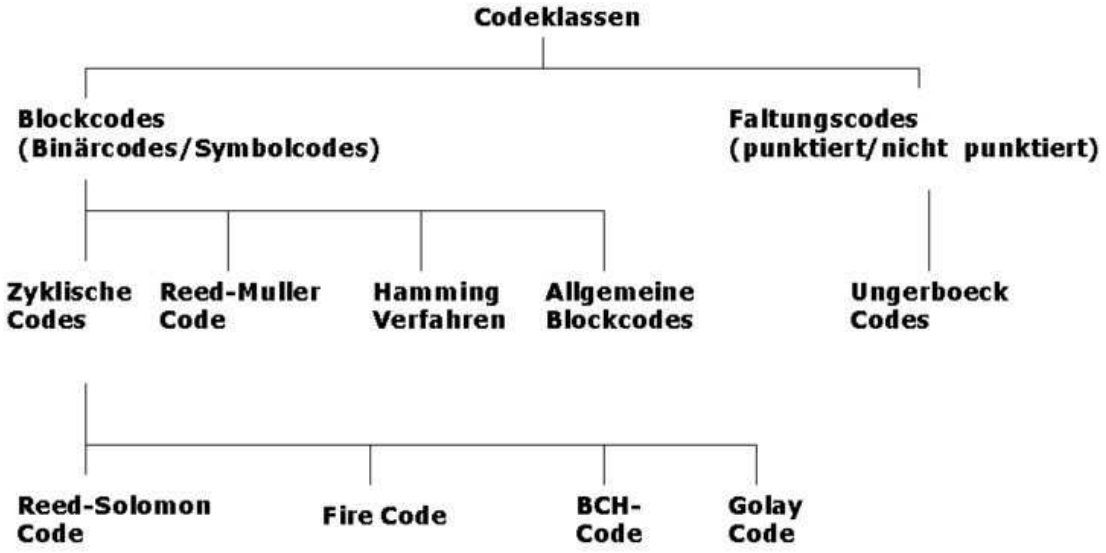

Abbildung 2.2: Diagramm aus [WikiK06]

Ein Blockcode ist dadurch gekennzeichnet, dass die benutzten Codewörter alle dieselbe Anzahl an Symbolen (in unserem Fall zweiwertig, also Bits) n haben. Faltungscodes führen keine blockweise Codierung aus. Sie sind auf einem fortlaufendem Datenstrom aufgebaut. Im Rahmen dieses Leitprogramms befassen wir uns nur mit den **Blockcodes**.

Desweiteren gibt Ihnen das Diagramm eine grobe Übersicht, was für Codetypen innerhalb der beiden Klassen existieren. In diesem Leitprogramm liegt der Schwerpunkt bei den zyklischen Codes. Wir werden diese im nächsten Kapitel noch etwas genauer untersuchen und im letzten Kapitel dann denn BCH-Code als Beispiel für zyklische Codes vorstellen.

2.4.1 Defintionen

Ein Blockcode wird als **linear** bezeichnet, wenn die Summe zweier Codewörter wieder ein Codewort ergibt.

Es sei nun der lineare Code $C(n,k)$ gegeben (Blocklänge n, Anzahl Nachrichtenbit k). Ein **Codewort** c eines **Informationswortes** i ist nun durch die **Generatormatrix** G (Dimension $k \times n$) definiert

$$c = i \cdot G$$

Dazu ist immer auch eine **Prüfsummenmatrix (parity check matrix)** H, der Dimension $(n-k) \times n$ definiert. Es gilt die Beziehung

$$(HG^T) \quad mod \quad 2 = 0$$

Mit Hilfe der Parity Check Matrix kann im Empfänger die Richtigkeit der Codewörter festgestellt werden. „Wenn r empfangen wird, berechnet der Empfänger den folgenden Vektor:

$$s = rH^T = (s_0, s_1, \cdots s_{n-k-1})$$

Dieser Vektor wird als Syndrom bezeichnet. $s = 0$, wenn und nur wenn d ein Codewort ist. Daher wissen Sie, wenn $s \neq 0$, dass r kein Codewort ist und ein Fehler passiert ist. Es ist möglich, dass die Fehler in gewissen Fehlervektoren nicht detektiert werden. Dies passiert wenn ein Fehlermuster identisch zu einem Codewort ist. Solche Fehlermuster werden als nichtdetektierbar bezeichnet.“(aus [Lin04, S. 72])

A2.4 Gegeben sei folgende Parity Check Matrix aus [Lin04, Beispiel 3.3 S. 71]

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

Ist der empfangene Vektor r= (1,1,1,1,0,0,0) ein korrektes Codewort?

Zur Notation. Ein Code C(n,k,d) hat folgende Eigenschaften:

- n ist die Länge der Codeworte.

- k ist die Länge der Datenworte.
- d ist die Mindestdistanz des Codes. (Wir werden in Kürze festhalten, was darunter zu verstehen ist.)

Ein Code wird systematisch genannt, falls die ersten k Bits jedes Codewortes identisch sind mit den entsprechenden Informationsbits. Die restlichen $(n-k)$ Bits sind Linearkombinationen der Informationsbits, definiert durch die Matrix P. Sie werden als **Prüfsummenbits** bezeichnet. Somit sieht die Generatormatrix wiefolgt aus

$$G = \begin{pmatrix} p_{11} & \cdots & p_{1,n-k} & 1 & & 0 \\ \vdots & & \vdots & & \ddots & \\ p_{k1} & \cdots & p_{k,n-k} & 0 & & 1 \end{pmatrix} = [P, I_k]$$

A2.5 Es sei folgender (5,3) Code gegeben:

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

Berechnen Sie zu allen möglichen Informationsworten der Länge 3 (also 000, 001, 010, 011..)die zugehörigen Codewörter.

2.5 Einige wichtige Definitionen

Wir werden im folgenden die Begriffe Distanz, Gewicht und Hamming Distanz einführen. Auf [WikiH06] finden sich folgende Zeilen:
„In information theory, the **distance** between two strings of equal length is the number of positions for which the corresponding symbols are different. Put another way, it measures the number of substitutions required to change one into the other, or the number of errors that transformed one string into the other. The **weight** of a string is its distance from the zero string (string consisting of all zeros) of the same length. That is for binary strings the number of elements in the string which are not zero.“

Ein Beispiel (ähnlich auch auf Wikipedia zu finden):

Ein Code $C(n,k,d)$ besteht aus folgenden drei Wörtern:

x = 1011101,

y = 1001001,

z = 0111001.

Die Distanz zwischen x = 1011101 und y = 1001001 ist 2. Die Distanz zwischen x und z ist 3.

A2.6 Berechnen Sie die Distanz zwischen y und z.

Das **Gewicht** eines Vektors c ist die Anzahl von 0 verschiedenen Elemente von c

$$wt(c) = \sum_{j=1}^{n-1} wt(c_j) \quad \text{mit} \quad wt(c_j) = \begin{cases} 1 & c_j \neq 0 \\ 0 & c_j = 0 \end{cases}$$

Die **Distanz** zweier Vektoren $dist(a,c)$ ist die Anzahl der unterschiedlichen Elemente von a und c. Somit gilt

$$dist(a,c) = wt(a \oplus c)$$

Die **Hamming Distanz** d eines Codes C ist die minimale Distanz zweier unterschiedlicher Codeworte aus der Menge aller möglichen Codewörter

$$d = \min\{dist(a,c)\} \text{ mit a und c Codewörter aus C}$$

Wenn Sie die drei Distanzen aus dem obigen Beispiel vergleichen, so sollten Sie erkennen, dass der kleinste Wert 2 ist. Somit ist die Hamming Distanz des Codes ebenfalls gleich 2.

Ein empfangener Vektor $r = c \oplus e$ mit $c \in C$ und Fehlervektor e wird eindeutig dem Codewort c zugeordnet werden, falls für die Distanz zu einem beliebigen Codewort a gilt

$$dist(c, c \oplus e) < dist(a, c \oplus e)$$

Das heisst, dass ein empfangenes Codewort demjenigen zugeordnet wird, zu welchem es die kürzeste Distanz hat.

Direkt daraus kann man die **Anzahl Fehler** $wt(e)$ ableiten, welche eindeutig korrigiert werden können. Für eine linearen Code C(n,k,d) sind dies:

$$wt(e) \leq \left\lfloor \frac{d-1}{2} \right\rfloor$$

Will man nur Fehler erkennen, so erkennt der Code maximal d-1 Fehler.

Damit wären wir bereits am Schluss des Theorieteil dieses Kapitels. Sie haben einige neue Ideen gesehen. Blättern Sie das Kapitel nochmals durch und denken Sie jene Punkte, die Ihnen noch nicht ganz klar sind, nochmals durch. Anschliessend können Sie sich der Lernkontrolle und dem Tutortest widmen.

Lösungen zu den Aufgaben

Aufgabe 2.1

Das Wichtigste ist zu sehen, dass die Kanalcodierung dem von der Quelle gesendeten Signal Redundanz hinzufügt, welche dann beim Empfänger wieder entfernt wird. Nach der Codierung wird das Signal moduliert und über den Kanal übertragen. Eine fehlerfreie Übertragung ist nicht möglich und somit treten bei der Übertragung der Daten unweigerlich Fehler auf. Genau hier setzt die Kanalcodierung an. Die zu Beginn hinzugefügte Redundanz ermöglicht es dem Empfänger Fehler zu erkennen oder gar zu korrigieren. Wir werden im Verlaufe des Kapitels feststellen, dass bei reiner Fehlererkennung der Empfänger den Sender auffordert, ihm das Codewort nochmals zu senden. Um das Modell jedoch einfach zu halten wurde dieser Rückführungskanal bewusst weggelassen. Falls Sie an weiteren Details interssiert sind, so können Sie diese in [Lin04, S.1-3] nachlesen.

Aufgabe 2.2

Repetionscode:

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

Parity Check:

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

Falls Sie nicht zum richtigen Resultat gekommen sind, dann schauen sie sich die vorgelösten Beispiele nochmals an.

Aufgabe 2.3

Durch den Vergleich sollten Sie erkennen, dass in den beiden Fällen unterschiedliche Anforderungen an den Code gestellt werden.
Der Hauptvorteil eines ARQ-Verfahrens ist, dass die *reine Fehlererkennung* geringere Anforderungen an die Codierungsverfahren stellt. Ausserdem wird den Datenworten *weniger Redundanz* hinzugefügt, weil Fehler nicht korrigiert werden müssen, sondern einfach das Paket nochmals

gesendet wird. Bei einer niedrigen Fehlerrate, wo nur ein kleiner Teil aller Pakete erneut angefordert werden muss, erweisen sich also ARQ-Verfahren als einiges effizienter.
Die Effizienz von ARQ-Verfahren ist aber bei einer *hohen Fehlerrate* stark eingeschränkt, da viele Pakete nochmals übertragen werden müssen. Desweiteren kann eine *lange Übertragungsstrecke* bei ARQ-Verfahren zu unerwünschten Wartezeiten führen. Zudem muss für die Implementierung eines ARQ-Verfahren ein *Rückkanal* für die Bestätigungs- bzw. Fehlermeldungen zur Verfügung stehen. Steht kein Rückkanal zur Verfügung, muss zwingend ein FEC-Verfahren gewählt werden.
Die Implementierung des FEC-Verfahrens sowie auch die Konstruktion des Senders und Empfängers ist um einiges einfacher. So ist unter anderem kein kompliziertes Protokoll für den Daten-/Bestätigungs-Austausch notwendig. Ist die *Einfachheit der Implementierung* ein Kriterium für die Auswahl des Verfahrens, wird sicherlich ein FEC-Verfahren gewählt. Mit den heutigen Technologien ist dies jedoch kein wichtiges Kriterium mehr.

Aufgabe 2.4

Man erhält den Syndromvektor $s = (0,0,1) \neq 0$ Das empfangene Codewort enthält somit einen Fehler.

Aufgabe 2.5

Datenwort	Codewort
(000)	(00000)
(001)	(01001)
(010)	(11010)
(011)	(10011)
(100)	(10100)
(101)	(11101)
(110)	(01110)
(111)	(00111)

Aufgabe 2.6

Die Hamming Distanz zwischen y = 1001001 und z = 0111001 ist 3.

Lernkontrolle

L1

Welchem Zweck dient die Kanalcodierung? Was ist die Grundidee, mit der dies erreicht werden soll?

L2

Gegeben sei folgende Generatormatrix

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

Berechnen Sie die Codeworte der folgenden zwei Datenworte:

$i_1 = (1,1,0,1)$

$i_2 = (1,0,0,0)$

L3

Ein Code C(4,3,2) kann 2 Fehler korrigieren. Stimmt diese Aussage?

Lösungen zur Lernkontrolle

L1

Die Kanalcodierung dient der Fehlererkennung und -korrektur bei der Übertragung von Daten. Die Grundidee, mit der dies erreicht wird, ist das Hinzufügen von Redundanz.

L2

$c_1 = i_1 \cdot G = (0,0,0,1,1,0,1)$
$c_2 = i_2 \cdot G = (1,1,0,1,0,0,0)$

L3

Ein C(4,3,2) Code besitzt die Mindestdistanz 2. Somit können mit dem Code $(d-1) = 1$ Fehler erkannt werden und $\lfloor \frac{d-1}{2} \rfloor$=0 Fehler korrigiert. Die Aussage ist somit falsch.

Kapitel 3

Zyklische Codes

3.1 Einführung und Lernziele

In diesem Kapitel beschäftigen wir uns mit einer Teilmenge der linearen Codes, den zyklischen Codes. Wir werden sehen, welche Eigenschaften ein linearer Code erfüllen muss, damit er als zyklisch bezeichnet wird. Die zyklischen Codes finden in der Praxis grosse Anwendung und eine weite Verbreitung. Aus diesem Grund bilden Sie einen Schwerpunkt unseres Leitprogrammes. Eines der promineteren Beispiele ist sicherlich Ethernet. Am Ende jedes Ethernetpacket, welches versendet wird, findet sich ein 32 Bit langer zyklischer Code. Wir werden in diesem Kapitel einige allgemeine Eigenschaften von zyklischen Codes erarbeiten und einige Möglichkeiten zur Berechnung des Codes aufzeigen.
Nach Abschluss des Kapitels sollten Sie in der Lage sein, einige einfache zyklische Codes zu berechnen. Desweitern vermitteln wir Ihnen ein grobes Verständnis für die Implementierung von zyklischen Codes.

LERNZIELE

Leitidee

Die zyklische Codierung stellt als Teilmenge der linearen Codes eine in der Anwendung oft verwendete Art von Code dar.

Dispositionsziele

Die Charakterisierung der zyklischen Codes ist Ihnen bekannt und Sie sind mit deren Anwendung vertraut.

Operationalisierte Lernziele

Sie können

- zyklische Codes erstellen.
- einen zyklischen Code erstellen, welcher sich zur Erkennung von bestimmten Fehlern eignet.
- einen zyklischen Code in Software implementieren.

3.2 Definitionen

Eine spezielle Klasse der linearen Codes sind die zyklischen Codes. Für diese ist die Codierung vergleichsweise einfach und es existieren auch gute Decodierverfahren.

„Ein linearer Code C(n,k) wird zyklisch genannt, wenn jede zyklische Verschiebung eines Codewortes, wieder ein Codewort in C ergibt.“ [Lin04, s.135]

Was heisst dies nun genau? Betrachten wir ein Codewort $(c_0 \cdots c_{n-1}) \in C$. Wir verschieben die einzelnen Zeichen zyklisch, so dass wir mit $(c_{n-k}, c_0 \cdots c_{n-(k+1)}) \in C$ das um k verschobene Codewort erhalten. Ist das so entstandene Codewort ebenfalls ein Element von C, so wird der Code als zyklisch bezeichnet.
Es ist nicht trivial, einem linearen Code anzusehen, ob er zyklisch ist. Es gibt jedoch mathematische Verfahren, dies festzustellen. Hier verzichten wir jedoch auf eine genauere Betrachtung.

Um die algebraischen Eigenschaften eines Codes besser zu verstehen, beschreiben wir im folgenden ein Wort $v = (v_0, v_1 \cdots v_{n-1})$ durch ein Polynom $v(x) = (v_0 + v_1 X + v_2 X^2 + \cdots + v_{n-1} X^{n-1})$. Somit entspricht jedes Wort einem Polynom mit maximalem Grad n-1.

Ein Beispiel:
Aus dem Codewort 11001 erhalten wir nach obiger Regel $1 + X + 0 \cdot X^2 + 0 \cdot X^3 + X^4$
In diesem Abschnitt wird somit das Datenwort zum Datenpolynom, das Codewort zum Codepolynom und das empfangenen Wort zum empfangen Polynom.
Um gleich zu Beginn einige Verwirrung zu vermeiden, verraten wir Ihnen hier schon vorweg, dass sich zyklische Codes durch Multiplikation mit einem Generatorpolynom generieren lassen.
Als Formel ausgedrückt:

$$c(X) = i(X) \cdot g(X)$$

3.3 Das Generatorpolynom g(X)

3.3.1 Allgemeine Eigenschaften von g(X)

Dieses Unterkapitel beschäftigt sich mit einigen Eigenschaften des Generatorpolynoms. Es kann gezeigt werden, dass ein zyklischer Code eindeutig durch ein Generatorpolynom generiert werden kann. Falls Sie sich für die Herleitung dieser Eigenschaften interessieren, sind die Seiten 136-140 [Lin04] wärmstens zu empfehlen.

A3.1 Berechnen Sie die Codewörter zu $i_1 = 101$ und $i_2 = 110$, welche mit $g(X) = 1 + x^2$ erzeugt werden. (Denken Sie daran: Die Multiplikation ist modulo zwei zu verstehen.)

Es kann eine grosse Anzahl von Polynomen gefunden werden, die zyklische Codes generieren. Einige dieser Polynome generieren Codes, mit denen Fehler besser erkannt werden können als andere. Wie diese Generatorpolynome ausgewählt werden sollen, um gute Codes zu erhalten ist ein schwieriges Problem. Es gibt jedoch einige gute Klassen von zyklischen Codes, welche heute praktische Anwendung finden.

Bevor Sie einige Aufgaben und Beispiele ansehen werden, stellen wir Ihnen noch die **systematische Form** von zyklischen Codes vor. Sie erinnern sich: Die systematische Form ist so aufgebaut, dass die ersten k Bits jedes Codewortes identisch mit den entsprechenden Informationsbits sind.

Um die systematische Form zu erhalten gehen Sie folgendermassen vor:

- Multiplizieren Sie das Informationspolynom $i(X)$ mit X^{n-k}
- Berechnen Sie den Rest $b(X)$ der Division von $X^{n-k}u(X)$ durch $g(X)$
- Das codierte Polynom $c(X)$ setzt sich aus $b(X) + X^{n-k}u(X)$ zusammen

Machen wir ein Beispiel (aus [Lin04, p142]):
Betrachten Sie den zyklischen Code C(7,4), welcher durch das Generatorpolynom $g(X) = 1 + X + X^3$ generiert werden soll. Es sei $u(X) = 1 + X^3$ die Nachricht, die codiert werden soll. Dividieren wir $X^{7-4}u(X) = X^3 + X^6$ durch g(X). Da wir finden, dass die Polynomdivision im binären Zahlenraum übersichtlicher ist, verwenden wir hier diese Darstellung. Zur Erinnerung $g = 1101$ und $X^3u(X) = 0001001$. Für die Polynomdivision müssen wir die Polynome

jedoch so schreiben, dass der höchste Koeffizient zuerst kommt. Wir erhalten somit die Division $X^3u(X)/g(X) = 1001000/1011$.

```
              ..

1 0 0 1 0 0 0  /  1 0 1 1  =  1 0 1 0
1 0 1 1 ----------------------+ | | |
-------                         | | |
  0 1 0 0                       | | |
  0 0 0 0    -------------------+ | |
  -------                         | |
    1 0 0 0                       | |
    1 0 1 1 ----------------------+ |
    -------                         |
      0 1 1 0                       |
      0 0 0 0 ----------------------+
      -------
        1 1 0      Rest = 110
```

Der Rest ergibt somit $X^2 + X$. Das Codewort ergibt sich dann zu $c(x) = b(X) + X^3u(X) = X + X^2 + X^3 + X^6$.

A3.2 Berechnen Sie zu folgenden Nachrichtenworten die Codeworte des C(7,4) Codes mit $g(X) = 1 + X + X^3$

a) $i(X) = X$

b) $i(X) = X + X^2 + X^3$

c) $i(X) = X + X^3$ in der systematischen Form

d) $i(X) = 1 + X + X^2 + X^3$ in der systematischen Form

3.3.2 Fehlerkorrekturfähigkeit

Wir möchten nun die Fähigkeit dieser Methode etwas genauer analysieren: Was für Fehler werden denn eigentlich detektiert?

Zuerst müssen Sie sich im Klaren sein, wie die Daten beim Empfänger verarbeitet werden:

Verarbeitung im Empfänger

Der Empfänger erhält das Polynom: $r(X) = c(X) + e(X) = i(X)g(X) + s(X)$. Zur Kontrolle, ob das Polynom ein Codewort ist, dividiert der Empfänger das empfangene Polynom durch g(X). Falls das empfangene Polynom ein Codewort ist, ist das **Syndrom** $s(X) = 0$.

Dividieren Sie das erhaltene Polynom $d(X)/g(X)$, so erkennen Sie, dass das Syndrom dem Rest der Division entspricht. Das Syndrom ist in zwei Fällen null. Einerseits, wie wir bereits gesehen haben, falls das empfangene Wort einem Codewort entspricht. Andererseits ist s(X) jedoch ebenfalls null, wenn e(X) identisch zu einem Codewort ist. Solche Fehler werden somit nicht erkannt.

Trotzdem: Das Fehlermuster $e(X)$ ist dem Empfänger unbekannt und der Empfänger muss $e(X)$ aufgrund des Syndroms schätzen.

Bei der Konstruktion des Generatorpolynoms ist es nun wichtig, Polynome zu finden, die so wenig wie möglich einer „Fehlergruppe“ entsprechen. Somit müssen Sie, bevor Sie eine Aussage über die Korrekturfähigkeit eines g(X) machen wollen, mögliche Fehlerarten kennen.

Einzelbitfehler

Unter einem Einzelbitfehler verstehen wir folgendes Fehlermuster: $E(X) = X^i$. Wir können garantieren, dass alle solche Fehler erkannt werden, wenn wir sicherstellen, dass $g(X)$ aus mindestens zwei Termen aufgebaut ist. Es lässt sich erkennen, dass durch Addition und Verschieben von g(X) unter keinen Umständen ein einzelner Term erzeugt werden kann.
Doch was nun, wenn nicht nur ein Einzelbitfehler, sondern zwei auftreten? Das Fehlerpolynom ist somit von folgender Gestalt: $E(X) = X^i + X^j = X^j(X^{i-j} + 1)$. Um solche Fehler zu erkennen, genügt es, wenn weder X noch $X^k + 1$ durch das Polynom $g(X)$ teilbar sind (k entspricht dem maximalen Wert von $i - j$).

Ungerade Anzahl Bitfehler

Als nächstes betrachten wir Fehlerarten mit ungerader Anzahl Bitfehler, wie zum Beispiel $E(X) = X^5 + X^2 + 1$. Ein Gegenbeispiel, bei welchem die Anzahl Fehler gerade ist, wäre $E(X) = X^3 + X$. Es kann gezeigt werden, dass jedes Polynom mit ungerader Anzahl Summanden nicht durch $X + 1$ dividierbar ist. Enthält somit das Polynom $g(X)$ $(X + 1)$ als Faktor, werden alle Fehler mit ungeraden Bitzahl erkannt.

Burstfehler

Als letztes betrachten wir noch Burstfehler. Dies ist wohl die wichtigste Klasse von Fehlern, die wir hier betrachten. Ein Burstfehler der Länge k kann folgendermassen dargestellt werden: $E(X) = X^i(X^{k-1} + \cdots + X + 1)$. Wenn nun k geringer ist als der Grad von g(X), kann gezeigt werden, dass der Rest niemals null ergeben kann und somit alle Burstfehler der Länge k kleiner dem Grad von g(X) erkannt werden können.

Weiter kann gezeigt werden, dass nur ein gewisser Anteil von Fehlern nicht detektiert werden kann, falls k den Grad des Polynoms überschreitet. Falls der Grad um eins grösser ist als der Grad des Polynoms so sind es $2^{-(k-1)}$ aller Fehler. Ist der Grad noch höher sind es nur 2^{-k} aller Fehler, die nicht detektiert werden.

A3.3 Betrachten wir nochmals das Generatorpolynom $g(X) = 1 + X + X^3$. Werden die zugehörigen Codeworte betrachtet, erkennt man, dass die minimale Distanz 3 ist.

a) Bestimmen Sie die Anzahl Einzelfehler, die mit dem Code detektiert und korrigiert werden können.

b) Bis zu welcher Länge werden alle Burstfehler erkannt und wie sieht es aus, wenn diese Länge überschritten wird?

3.4 Implementierung

Kommen wir nun zur konkreten Implementierung eines zyklischen Codes. Zyklische Codes können relativ gut in Hardware und Software abgearbeitet werden. Wir werden Ihnen ein Hardwarebeispiel vorstellen und Sie werden eine Software-Routine erarbeiten.

3.4.1 Hardware

Der folgende Abschnitt ist grösstenteils aus [Lin04, S. 147ff] übernommen.

Wir haben bis anhin im Kapitel gesehen, dass für das Berechnen der systematischen Form folgende drei Schritte notwendig sind:

- Multiplizieren Sie das Polynom $u(X)$ mit X^{n-k}
- Dividieren Sie das Resultat durch $g(X) = 1 + g_1X + \cdots g_{n-k-1}X^{n-k-1} + X^{n-k}$ um den Rest $b(X) = b_0 + b_1X + b_2X^2 + \cdots b_{n-k-1}X^{n-k-1}$ zu erhalten
- Bilden Sie das Codewort aus $b(X) + X^{n-k}u(X)$.

Dies kann durch die folgende Schaltung erreicht werden:

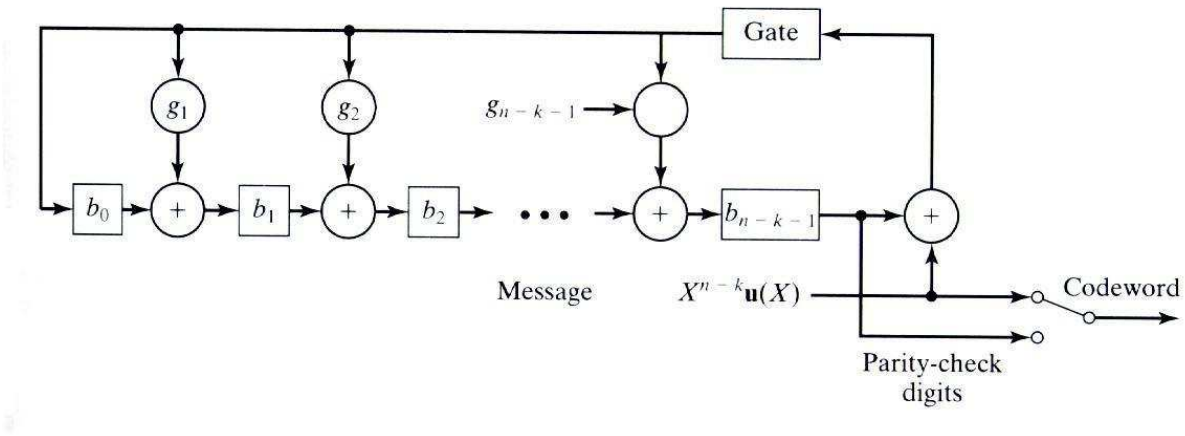

Abbildung 3.1: Hardwaremodell aus [Lin04, S. 148]

Das Codewort erhält man nun folgendermassen:

- Schalten Sie das Gate ein. Die k Informationsbit des Polynoms $u(X) = u_0 + u_1X + \cdots + u_{k-1}X^{k-1}$ werden in den Kommunikationskanal (mit Codeword gegekennzeichnet) und

gleichzeitig über das Gate in die Schaltung gespeist. Im Modell wird dieser Eingang mit $X^{n-k}u(X)$ beschrieben. Das Einspeisen in den Kommunikationskanal mit dem höchstwertigen Bit zuerst kommt einer Multiplikation mit X^{n-k} gleich, wenn anschliessend noch die Bits des Restpolynoms eingespiesen werden.

- Schalten Sie das Gate aus.
- Ist das letzte Bit des Nachrichtenworts eingespeist, entspricht der momentane Inhalt des Shiftregister den Bits des Restpolynoms. Speisen Sie auch diese in den Kommunikationskanal.

Somit haben Sie eine einfache Schaltung gesehen, die die nötigen Anforderungen für die Codierung einer Bitsequenz erfüllt.

3.4.2 Software

Um das Softwareprogramm zu entwerfen schauen wir uns nochmals die Polynomdivision etwas genauer an: Wir hatten in Aufgabe 3.1 d) folgendes Beispiel:

```
1 1 1 1 0 0 0  /  1 0 1 1  =  1 1 0 1
1 0 1 1 ----------------------+ | | |
-------                         | | |
  1 0 0 0 0 0                   | | |
  1 0 1 1    -------------------+ | |
  -------                         | |
    0 1 1 0 0                      | |
    0 0 0 0 ----------------------+ |
    -------                         |
      1 1 0 0                       |
      1 0 1 1 ----------------------+
      -------
        1 1 1      Rest =X^2 + X + 1
```

Um das Softwareprogramm zu implementieren, formulieren wir die Polynomdivision etwas um. Im Grunde machen Sie doch Folgendes:

Sie nehmen 1111000 und falls sich an der ersten Stelle eine 1 befindet, so addieren Sie dazu 1011. Dasselbe machen Sie mit dem Resultat der Addition, bis nur noch drei Datenbits übrig sind.

A3.4 Formulieren Sie das Ganze in Pseudocode

Lösungen der Aufgaben

Aufgabe 3.1.

Das Codewort erhalten Sie mit $c(X) = i(X) \cdot g(X)$. Somit sollten Sie $c_1 = (1+X^2) \cdot (1+X^2) = (1+X^4) = 1001$ und $c_2 = (1+X) \cdot (1+X^2) = (1+X+X^2+X^3) = 1111$ erhalten haben.

Aufgabe 3.2.

a) $c(X) = i(X)g(X) = X + X^2 + X^4$

b) $c(X) = i(X)g(X) = X + X^5 + X^6$

c) $i(X) = X + X^3$ in der systematischen Form

```
1 0 1 0 0 0 0  /  1 0 1 1  =  1 0 0 1
1 0 1 1 ----------------------+ | | |
-------                         | | |
  0 0 1 0                       | | |
  0 0 0 0    -------------------+ | |
  -------                         | |
    0 1 0 0                       | |
    0 0 0 0 ----------------------+ |
    -------                         |
      1 0 0 0                       |
      1 0 1 1 ----------------------+
      -------
        0 1 1      Rest =$0\cdot X^2 + X + 1$
```

Der Rest ergibt somit $X+1$. Somit sollten Sie $c(X) = b(X) + i(X)X^3 = 1 + X + X^4 + X^6$ erhalten.

d) $i(X) = 1 + X + X^2 + X^3$ in der systematischen Form

```
1 1 1 1 0 0 0  /  1 0 1 1  =  1 1 0 1
1 0 1 1 ----------------------+ | | |
```

```
-------                          | | |
  1 0 0 0                        | | |
  1 0 1 1    -------------------+ | |
  -------                          | |
    0 1 1 0                        | |
    0 0 0 0 ----------------------+ |
    -------                          |
      1 1 0 0                        |
      1 0 1 1 ----------------------+
      -------
        1 1 1      Rest =$X^2 + X + 1$
```

Der Rest ergibt somit X^2+X+1. Somit sollten Sie $c(X) = b(X)+i(X)X^3 = 1+X+X^2+X^3+X^4+X^5+X^6$ erhalten.

Aufgabe 3.3

a) Aus Kapitel 2 wissen Sie: Die Anzahl der erkannten Einzelfehler ist: $d-1=2$. Für die Anzahl der korrierbaren Fehler gilt: $\frac{d-1}{2}=1$.

b) Es können alle Burstfehler mit einer Länge bis zum Grade des Polynoms erkannt werden. In unserem Fall wäre das die Länge 3. Falls die Länge grösser als 3 ist, gilt dass ebenfalls ein grosser Anteil der Fehler erkannt werden.

Aufgabe 3.4

Laden Sie in ein Register das Datenwort mit den angehängten Nullen.
Laden Sie das Generatorpolynom ebenfalls in ein Register
for (i;i<n-k;i++)
    Schieben Sie das Datenegister ein Bit nach links.
    Falls Sie eine 1 finden, so bilden Sie die Addition zwischen dem Daten- und dem Generatorpolynomregister und speicheren Sie diese im Datenregister.
End

Die ersten k Bits des Datenregisters erhalten nun den Rest.

Lernkontrolle

L1

Berechnen Sie zu folgenden Nachrichtenworten die Codeworte des C(7,4) Codes mit $g(X) = 1 + X + X^3$

a) $i(X) = 1 + X$

b) $i(X) = 1 + X^2 + X^3$ in der systematischen Form

d) $i(X) = 1$ in der systematischen Form

L2

Welche Eigenschaften muss g(X) haben, um

- Einzelfehler
- eine ungerade Anzahl an Fehlerbits

zu erkennen?

L3

Aus [Lin04]: Zeichnen Sie die in Kapitel 3.4 gegebene Schaltung für das Polynom $g(X) = 1 + X + X^3$.

Lösungen zur Lernkontrolle

L1

a) $c(X) = i(X)g(X) = 1 + X^2 + X^3 + X^4$

b) $i(X) = X + X^3$ in der systematischen Form

```
1 1 0 1 0 0 0  /  1 0 1 1  =  1 1 1 1
1 0 1 1 ----------------------+ | | |
-------                         | | |
  1 1 0 0                       | | |
  1 0 1 1    --------------------+ | |
  -------                         | |
    1 1 1 0                       | |
    1 0 1 1 -----------------------+ |
    -------                         |
      1 0 1 0                       |
      1 0 1 1 -----------------------+
      -------
        0 0 1      Rest = 1
```

Der Rest ergibt somit 1. Somit sollten Sie $c(X) = b(X) + i(X)X^3 = 1 + X^3 + X^5 + X^6$ erhalten.

c) $i(X) = 1 + X + X^2 + X^3$ in der systematischen Form

```
0 0 0 1 0 0 0  /  1 0 1 1  =  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                        |
  1 0 1 1 ----------------------+
  -------
    0 1 1      Rest = 011
```

Somit sollten Sie $c(X) = b(X) + i(X)X^3 = 1 + X + X^3$ erhalten.

L2

- Um solche Fehler zu erkennen, genügt es, wenn weder X noch $X^k + 1$ durch das Polynom $g(X)$ teilbar sind. k entspricht dem maximalen Wert von $i - j$.
- Enthält das Generatorpolynom $g(X)$ $X + 1$ als Faktor, werden alle Fehler mit ungerader Bitzahl erkannt.

L3

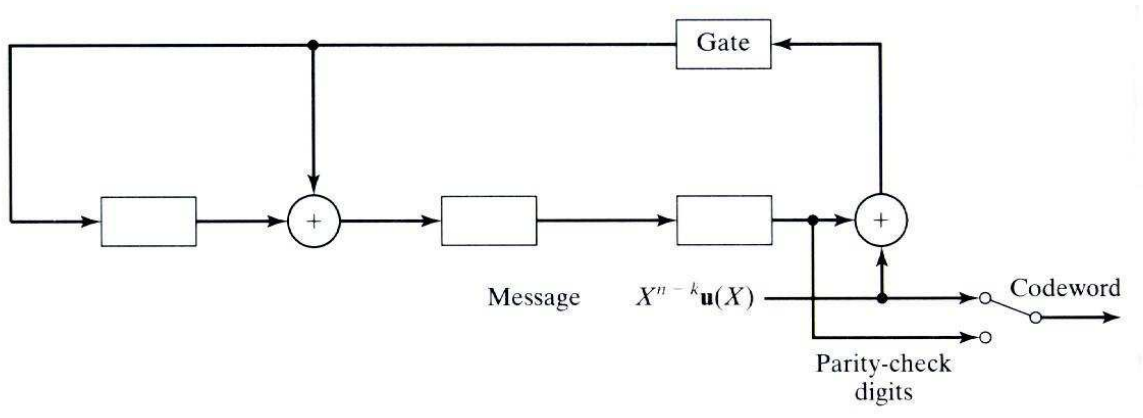

Abbildung 3.2: aus [Lin04, S. 148]

Kapitel 4

Ergänzende mathematische Grundlagen

4.1 Einführung und Lernziele

Was geschah bisher? In den vergangenen Kapiteln haben Sie die Rechnungen in der Modulo-2-Arithmetik bzw. mit Polynomen gelöst. Für die Modulo-2-Arithmetik hatten wir zwei Wahrheitstafeln eingeführt. Diese Tafeln legten den Grundstein für die Modulo-2-Arithmetik und zeigten, wie man binäre Zahlen miteinander zu verrechnen hat.

Was geschieht jetzt? Wir beschreiten nun einen neuen Weg: Wir verallgemeinern diese Methoden. Dies wird es uns im Kapitel 5 erlauben, bessere Codes zu verstehen und anzuwenden! Bei der Modulo-2-Arithmetik handelt es sich nämlich nur um einen Spezialfall einer allgemeinen Arithmetik.

Wie geschieht das? Wir werden

- in einem ersten Schritt die Modulo-2-Arithmetik allgemeiner darstellen und definieren.
- eine spezielle Art dieser Konstrukte genauer behandeln.
- den Umgang mit dieser speziellen Art üben.

Wir wollen Ihnen vor allem ein Verständnis vermitteln und uns weniger auf mathematische Details konzentrieren.

LERNZIELE

Leitidee

Bei den bis jetzt besprochenen Codes reichten wenige mathematische Grundlagen wie Polynom- oder Matrizenrechnung aus. Zur Verbesserung der Effizienz und Sicherheit von Codes sind weitere mathematische Kenntnisse von Nöten. Hier soll deshalb in die Theorie der Galois-Körper eingeführt werden.

Dispositionsziele

Wir verfolgen zwei Dispositionsziele:

- Sie erweitern Ihr Zahlenverständnis um die Klasse der Galois-Körper. Damit lernen Sie eine neue Art von Algebra kennen.
- Sie erarbeiten sich die Grundlagen für das Kapitel 5.

Operationalisierte Lernziele

Sie

- können die Eigenschaften von Körpern nennen und überprüfen
- kennen die Eigenschaften von Galois-Körpern und von ihren Elementen.
- sind in der Lage, einfache Galois-Körper herzuleiten.
- können Gleichungen über bestimmten Galois-Körpern lösen und Formeln vereinfachen.

4.2 Körper

4.2.1 Definition

Ein **Körper** ist definiert als eine Menge von Objekten (Zahlen), die mit zwei Operationen (Rechnungsarten) miteinander verknüpft sind. Im Normalfall geht man hier von Addition und Multiplikation aus. Ausserdem müssen diese beiden Operationen umkehrbar sein. Die fast wichtigste Eigenschaft ist aber, dass jede Verknüpfung von Objekten aus der Menge wieder ein Objekt dieser Menge ergeben muss (Abgeschlossenheit).

Bsp. 4.1 Ein Beispiel für einen Körper ist die Menge der reellen Zahlen mit den Verknüpfungen Addition und Multiplikation $\mathbb{R}(+,\times)$. Egal, welche reellen Zahlen Sie mit den beiden Operationen verknüpfen, Sie werden wiederum eine reelle Zahl erhalten. Die Addition lässt sich mit Subtrahieren, die Multiplikation mit Dividieren invertieren. Unser „normales“ Zahlenverständnis basiert also auf Körpern!

A4.1 Überlegen Sie sich kurz, ob auch die rationalen Zahlen einen Körper bilden. Ergibt das Verrechnen zweier rationaler Zahlen wiederum eine rationale Zahl? Zeigen Sie dies anhand einiger Beispiele, argumentieren Sie also ohne grosse Mathematik.
Zur Erinnerung: Rationale Zahlen sind all jene Zahlen, die sich als Bruch darstellen lassen, z.B. $\frac{3}{4}$.

Im Gegensatz zu Bsp. 4.1, wo wir Ihnen einen unendlich grossen Körper vorgestellt hatten, verwendet man in der Codierungstheorie **endliche Körper**. Diese besitzen eine definierte Anzahl an Elementen. Man nennt diese Körper zu Ehren des französischen Mathematikers Evariste Galois **Galois-Körper**.

Notation: Den Galois-Körper mit q Elementen bezeichntet man mit $GF(q)$ oder $\mathbb{F}_q$.

Bsp. 4.2 Die im Kapitel 1 eingeführte Modulo-2-Arithmetik bildet ebenfalls einen Körper! Wir rechneten damals mit 2 Elementen, nämlich 0 und 1. Als Verknüpfungen haben wir Addition und Multiplikation verwendet, die wir beide mit Wahrheitstafeln eingeführt hatten.

4.2.2 Eigenschaften

Anhand dieser Modulo-2-Arithmetik erläutern wir Ihnen nun die Eigenschaften eines (endlichen) Körpers. Wir geben zu den Definitionen (aus [Sweeney91]) immer gerade das Entsprechende aus der Modulo-2-Arithmetik an.

1. *Es sind zwei Operationen definiert.*

   Das sind in unserem Fall die Addition und Multiplikation aus den Wahrheitstafeln.

2. *Die Verknüpfung zweier Elemente mit diesen Operationen ergibt wieder ein Element des Körpers (Abgeschlossenheit).*

   Dies erkennt man leicht an den Wahrheitstafeln, so ist z.B. $1+1=0$, also wieder ein Element des Körpers.

3. *Ein Element ist das Null-Element, so dass $a+0=a$ für alle a aus dem Körper.*

   Wir haben hierfür das Element 0. So ist z.B. $1+0=1$.

4. *Ein Element ist das Eins-Element, so dass $a \cdot 1 = a$ für alle a aus dem Körper.*

   In der Modulo-2-Arithmetik ist das Eins-Element gleich 1. Ein anderes Beispiel wäre bei Matrizen die Einheitsmatrix als Einselement.

5. *Für jedes Element a aus dem Körper existiert ein additiv inverses Element $-a$, so dass $a+(-a)=0$.*

   In der Modulo-2-Arithmetik ist jedes Element auch gleich sein inverses, so ist z.B. $1+(-1)=1-1=1+1=0$.

   

   Wir erinnern Sie daran, **dass Addition und Subtraktion ja genau gleich sind!**

6. *Für jedes Nicht-Null Element a existiert ein multiplikativ inverses Element a^{-1}, so dass $a \cdot a^{-1} = 1$.*

   Das einzige Nicht-Null Element in der Modulo-2-Arithmetik ist 1. Auch hier ist das multiplikativ inverse Element genau gleich dem Element, $1 \cdot 1 = 1$.

7. *Es gelten die drei folgenden Gesetze:*

$$\begin{array}{llcllcl}
\textit{Assoziativgesetz} & a+(b+c) & = & (a+b)+c & \textit{und} & a\cdot(b\cdot c) & = & (a\cdot b)\cdot c \\
\textit{Kommutativgesetz} & a+b & = & b+a & \textit{und} & a\cdot b & = & b\cdot a \\
\textit{Distributivgesetz} & a\cdot(b+c) & = & a\cdot b+a\cdot c & & & &
\end{array}$$

Alle diese Gesetze gelten ebenfalls für die Modulo-2-Arithmetik. Als Beispiel das Kommutativgesetz: $1+0=1=0+1$ und $1\cdot 0=0=0\cdot 1$.

Nur endliche Körper, deren Grösse eine Primzahl oder ein Vielfaches einer Primzahl ist, können diese 7 Eigenschaften erfüllen.

A4.2 Zeigen Sie, dass die Menge $\{0,1,2,3,4,5\}$ kein endlicher Körper bezüglich Addition und Multiplikation ist. Beachten Sie, dass die Operationen natürlich modulo 6 zu verstehen sind, z.B. $5+1=0$!
Tipp: Schauen Sie sich z.B. die additiv inversen Elemente an.

Sie sehen, dass es sich bei der Modulo-2-Arithmetik genau genommen um einen Galois Körper der Grösse 2 mit den Verknüpfungen Addition und Multiplikation handelt, also $\{0,1\}(\oplus,\otimes)$.
Achtung: Sie müssen klar zwischen *logischen* und *algebraischen* Operationen unterscheiden. Während wir bei Gattern logische Operationen wie AND oder EXOR verwenden, sind es bei den Galoiskörpern algebraische Operationen wie Addition oder Multiplikation.
Bei der logischen Operation *AND* handelt es sich z.B. um die algebraische Operation *Multiplikation*, und bei der logischen Operation *EXOR* um die algebraische Operation *Addition*.

Wir fassen nochmals zusammen:

- Körper müssen die oben genannten 7 Eigenschaften erfüllen.
- Die Grösse von endlichen Körpern ist entweder eine Primzahl oder ein Vielfaches einer Primzahl.
- Ein Galois-Körper $GF(q)$ ist ein endlicher Körper der Grösse q.
- Die uns mittlerweile gut bekannte Modulo-2-Arithmetik bildet den Galois-Körper $GF(2)$.

4.3 Galois-Körper $GF(2^3)$

4.3.1 Motivation

Endliche Körper können neben den uns bekannten „Ziffern“ auch andere Elemente enthalten. Um diese Elemente zu veranschaulichen, machen wir einen Vergleich:
Die Gleichung $4X^2 + 1 = 0$ besitzt keine reelle Lösung. Um sie trotzdem lösen zu können, erweiterte man den Körper der reellen Zahlen zu jenem der komplexen Zahlen, indem man $i = \sqrt{-1}$ und weitere Rechenregeln einführte.
Wechseln wir nun in den binären Zahlenraum und betrachten uns die Gleichung $X^3 + X + 1 = 0$. Obwohl wir als Koeffizienten binäre Zahlen genommen haben, existiert keine Lösung im binären Zahlenbereich 0,1. Das kann man durch Einsetzen leicht erkennen. Um diese Gleichung lösen zu können, muss man auch hier eine „neue“ Zahl einführen. Wir nennen diese α . Wir wissen nun aber nicht viel über α, sondern nur, dass α die Gleichung $\alpha^3 + \alpha + 1 = 0$ erfüllt...

4.3.2 Konstruktion

Wir erweitern jetzt den Galois-Körper $GF(2)$. Anhand der Gleichung

$$\alpha^3 + \alpha + 1 = 0$$

lässt sich schrittweise der Galois-Körper $GF(8)$ herleiten (siehe Tabelle 4.1). Dieser Körper verfügt über $8 = 2^3$ Elemente, weshalb wir ihn in Zukunft als $GF(2^3)$ bezeichnen. Mit der 2 zeigen wir so gleichzeitig, dass dieser Körper nichts anderes als eine Erweiterung der Modulo-2-Arithmetik ist. Die im Kapitel 1 eingeführten Wahrheitstafeln gelten also weiterhin! Ausserdem sind natürlich 0 und 1 ebenfalls Elemente dieses Galois-Körpers.
Die weiteren Elemente des Galois-Körpers $GF(2^3)$ kann man nun durch **fortlaufendes Potenzieren** von α konstruieren. So erhalten wir zu 0, 1 und α also

- α^2
- α^3

  Dieses Element können wir aber weiter vereinfachen, denn wir kennen ja die Gleichung $\alpha^3 + \alpha + 1 = 0$.

Wie bereits bei der Modulo-2-Arithmetik ist auch in den Galois-Körpern die **Addition und Subtraktion gleich**, so z.B. $\alpha+\alpha=\alpha\cdot(1+1)=\alpha\cdot(0)=0$.

Diese Eigenschaft gilt für alle Elemente. Die ursprüngliche Gleichung lässt sich also umschreiben zu

$$\alpha^3=\alpha+1$$

Somit können wir α^3 auf eine Summme von kleineren Elementen zurückführen. Man bezeichnet diese Darstellung als **Polynomdarstellung**.

- Das nächste Element α^4 wird zu (Einsetzen von $\alpha^3=\alpha+1$)

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

die weiteren Elemente α^5 und α^6 lassen sich analog herleiten.

$$\alpha^5=\ldots=\alpha^2+\alpha+1 \tag{4.1}$$

$$\alpha^6=\ldots=\alpha^2+1 \tag{4.2}$$

- Als nächstes Element erhält man

$$\alpha^7=\alpha\cdot\alpha^6=\alpha\cdot(\alpha^2+1)=\alpha^3+\alpha=(\alpha+1)+\alpha=1$$

Dieses Element ist aber bereits Bestandteil des Galois-Körpers. Also kann man hier abbrechen, da alle weiteren Potenzen sich auf die 8 Elemente zurückführen lassen.

$$\begin{array}{lcl}
0 & & \\
1 & & \\
\alpha & & \\
\alpha^2 & & \\
\alpha^3 & = & \alpha+1 \\
\alpha^4 & = & \alpha^2+\alpha \\
\alpha^5 & = & \alpha^2+\alpha+1 \\
\alpha^6 & = & \alpha^2+1
\end{array}$$

Tabelle 4.1: **Elemente des Galois-Körpers** $GF(2^3)$

Wir wiederholen nochmals die bei der Herleitung offensichtlich gewordenen Tatsachen:

- Addition und Subtraktion sind gleich!
- Alle Potenzen von α grösser als a^2 lassen sich auch als Polynome kleinerer Ordnung darstellen. So lässt sich auch die Addition einfach durchführen, z.B. $\alpha+\alpha^5 = (\alpha)+(\alpha^2+\alpha+1) = \alpha^2+1 = \alpha^6$.
- $\alpha^7 = 1$, d.h alle höheren Potenzen von α lassen sich auf die 8 angegebenen Elemente reduzieren.

A4.3 Berechnen Sie in $GF(2^3)$

a) die Polynomdarstellung von α^5 und α^6, leiten Sie also die Gleichungen (4.1) und (4.2) her.

b) $\alpha^5+\alpha+1$

Sie finden weitere Galois-Körper, die Sie im Rest des Leitprogrammes brauchen werden, im Anhang A.

4.3.3 Eigenschaften

Primitive Elemente, Ordnung eines Elementes

Ein Element eines Galois-Körpers ist **primitiv**, wenn man aus ihm durch Potenzieren ausser 0 alle anderen Elemente konstruieren kann.

Bsp. 4.3 Im Galois-Körper $GF(2^3)$ ist α^5 ein primitives Element. Sei β ein beliebiges Element des Galois-Körpers. Wenn wir $\beta = \alpha^5$ setzen, so ergibt das fortlaufende Potenzieren von β:

$$\begin{aligned}
\beta^2 &= \alpha^{10} = \alpha^7 \cdot \alpha^3 = \alpha^3 \\
\beta^3 &= \alpha^{15} = \alpha^7 \cdot \alpha^7 \cdot \alpha = \alpha \\
\beta^4 &= \alpha^{20} = \alpha^6 \\
\beta^5 &= \alpha^{25} = \alpha^4 \\
\beta^6 &= \alpha^{30} = \alpha^2 \\
\beta^7 &= \alpha^{35} = (\alpha^7)^5 = 1
\end{aligned}$$

Es lassen sich also alle Elemente, abgesehen von 0, aus α^5 herleiten.
Ob ein Element eines Galois-Körpers primitiv ist, lässt sich auch aus seiner Ordnung herleiten, wie wir gleich zeigen werden.

Definition: Sei β ein beliebiges Element eines Galois-Körpers. Dann definiert man die **Ordnung** n dieses Elementes als die kleinste natürliche Zahl, so dass $\beta^n = 1$.

Es gilt für alle primitiven Elemente aus $GF(2^m)$, dass ihre Ordnung $2^m - 1$ ist.

Bsp. 4.4 Die Ordnung von α^5 im Galois-Körper $GF(2^3)$ ergibt sich zu 7, denn erst $(\alpha^5)^7 = \alpha^{35}$ ergibt 1. Da $7 = 2^m - 1 = 2^3 - 1 = 7$ ist, wissen wir auch gleich, dass α^5 ein primitives Element ist!

A4.4 Berechnen Sie die Ordnung der restlichen Elemente (ausser 0) von $GF(2^3)$. Was fällt Ihnen dabei auf?

Minimales Polynom

Wichtiger Bestandteil der Galois-Körper sind die minimalen Polynome. Wir werden diese im nächsten Kapitel verwenden, um Codes zu konstruieren. Zu jedem Element eines Galois-Körpers existiert ein solches minimales Polynom. Ausserdem können mehrere Elemente das gleiche minimale Polynom haben.
Wir zeigen Ihnen das allgemeine Vorgehen gleich zusammen mit einem Beispiel:

Bsp. 4.5 Wir bestimmen das minimale Polynom von α^2. Das Vorgehen ist sehr systematisch:

1. Sei β ein beliebiges Element eines Galois-Körpers. Wir wählen $\beta = \alpha^2$.
2. Man suche nun die **konjugierten Elemente** von β. Das sind diejenigen Elemente, die aus β durch **fortlaufendes Quadrieren** entstehen, also:

$$\beta^2, \beta^4, \beta^8, \beta^{16}, \ldots$$

Abgebrochen wird das Quadrieren, wenn man wieder das ursprüngliche Element, also β erhält. In unserem Fall ergibt sich

$$\begin{aligned}
\beta^2 = (\alpha^2)^2 &= \alpha^4 \\
\beta^4 = (\alpha^2)^4 &= \alpha^8 = \alpha^7 \cdot \alpha = \alpha \\
\beta^8 = (\alpha^2)^8 &= \alpha^{16} = \alpha^7 \cdot \alpha^7 \cdot \alpha^2 = \alpha^2 = \beta
\end{aligned}$$

Die konjugierten Elemente von α^2 sind also α und α^4.

3. Nun kann man bereits das **minimale Polynom** m_β bilden, mit:

$$m_\beta(X) = (X+\beta) \cdot (X+\beta^2) \cdot (X+\beta^4) \cdot (X+\beta^8) \cdot \ldots$$

Für uns heisst das

$$m_{\alpha^2}(X) = m_2(X) = (X+\alpha) \cdot (X+\alpha^2) \cdot (X+\alpha^4) = \ldots = X^3 + X + 1$$

Dabei kürzen wir hier die Schreibweise ab mit $m_{\alpha^i} = m_i$.

Bsp. 4.6 Wir setzen α in sein minimales Polynom ein:

$$m_1(\alpha) = \alpha^3 + \alpha + 1 = (\alpha+1) + \alpha + 1 = (\alpha+\alpha) + (1+1) = 0$$

Genau gleich würde sich durch Einsetzen von α^2 und α^4 die Lösung 0 ergeben. Man bezeichnet das minale Polynom deshalb als minimal, weil es das kleinste Polynom ist, dass die konjugierten Elemente als Nullstellen hat.

A4.5 Wir haben also für drei Elemente bereits das minimale Polynom hergeleitet. Abgesehen von den beiden minimalen Polynomen von 0 und 1 existiert noch ein zusätzliches minimales Polynom im Galois-Körper $GF(2^3)$. Bestimmen Sie dieses! Gehen Sie dabei vom Element α^3 aus.

4.4 Lösen von Gleichungen in Galois-Körpern

Um Ihnen die Galois-Körper noch etwas näher zu bringen, werden Sie in diesem Kapitel Gleichungen in Galois-Körpern lösen. Sie werden in diesem Kapitel die Gemeinsamkeiten erkennen, die unser „ganz normaler“ Zahlenkörper $\mathbb{R}$ mit den Galois-Körpern hat.

Bsp. 4.7 Wir lösen die Gleichung $\alpha^3 \cdot X + 1 = 0$ in $GF(2^3)$.

$$\alpha^3 \cdot X = 1$$
$$X = \frac{1}{\alpha^3} = \frac{\alpha^7}{\alpha^3} = \alpha^4$$

Hier haben wir im Zähler 1 durch α^7 ersetzt und anschliessend gekürzt.

Bemerkung: Man bezeichnet jene Elemente, deren Einsetzen in eine Formel 0 ergibt, als **Wurzeln** oder **Nullstellen** der Formel. Hier ist also α^4 eine Wurzel der Formel $\alpha^3 \cdot X + 1$.

A4.6 Lösen Sie die Gleichung $\alpha^2 \cdot X + \alpha^3 = \alpha^4$.

Zum Abschluss dieses kurzen Kapitels haben Sie noch die Gelegenheit ein Gleichungssystem 2. Ordnung in $GF(2^3)$ und eine Gleichung in $GF(2^4)$ zu lösen.

A4.7 Lösen Sie in $GF(2^3)$ das Gleichungssystem

$$\alpha X + \alpha^4 Y = \alpha^3$$
$$X + \alpha^7 Y = \alpha^{11}$$

Tipp: Lösen Sie die zweite Gleichung nach X auf und setzen Sie sie in die erste Gleichung ein.

4.8 Lösen Sie in $GF(2^4)$ die Gleichung

$$x^3 + \alpha^5 x^2 + \alpha x + \alpha^{10} = 0.$$

Tipp: Versuchen Sie die Lösung durch Einsetzen zu berechnen. Sie finden den Galois-Körper $GF(2^4)$ im Anhang A.

Beachten Sie, dass in $GF(2^4)$ gilt: $\alpha^{15} = 1$.

Voilà! Widmen Sie sich nun der Lernkontrolle. Sie werden dort insbesondere den Umgang mit $GF(2^4)$ etwas weiter vertiefen. Nach dem bestandenen Tutortest sind Sie somit reif für Kapitel 5!

Lösungen der Aufgaben

Aufgabe 4.1 Welche rationalen Zahlen Sie auch immer miteinander verknüpfen, Sie werden als Resultat eine rationale Zahl erhalten. Zum Beispiel ergibt $\frac{2}{3} \cdot \frac{4}{5} = \frac{2 \cdot 4}{2 \cdot 5} = \frac{8}{15}$. Sie sollten mit ihren Beispielen zum gleichen Schluss gekommen sein. Also bilden auch die rationalen Zahlen einen Körper.

Aufgabe 4.2 Es lassen sich viele Argumente finden, warum GF(6) nicht existiert. So hat zum Beispiel 2 kein multiplikativ inverses Element. Es gibt also kein Element X in der Menge, so dass $2 \cdot X = 1$.

Aufgabe 4.3

a) $\alpha^5 = \alpha \cdot \alpha^4 = \alpha \cdot (\alpha^2 + \alpha) = \alpha^3 + \alpha^2 = \alpha^2 + \alpha + 1$

$\alpha^6 = \alpha \cdot \alpha^5 = \alpha \cdot (\alpha^2 + \alpha + 1) = \alpha^3 + \alpha^2 + \alpha = (\alpha + 1) + \alpha^2 + \alpha = \alpha^2 + 1$

b) $\alpha^5 + \alpha^2 + 1 = (\alpha^2 + \alpha + 1) + \alpha^2 + 1 = \alpha$

Aufgabe 4.4 Ausser 1 haben alle Elemente die Ordnung 7, sind also primitiv!
Wir geben Ihnen für 2 Elemente, α^2 und α^4 den Lösungsweg an: Links für $\beta = \alpha^2$ und rechts für $\beta = \alpha^4$.

$$\begin{aligned}
\beta &= \alpha^2 \\
\beta^2 &= \alpha^4 \\
\beta^3 &= \alpha^6 \\
\beta^4 &= \alpha^8 = \alpha^7 \cdot \alpha = \alpha \\
\beta^5 &= \alpha^{10} = \alpha^7 \cdot \alpha^3 = \alpha^3 \\
\beta^6 &= \alpha^{12} = \alpha^7 \cdot \alpha^5 = \alpha^5 \\
\beta^7 &= \alpha^{14} = \alpha^7 \cdot \alpha^7 = 1
\end{aligned}
\qquad
\begin{aligned}
\beta &= \alpha^4 \\
\beta^2 &= \alpha^8 = \alpha^7 \cdot \alpha = \alpha \\
\beta^3 &= \alpha^{12} = \alpha^7 \cdot \alpha^5 = \alpha^5 \\
\beta^4 &= \alpha^{16} = (\alpha^7)^2 \cdot \alpha^2 = \alpha^2 \\
\beta^5 &= \alpha^{20} = (\alpha^7)^2 \cdot \alpha^6 = \alpha^6 \\
\beta^6 &= \alpha^{24} = (\alpha^7)^3 \cdot \alpha^3 = \alpha^3 \\
\beta^7 &= \alpha^{28} = (\alpha^7)^4 = 1
\end{aligned}$$

Aufgabe 4.5

1. Wir wählen $\beta = \alpha^3$.

2. Durch fortlaufendes Quadrieren ergibt sich

$$\begin{aligned}
\beta^2 = (\alpha^3)^2 &= \alpha^6 \\
\beta^4 = (\alpha^3)^4 &= \alpha^{12} = \alpha^7 \cdot \alpha^5 = \alpha^5 \\
\beta^8 = (\alpha^3)^8 &= \alpha^{24} = \alpha^{21} \cdot \alpha^3 = \alpha^3
\end{aligned}$$

   Die konjugierten Elemente von α^3 sind also α^5 und α^6

3. Das minimale Polynom ergibt sich zu

$$m_{\alpha^3} = m_3 = (X + \alpha^3) \cdot (X + \alpha^5) \cdot (X + \alpha^6) = \ldots = X^3 + X^2 + 1$$

Zur Übersicht finden Sie in Tabelle 4.2 nochmals alle Elemente aus $GF(2^3)$ mit ihren minimalen Polynomen.

Element(e)	Minimales Polynom
0	X
1	$X+1$
α, α^2, α^4	X^3+X+1
α^3, α^5, α^6	X^3+X^2+1

Tabelle 4.2: **Minimale Polynome des Galois-Körpers** $GF(2^3)$

Aufgabe 4.6

$$\alpha^2 \cdot X = \alpha^4 + \alpha^3$$
$$X = \frac{\alpha^4}{\alpha^2} = \frac{\alpha^3}{\alpha^2} = \alpha^2 + \alpha = \alpha^4$$

Im letzten Schritt haben wir ausgenutzt, dass $\alpha^2 + \alpha$ die Polynomdarstellung von α^4 ist. Es sind aber natürlich beide Resultate richtig.

Aufgabe 4.7 Wir lösen die zweite Gleichung nach X auf und vereinfachen Sie. Beachten Sie, dass $\alpha^7 = 1$

$$X = \alpha^7 Y + \alpha^{11} = Y + \alpha^4$$

Wir setzen nun in der ersten Gleichung ein.

$$\begin{aligned}
\alpha \cdot (Y + \alpha^4) + \alpha^4 Y &= \alpha^3 \\
\alpha \cdot Y + \alpha^5 + \alpha^4 Y &= \alpha^3 \\
(\alpha^4 + \alpha) \cdot Y &= \alpha^5 + \alpha^3 \\
(\alpha^2 + \alpha + \alpha) \cdot Y &= \alpha^5 + \alpha^3 \\
\alpha^2 \cdot Y &= \alpha^5 + \alpha^3 \\
Y = \alpha^3 + \alpha &= (\alpha + 1) + \alpha = 1
\end{aligned}$$

Wir gehen wieder zur Gleichung 2 und setzen $Y = 1$ ein

$$X = Y + \alpha^4 = 1 + \alpha^4 = 1 + \alpha^2 + \alpha = \alpha^5$$

Die Lösungen sind also $X = \alpha^5$ und $Y = 1$.

Aufgabe 4.8 α, α^3 und α^6. Die Lösung ergibt sich durch Einsetzen. Für z.B. $x = \alpha^6$ berechnet sich die Gleichung wie folgt:

$$\begin{aligned}
&x^3 + \alpha^5 x^2 + \alpha x + \alpha^{10} = (\alpha^6)^3 + \alpha^5 (\alpha^6)^2 + \alpha(\alpha^6) + \alpha^{10} = \alpha^{18} + \alpha^{17} + \alpha^7 + \alpha^{10} = \\
&= \alpha^3 + \alpha^2 + \alpha^7 + \alpha^{10} = \alpha^3 + \alpha^2 + (\alpha^3 + \alpha + 1) + (\alpha^2 + \alpha + 1) = 0
\end{aligned}$$

Lernkontrolle

L.1

Berechnen Sie in $GF(2^3)$ die Formel $d(X) = X^7 + X^5 + X^4 + 1$ für

a) $X = \alpha$

b) $X = \alpha^3$

L.2

Lösen Sie die Gleichung $\alpha X^2 + \alpha^2 X + \alpha^2 + 1 = 0$. Suchen Sie die Lösung durch Einsetzen der Elemente des Galois-Körpers $GF(2^3)$.

L.3

Es existieren natürlich weitere Galois-Körper, so zum Beispiel $GF(2^4)$. In Bsp. 4.3 haben wir Ihnen gezeigt, wie sich aus einem primitiven Element der ganze Galois-Körper herleiten lässt. Versuchen Sie nun selbst, mit Hilfe der Tabelle „Elemente“ auf Seite 79 im Anhang A, die Ordnung der Elemente in $GF(2^4)$ herzuleiten. Bestimmen Sie ausserdem, welche dieser Elemente primitiv sind.
Hier 2 Tipps:

- Formeln lassen sich vereinfachen, indem man Elemente mit ihrer Polynomdarstellung schreibt, z.B. in $GF(2^4)$ für α^6 den Ausdruck $\alpha^3 + \alpha^2$ verwendet.
- Die Ordnung der primitiven Elemente eines Galois-Körpers $GF(2^m)$ ist $2^m - 1$.

L.4

Lösen Sie in $GF(2^3)$ das Gleichungssystem

$$\alpha^2 X + \alpha^5 Y = \alpha^3$$
$$\alpha^3 X + \alpha Y = 0$$

Lösungen der Lernkontrolle

L.1

a) $d(\alpha) = \alpha^7 + \alpha^5 + \alpha^4 + 1 = (1) + (\alpha^2 + \alpha + 1) + (\alpha^2 + 1) + 1 = ... = \alpha$

b) $d(\alpha^3) = \alpha^{21} + \alpha^{15} + \alpha^{12} + 1 = (1) + (\alpha) + (\alpha^5) + 1 = ... = \alpha^2 + 1 = \alpha^4$

Beachten Sie, dass man Ausdrücke wie α^{13} umschreiben kann zu $\alpha^{13} = \alpha^7 \cdot \alpha^6 = 1 \cdot \alpha^6 = \alpha^6$.

L.2

α^2 und α^5.

L.3

Die primitiven Elemente haben in $GF(2^4)$ alle die Ordnung $2^4 - 1 = 15$. Die folgende Tabelle zeigt die Ordnung aller Elemente.

Element(e)	**Ordnung**
1	1
α, α^2, α^4, α^8	15
α^3, α^6, α^9, α^{12}	3
α^5, α^{10}	5
α^7, α^{11}, α^{13}, α^{14}	15

Also sind α, α^2, α^4, α^7, α^8, α^{11}, α^{13} und α^{14} primitiv.

L.4

Die Lösungen sind $X = \alpha^4$ und $Y = \alpha^6$. Diese ergeben sich zum Beispiel wie folgt:
Wir lösen die zweite Gleichung nach Y auf: $Y = \alpha^2 X$. Dies setzen wir nun in die erste Gleichung ein: $\alpha^2 X + \alpha^5 \cdot (\alpha^2 X) = \alpha^2 X + \alpha^7 X = \alpha^2 X + X = (\alpha^2 + 1)X = \alpha^6 X = \alpha^3$. Daraus folgt $X = \frac{\alpha^3}{\alpha^6} = \alpha^{-3} = \alpha^{-3} \cdot \alpha^7 = \alpha^4$. Nun können wir dieses Resultat in die zweite Gleichung einsetzen und erhalten: $\alpha^7 + \alpha Y = 1 + Y = 0$. Also ist $Y = 1$.

Kapitel 5

Additivum: BCH-Codes

5.1 Einführung und Lernziele

Was geschah bisher? In Kapitel 4 haben Sie die Galois-Körper kennengelernt. Sie können mit Elementen aus Galois-Körpern mathematische Gleichungen und Formeln berechnen.
Was geschieht jetzt? In diesem Kapitel benutzen Sie die Galois-Körper, um eine neue Klasse von Codes zu konstruieren. Diese Codes sind bekannt unter dem Namen **BCH-Codes**. BCH steht für die Entwickler dieses Codes, namentlich **Bose, Ray-Chaudhuri und Hocquenghem**.
Wie geschieht das? Wir gehen genau so vor, wie Sie in einem Übertragungsysteme ein Wort senden würden:

- In einem ersten Schritt zeigen wir Ihnen, wie man aus einem gegebenen Galois-Körper für eine verlangte Anforderung einen passenden Code **konstruiert**.
- Anschliessend werden Sie selber einen solchen Code konstruieren
- Diesen konstruierten Code verwenden Sie dann zur **Codierung**, indem Sie nach den in Kapitel 3 vorgestellten Verfahren vorgehen.
- Wir zeigen Ihnen dann, wie Sie die empfangene Sequenz wieder **decodieren und korrigieren** können und geben Ihnen die Gelegenheit, dies anhand eines empfangenen Codewortes zu üben.

BCH-Codes werden zum Beispiel bei DAB (Digital Audio Broadcasting) verwendet. DAB soll bis Ende 2008 in der ganzen Schweiz zu empfangen sein.

LERNZIELE

Leitidee

Oft ist es wichtig, für gegebene Spezifikationen einen genau passenden Algorithmus zu entwickeln. BCH-Codes erlauben es, ganz spezifische Anforderungen zu erfüllen.

Dispositionsziele

Sie erkennen, wie sich anhand mathematischer Konstruke praktikable Codes erstellen lassen.

Operationalisierte Lernziele

Sie können

- mit einem gegebenen Galois-Körper einen BCH-Code für eine verlangte Spezifikation konstruieren.
- diesen Code zur Codierung verwenden.
- Fehler bei der Decodierung erkennen und korrigieren.
- die zusätzlichen Eigenschaften von BCH-Codes nennen.

5.2 Konstruktion

Nach der vorangegangenen Einführung in endliche Körper wollen wir diese nun das erste Mal anwenden. Die BCH-Codes gehören zu den zyklischen Codes. Sie werden so konstruiert, dass sie **eine definierte Anzahl von Fehlern korrigieren können**. Und darin besteht auch ihr grosser Vorteil!

Es gilt, dass für jeden positiven Wert m und t ein BCH-Code mit folgenden **Eigenschaften** existiert:

- Blocklänge: $n = 2^m - 1$
- Anzahl korrigierbare Fehler: $t < 2^{m-1}$

Aus m folgt direkt der zu verwendende Galois-Körper $GF(2^m)$.

Überlegen Sie kurz: Was sagt die Bedingung $t < 2^{m-1}$ aus? Genau: Um mehr Fehler korrigieren zu können, muss man grössere Galois-Körper nehmen. Sie können sich gut vorstellen, dass die Sache also mit zunehmender Fehleranzahl nicht einfacher wird!

Das **Generatorpolynom** $g(X)$ dieses Codes ist das Polynom mit tiefstem Grad, das als Nullstellen $\alpha, \alpha^2, \alpha^3, \ldots, \alpha^{2t}$ hat. Man definiert das Polynom dabei natürlich über dem gleichen Galois-Körper $GF(2^m)$.

Nimmt man $m_i(X)$ als das minimale Polynom für die obigen α^i, dann ist $g(X)$ **das kleinste gemeinsame Vielfache der einzelnen minimalen Polynome (least commom multiple (LCM))**:

$$g(X) = LCM(m_1(X), m_2(X), \ldots, m_{2t}(X))$$

Das kleinste gemeinsame Vielfache ist hier bezogen auf Polynome. D.h. wenn 2 Polynome $m_i(X)$ und $m_j(X)$ gleich sind, so werden sie nur einmal verrechnet.

Dies lässt sich weiter vereinfachen zu

$$g(X) = LCM(m_1(X), m_3(X), \ldots, m_{2t-1}(X)), \tag{5.1}$$

was wir aber nicht beweisen werden. Die Betrachtung der $m_i(X)$ für i ungerade reicht also aus.

So: Das Ganze wirkt vom praktischen Standpunkt her eigentlich ziemlich einfach, oder? Um Ihnen das zu ermöglichen, haben wir Sie im Kapitel 4 auch etwas gefordert. Jetzt wird die Konstruktion von Codes nämlich schon fast „bahnbrechend“ einfach! Sie müssen nur die Formel (5.1) im Hinterkopf behalten.

Zur Erinnerung: Sie finden alle in diesem Kapitel verwendeten Galois-Körper im Anhang A zum Nachschlagen.

Bsp. 5.1 Wir illustrieren Ihnen anhand eines Beispieles Schritt für Schritt das Vorgehen, um einen BCH-Code zu konstruieren. Wir wollen mit dem Galois-Körper $GF(2^3)$ einen 1-Fehler-korrigierenden Code herleiten.

1. Wir wollen 1 Fehler korrigieren, d.h. wir wählen $t = 1$.
2. Der Galois-Körper $GF(2^3)$ ist in der Tabelle 4.1 angegeben. Sei α das primitive Element.
3. Aus Gleichung 5.1 folgt: $g(X) = LCM(m_1(X))$
4. In Tabelle 4.2 sind die minimalen Polynome des Galois-Körpers $GF(2^3)$ angegeben. Es folgt $m_1(X) = X^3 + X + 1$, also
5. $g(X) = X^3 + X + 1$

Das Generatorpolynom dieses Codes sollte Ihnen bekannt vorkommen: Genau diesen Code haben Sie im Kapitel Zyklische Codes schon angetroffen. Er hat eine Blocklänge von 7 und eine Informationslänge von maximal 4 Bit. Wir bezeichnen ihn jetzt als (7,4)-BCH Code.

A5.1 Konstruieren Sie einen 2-Fehler-korrigierenden BCH Code über dem Galois-Körper $GF(2^4)$ (siehe Anhang A, Seite 79). Welche Blocklänge und maximale Informationslänge besitzt er?
Um welchen BCH-Code handelt es sich?

(Blocklänge, Informationslänge): .....

Generatorpolynom: .....

Tipp: Die Blocklänge n berechnet sich nach der Formel $n = 2^m - 1$, wobei m aus dem verwendeten Galois-Körper $GF(2^m)$ folgt.

5.3 Codierung

Wir repetieren in diesem Abschnitt nochmals das Vorgehen aus Kapitel 3. Dort haben Sie gelernt, wie man bei den zyklischen Codes mit dem Generatorpolynom Codewörter erzeugt.
Wir halten uns bewusst kurz, denn das Augenmerk in diesem ganzen Kapitel 5 soll auf der Konstruktion des Codes und auf der Decodierung liegen.
Das Codewort $c(X)$ bildet man mit

$$c(X) = i(X) \cdot g(X),$$

wobei $i(X)$ das Informationspolynom ist.

Bsp. 5.2 Sei $i(X) = X + 1$. Dann ergibt sich mit dem (7,4)-BCH Code $c(X)$ zu

$$c(X) = i(X) \cdot g(X) = (X+1) \cdot (X^3 + X + 1) = X^4 + X^3 + X^2 + 1$$

Wir verzichten hier auf eine weitere Aufgabe. Schauen Sie in Kapitel 3 nach, falls Sie sich nicht mehr sicher sind!

5.4 Decodierung und Fehlererkennung

Zur Decodierung und Fehlererkennung von empfangenen Wörtern stehen uns mehrere Methoden zur Verfügung. Wir verwenden die Einleuchtendste. Die Idee dahinter ist einfach zu verstehen: Aus dem empfangenen Wort bildet man mit gegebenen Regeln ein weiteres Polynom. Dieses führt schliesslich zur Position der Fehler.

Sei $c(X)$ das gesendete Codewort. Das beim Empfänger eingetroffene Wort bezeichnen wir mit $r(X)$. Es sei nun

$$r(X) = c(X) + e(X),$$

d.h. das empfangene Wort besteht aus dem eigentlichen Codewort $c(X)$ und dem Fehler $e(X)$. Das wissen Sie bereits aus Kapitel 3.

Das Generatorpolynom eines BCH-Codes, der t Fehler korrigieren kann, hat ja als Nullstellen α, α^2, α^3, ..., α^{2t}. Daraus folgt, dass auch $c(\alpha^i) = 0$ ist. Zur Erinnerung: $c(X) = i(X) \cdot g(X)$. Also sind die **Nullstellen von** $g(X)$ **auch Nullstellen von** $c(X)$.

Setzen wir also α in $r(X)$ ein. Wir erhalten

$$r(\alpha^i) = c(\alpha^i) + e(\alpha^i) = 0 + e(\alpha^i) = e(\alpha^i)$$

Also wird $r(\alpha^i)$ nur dann für alle α^i null sein, wenn kein Fehler aufgetreten ist. Aus diesen **Fehler-Syndromen** lässt sich nun die Position der/s Fehler/s eindeutig bestimmen.

A5.2 Überlegen Sie sich, wann diese Methode wohl nicht mehr funktioniert.

Fehlerbestimmung für BCH-Codes mit t=1

Wir nehmen als Beispiel den bekannten 1-Fehlerkorrigierenden (7,4)-BCH Code mit dem Generatorpolynom $g(X) = X^3 + X + 1$. Ein Codewort davon ist, wie Sie in Kapitel 3 erfahren haben, $c_1(X) = X^3 + X + 1$.

Sei nun der Fehler $e(X) = X^4$, d.h. das empfangene Wort ist $r(X) = X^4 + X^3 + X + 1$.
Setzen wir nun die Nullstellen des Generatorpolynoms α und α^2 in $r(X)$ ein, so folgt bereits aus α

$$r(\alpha) = \alpha^4 + \alpha^3 + \alpha + 1 = ... = \alpha^4$$

$r(\alpha)$ gibt die Stelle des Fehlers an! Hier also $\alpha^4 \Rightarrow X^4$. Dieses System versagt natürlich, wenn mehr als 1 Fehler auftritt.

Wir fassen zusammen: Bei 1-Fehlerkorrigierenden Codes gibt $r(\alpha)$ direkt die Position des Fehlers an!
Wenn man sich nun aber mit Codes beschäftigt, die 2 oder mehr Fehler korrigieren können, ist das Ganze nicht mehr so einfach. In diesen Fällen weisen die $r(\alpha^i)$ nicht mehr direkt auf den Fehler hin.
Für $t = 2$ ist die Vorgehensweise aber machbar. Wir beschränken uns deshalb bei der Decodierung auf BCH-Codes, die maximal 2 Fehler korrigieren können. Wir geben Ihnen direkt die Methode an, wie sich die Fehlerpositionen bestimmen lassen.

Fehlerbestimmung für BCH-Codes mit t=2

1. Das empfangene Wort sei $r(X)$. Bestimmen Sie $a = r(\alpha)$ und $b = r(\alpha^3)$.

2. Suchen Sie durch Einsetzen die Nullstellen der Fehlergleichung

$$X^2 + a \cdot X + \frac{a^3 + b}{a} = 0.$$

3. Die (max. 2) Lösungen der obigen Gleichung geben Ihnen die Position des Fehlers an.

Versuchen Sie es gleich selbst:

Aufg. 5.3 Wir verwenden den (15,7)-BCH Code. Sie empfangen das Wort $r(X) = X^{11} + X^{10} + X^8 + X^7 + X^6 + X^3$. Suchen Sie die 2 aufgetretenen Fehler. Aus [Gravano01, S. 194f].

Lösungen der Aufgaben

Aufgabe 5.1

1. Wir wollen 2 Fehler korrigieren, d.h. wir wählen $t = 2$.

2. Der Galois-Körper $GF(2^4)$ ist auf Seite 79 angegeben. Wir wählen α als primitives Element.

3. Aus Gleichung 5.1 folgt: $g(X) = LCM(m_1(X), m_3(X))$.

4. Ebenfalls auf Seite 79 sind die minimalen Polynome des Galois-Körpers $GF(2^4)$ angegeben. Daraus folgen $m_1(X) = X^3 + X + 1$ und $m_3(X) = X^4 + X^3 + X^2 + X + 1$.

5. Das Generatorpolynom ergibt sich zu $g(X) = X^8 + X^7 + X^6 + X^4 + 1$.

Die Blocklänge ist dabei $n = 2^4 - 1 = 15$. Die Informationslänge ergibt sich zu 15 - (Grad des Generatorpolynoms) $= 15 - 8 = 7$. Es handelt sich hier also um den (15,7)-BCH Code.

Aufgabe 5.2

Es gibt zwei Möglichkeiten: Wenn mehr als t Fehler im empfangenen Wort auftreten. Oder wenn sich durch den Fehler ein anderes gültiges Codewort ergibt.

Aufgabe 5.3

Wir gehen nach der auf Seite 71 beschriebenen Methode vor. Der verwendete Galois-Körper ist natürlich $GF(2^4)$.

1. Als Erstes berechnen wir die beiden Syndrome

$$r(\alpha) = ... = \alpha^4$$

$$r(\alpha^3) = ... = \alpha^{13}$$

2. $\frac{a^3+b}{a} = ... = \alpha^{12}$. Beachten Sie hier, dass z.B. $\frac{\alpha}{\alpha^3} = \alpha^{-2} = \alpha^{13}$.

Also heisst die Gleichung

$$X^2 + \alpha^4 X + \alpha^{12} = 0.$$

3. Mit Einsetzen erhalten wir die Lösungen α^2 und α^{10}.

4. Das gesendete Codewort war also $c(X) = r(X) - (X^2 + X^{10}) = X^{11} + X^8 + X^7 + X^6 + X^3 + X^2$.

Lernkontrolle

L.1

Konstruieren Sie einen 1-Fehler-korrigierenden Code über $GF(2^4)$ und geben Sie das Generatorpolynom an.

L.2

Geben Sie an, ob folgende zwei Wörter mit dem (7,4)-BCH Code korrekt empfangen wurden. Falls nicht, geben Sie das korrekte Codewort an.
Zur Erinnerung: Das Generatorpolynom ist $g(X) = X^3 + X + 1$.

a) $X^6 + X^4 + X^3$

a) $X^5 + X^3 + X^2 + 1$

L.3

In dieser Aufgabe verwenden wir den (15,7)-BCH Code. Sie empfangen das Wort $d = X^8 + 1$. Wurde dieses Wort richtig empfangen? Wenn nicht, wie heisst das korrekte Codewort?

Lösungen der Lernkontrolle

L.1

Das Generatorpolynom ist $g(X) = X^4 + X + 1$.

L.2

a) Das Wort wurde korrekt empfangen.

a) Das Wort wurde fehlerhaft empfangen. Das richtige Codewort ist $X^5 + X^3 + X^2$.

L.3

Wir gehen nach der auf Seite 71 beschriebenen Methode vor. Der verwendete Galois-Körper ist natürlich $GF(2^4)$.
Als Erstes berechnen wir die beiden Syndrome

$$r(\alpha) = ... = \alpha^2$$

$$r(\alpha^3) = ... = \alpha^3 + \alpha + 1 = \alpha^7.$$

Die Fehlergleichung ergibt sich zu

$$X^2 + \alpha^2 X + \alpha^8 = 0.$$

Aus den Nullstellen dieser Gleichung folgt das Fehlerwort $e(X) = X^8 + 1$ Das heisst, dass gar nichts gesendet wurde, da $c(X) = r(X) - e(X) = 0$.

Literaturverzeichnis

[Meyer02] Meyer, M.: Kommunikationstechnik. Braunschweig (Vieweg). 2002, 2. Auflage.

[Haykin01] Haykin, S.: Communication Systems. Hoboken, NJ (John Wiley & Sons, Inc.). 2001, 4th Edition.

[Gravano01] Gravano, S.: Error Control Codes. Oxford (Oxford University Press). 2001.

[Sweeney91] Sweeney, P.: Error control coding: An introduction. Hemel Hempstead (Prentice Hall). 1991

[Junnickel95] Junnickel, D.: Codierungstheorie. Heidelberg, Berlin, Oxford (Spektrum akademischer Verlag). 1995

[Peterson03] Peterson, L. Davie B.: Computernetze. Heidelberg (dpunkt.verlag Gmbh). 2004

[Tannenbaum03] Tannenbaum, A.: Communication Networks. Upper Saddle River, New Jersey 07458 (Pearson Education, Inc.). 2003, 4th Edition.

[komsys] Wittneben A.: Vorlesungsunterlagen Kommununikationssysteme. ETH Zürich. Wintersemester 05/06

[WikiM06] *http://de.wikipedia.org/wiki/Matrizenrechnung*, Stand 01.07.2006

[WikiGr06] *http://de.wikipedia.org/wiki/Hierarchie_mathematischer_Strukturen*, Stand 04.07.2006

[WikiH06] *http://en.wikipedia.org/wiki/Hamming_distance*, Stand 15.07.2006

[WikiK06] *http://de.wikipedia.org/wiki/Kanalkodierung*, Stand 15.07.2006

[WikiCRCE06] *http://en.wikipedia.org/wiki/Cyclic_redundancy_check*, Stand 03.08.2006

[Bruenner06] *http://www.arndt-bruenner.de/mathe/scripts/polynomdivision.htm*, Stand 01.06.2006

[WikiCRCD06] *http://de.wikipedia.org/wiki/Zyklische_Redundanzpr%C3%BCfung* Stand 03.08.2006

[Lin04] Lin, S. Costello D.: Error Control Coding. Upper Saddle River (Pearson Prentice Hall). 2004, 2nd Edition

[ross] *http://www.ross.net/crc/download/crc_v3.txt* Stand 02.08.2006

Anhang A

Galois-Körper

$GF(2^3)$

Elemente

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

Minimale Polynome

Element(e)	**Minimales Polynom**
0	x
1	$x+1$
α, α^2, α^4	x^3+x+1
α^3, α^5, α^6	x^3+x^2+1

$GF(2^4)$

Elemente

$$
\begin{aligned}
&0 \\
&1 \\
&\alpha \\
&\alpha^2 \\
&\alpha^3 \\
\alpha^4 &= \alpha+1 \\
\alpha^5 &= \alpha^2+\alpha \\
\alpha^6 &= \alpha^3+\alpha^2 \\
\alpha^7 &= \alpha^3+\alpha+1 \\
\alpha^8 &= \alpha^2+1 \\
\alpha^9 &= \alpha^3+\alpha \\
\alpha^{10} &= \alpha^2+\alpha+1 \\
\alpha^{11} &= \alpha^3+\alpha^2+\alpha \\
\alpha^{12} &= \alpha^3+\alpha^2+\alpha+1 \\
\alpha^{13} &= \alpha^3+\alpha^2+1 \\
\alpha^{14} &= \alpha^3+1
\end{aligned}
$$

Minimale Polynome

Element(e)	**Minimales Polynom**
0	x
1	$x+1$
α, α^2, α^4, α^8	x^4+x+1
α^5, α^{10}	x^2+x+1
α^3, α^6, α^9, α^{12}	$x^4+x^3+x^2+x+1$
α^7, α^{11}, α^{13}, α^{14}	x^4+x^3+1

$GF(2^5)$

Elemente

0			α^{15}	$=$	$\alpha^4+\alpha^3+\alpha^2+\alpha+1$
1			α^{16}	$=$	$\alpha^4+\alpha^3+\alpha+1$
α			α^{17}	$=$	$\alpha^4+\alpha+1$
α^2			α^{18}	$=$	$\alpha+1$
α^3			α^{19}	$=$	$\alpha^2+\alpha$
α^4			α^{20}	$=$	$\alpha^3+\alpha^2$
α^5	$=$	$\alpha^2+\alpha$	α^{21}	$=$	$\alpha^4+\alpha^3$
α^6	$=$	$\alpha^3+\alpha$	α^{22}	$=$	$\alpha^4+\alpha^2+1$
α^7	$=$	$\alpha^4+\alpha^2$	α^{23}	$=$	$\alpha^3+\alpha^2+\alpha+1$
α^8	$=$	$\alpha^3+\alpha^2+1$	α^{24}	$=$	$\alpha^4+\alpha^3+\alpha^2+\alpha$
α^9	$=$	$\alpha^4+\alpha^3+\alpha$	α^{25}	$=$	$\alpha^4+\alpha^3+1$
α^{10}	$=$	α^4+1	α^{26}	$=$	$\alpha^4+\alpha^2+\alpha+1$
α^{11}	$=$	$\alpha^2+\alpha+1$	α^{27}	$=$	$\alpha^3+\alpha+1$
α^{12}	$=$	$\alpha^3+\alpha^2+\alpha$	α^{28}	$=$	$\alpha^4+\alpha^2+\alpha$
α^{13}	$=$	$\alpha^4+\alpha^3+\alpha^2$	α^{29}	$=$	α^3+1
α^{14}	$=$	$\alpha^4+\alpha^3+\alpha^2+1$	α^{30}	$=$	$\alpha^4+\alpha$

Anhang B

Tutortests

Tutortest Kapitel 1

Aufgaben

Diesen Tutortest sollten Sie ohne Mühe lösen können. Es geht darum, dass Sie die vorgestellten Verfahren für die kommenden Themen beherrschen.

Aufgabe 1

Gegeben seien a und B. Berechnen Sie $B^T a$.

$$a = \begin{pmatrix} 4 \\ 1 \\ 0 \\ 2 \end{pmatrix} \qquad B = \begin{pmatrix} 2 & 0 \\ 1 & -2 \\ 1 & -3 \\ 0 & -1 \end{pmatrix}$$

Erklären Sie, warum $B \cdot a$ keine gültige Rechnung ist.

Aufgabe 2

Berechnen Sie $11101101/1101$ (Modulo-2-Arithmetik).

Aufgabe 3

Dividieren Sie $4x^6 - x^4 + 4x^3 - 14x^2 + 8x - 14$ durch $-5 - 2x^2 + x^3 + 2x^4$. Achten Sie auf die Sortierung der Polynome!

Aufgabe 4

Wir haben für die Modulo-2-Arithmetik zwei Wahrheitstafeln eingeführt. Erstellen Sie für die Modulo-3-Arithmetik die Wahrheitstafel für die Addition. Als Hilfe hier nochmals jene der Modulo-2-Arithmetik:

$$\begin{array}{ccccc} 0 & + & 0 & = & 0 \\ 0 & + & 1 & = & 1 \\ 1 & + & 0 & = & 1 \\ 1 & + & 1 & = & 0 \end{array}$$

Lösungen Tutortest Kapitel 1

Aufgabe 1 (K2)

Das Resultat ergibt sich zu $\begin{pmatrix} 9 \\ -4 \end{pmatrix}$.

$B \cdot a$ ist keine gültige Rechnung, weil die Dimensionen des Vektors und der Matrix nicht aufeinander abgestimmt sind! Bei der Rechnung $B \cdot a$ müsste B die Dimension $c \times 4$ haben, wobei c eine natürliche Zahl ist. Als Beispiel hat bei der gültigen Rechnung $B^T a$ die Matrix B^T die Dimension 2×4.

Aufgabe 2 (K1)

Die Lösung ist 10101. Der Lösungsweg lautet wie folgt:

```
1 1 1 0 1 1 0 1 / 1 1 0 1  =  1 0 1 0 1
1 1 0 1 ----------------------+ | | | |
-------                         | | | |
  0 1 1 1 1 0 1                 | | | |
  0 0 0 0       ----------------+ | | |
  -------                         | | |
    1 1 1 0 0 1                   | | |
    1 1 0 1     ------------------+ | |
    -------                         | |
      0 1 1 0 1                     | |
      0 0 0 0   --------------------+ |
      -------                         |
        1 1 0 1                       |
        1 1 0 1 ----------------------+
        -------
              0 --> Rest 0
```

Aufgabe 3 (K1)

Die Lösung lässt sich auf zwei Arten angeben:

$2x^2 - x + 2 + \frac{3x-4}{2x^4+x^3-2x^2-5}$

oder anders geschrieben:

$2x^2 - x + 2$ Rest $3x - 4$.

Der Lösungsweg sieht wie folgt aus:

```
(4x^6          -  x^4  + 4x^3  - 14x^2  + 8x  - 14) : (2x^4 + x^3 - 2x^2 - 5)


                                                =  2x^2 - x + 2   Rest  3x - 4


 4x^6  + 2x^5  - 4x^4          - 10x^2
 ------------------------------------------------
       - 2x^5  + 3x^4  + 4x^3  -  4x^2  + 8x  - 14
       - 2x^5  -  x^4  + 2x^3           + 5x
       ----------------------------------------
                 4x^4  + 2x^3  -  4x^2  + 3x  - 14
                 4x^4  + 2x^3  -  4x^2        - 10
                 --------------------------------
                                         3x  -  4
```

Aufgabe 4 (K3)

$$
\begin{array}{ccccc}
0 & + & 0 & = & 0 \\
0 & + & 1 & = & 1 \\
0 & + & 2 & = & 2 \\
1 & + & 0 & = & 1 \\
1 & + & 1 & = & 2 \\
1 & + & 2 & = & 0 \\
2 & + & 0 & = & 2 \\
2 & + & 1 & = & 0 \\
2 & + & 2 & = & 1
\end{array}
$$

Tutortest Kapitel 2

Aufgaben

Aufgabe 1

Warum wird für Broadcast und für Echtzeitanwendungen ein FEC- und nicht ein ARQ-Verfahren verwendet?

Aufgabe 2

Bestimmen Sie von den zwei Datenworten $i_1 = (1,0,1)$ und $i_2 = (0,0,1)$ die durch einen (2,1)-Repetitionscode generierten Codeworte.

Aufgabe 3

Bestimmen Sie die 2^k möglichen Codewörter durch den „single-paritiy check“ entstandenen Code für $k = 3$.

Aufgabe 4

Wieviele Fehler kann der lineare Blockcode C(7,4,3) korrigieren, respektive erkennen?

Lösungen

Aufgabe 1 (K3)

ARQ-Verfahren sind für Punkt-zu-Punkt-Verbindungen gedacht. Bei einem Broadcast hingegen sendet 1 Sender an mehrere (viele) Empfänger. Da diese nun aber jedes Paket einzeln bestätigen müssen, wäre der Sender gar nicht in der Lage, alle Bestätigungen zu verarbeiten.
Für Echtzeit-Anwendungen wie z.B.Videoübertragung ist ein ARQ-Verfahren kaum verwendbar. Entscheidend ist bei hier vor allem, dass die Veränderung der Verzögerung klein bleibt. Träte nun bei einem ARQ-Verfahren ein Fehler auf, so müsste man auf das erneute Eintreffen des Paketes warten. Bei einem FEC-Verfahren treffen hingegen die Pakete immer mit der gleichen Verzögerung ein und allfällige Fehler können direkt im Empfänger behoben werden. Diese lokale Behebung der Fehler zieht kaum Verzögerungen mit sich.

Aufgabe 2 (K1)

$c_1 = (1,1,0,0,1,1)$ $c_2 = (0,0,0,0,1,1)$

Aufgabe 3 (K2)

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

Somit erhält man:

000	0000
001	1001
010	1010
011	0011
100	1100
101	0101
110	0110
111	1111

Aufgabe 4 (K2)

Ein C(7,4,3) Code besitzt die Mindestdistanz 3. Somit können mit dem Code $(d-1) = 2$ Fehler erkannt werden und $\left\lfloor \frac{d-1}{2} \right\rfloor = 1$ Fehler korrigiert werden.

Tutortest Kapitel 3

Aufgaben

Aufgabe 1

Was ist die Definition eines zyklischen Codes?

Aufgabe 2

Was ist ein Burstfehler und wann kann ein Code, generiert mit einem Polynom vom Grad 4, solche Fehler korrigieren?

Aufgabe 3

Berechnen Sie zu folgenden Nachrichtenworten die Codeworte des C(7,4) Codes mit $g(X) = 1+X+X^3$

a) $i(X) = 1+X+X^3$

b) $i(X) = 1+X+X^3$ in der systematischen Form

d) $i(X) = X^2$ in der systematischen Form

Aufgabe 4

Bestimmen Sie mit der im Kapitel 3.4 gegeben Schaltung das Codewort $c(X)$ wenn $u(X) = 1+X^2+X^4$ und $g(X) = 1+X+X^3$ gegeben sind.

Lösungen Tutortest Kapitel 3

Aufgabe 1 (K1)

Ein linearer Code C(n,k) wird zyklisch genannt, wenn jede zyklische Verschiebung eines Codewortes wieder ein Codewort in C ergibt.

Aufgabe 2 (K2)

Ein „Burst“-Fehler der Länge k kann folgendermassen definiert werden: $E(X) = X^i(X^{k-1} + \cdots + 1)$. Im Kapitel haben Sie gesehen, dass ein Code mit Generatorpolynom vierten Grades alle Burstfehler der Länge 4 und kürzer erkennen kann. Überschreitet die Länge des Burst den Grad des Polynoms, wird trotzdem noch ein relativ grosser Anteil der Fehlerpolynome erkannt.

Aufgabe 3 (K1)

a) $c(X) = i(X)g(X) = 1 + X^2 + X^6$

b) $i(X) = 1 + X + X^3$ in der systematischen Form

```
1 0 1 1 0 0 0  /  1 0 1 1  =  1 0 0 0
1 0 1 1 ----------------------+ | | |
-------                         | | |
  0 0 0 0                       | | |
  0 0 0 0    --------------------+ | |
  -------                         | |
    0 0 0 0                       | |
    0 0 0 0 ----------------------+ |
    -------                         |
      0 0 0 0                       |
      0 0 0 0 ----------------------+
      -------
        0 0 0      Rest = 0
```

Der Rest ergibt somit 0. Somit erhält man $c(X) = b(X) + i(X)X^3 = X^3 + X^4 + X^6$.

c) $i(X) = X^2$ in der systematischen Form

```
0 1 0 0 0 0 0  /  1 0 1 1  =  0 1 0 1
0 0 0 0 ----------------------+ | | |
-------                         | | |
  1 0 0 0                       | | |
  1 0 1 1    -------------------+ | |
  -------                         | |
    0 1 1 0                       | |
    0 0 0 0 ----------------------+ |
    -------                         |
      1 1 0 0                       |
      1 0 1 1 ----------------------+
      -------
        1 1 1      Rest = 111
```

Der Rest ergibt somit X^2+X+1. Somit sollten Sie $c(X) = b(X)+i(X)X^3 = 1+X+X^2+x^5$ erhalten.

Aufgabe 4 (K2)

Die Registerinhalte berechnen sich folgendermassen:

Input	Registerinhalte
	000(Startzustand)
1	110 (erster Shift)
1	101 (zweiter Shift)
0	100 (dritter Shift)
1	100 (vierter Shift)

Der vierte Shift entspricht dem Rest der Polynomdivision. Somit ergibt sich für das Codewort: $c(X) = 1+X^3+X^5+X^6$.

Tutortest Kapitel 4

Aufgaben

Aufgabe 1

Konstruieren Sie einen endlichen Körper aus der Gleichung $x^2+x+1=0$. Tipp: Gehen Sie von der Modulo-2-Arithmetik aus.

Aufgabe 2

Lösen Sie in $GF(2^4)$ die Gleichung

$$x^3+\alpha x^2+\alpha^2 x+\alpha^{10}=0.$$

Tipp: Lassen Sie beim Berechnen α^4 und α^5 aus.

Aufgabe 3 (aus [Gravano01, S.183])

Lösen Sie in $GF(2^4)$ das Gleichungssystem

$$\begin{aligned} x+\alpha^4 y &= \alpha^5 \\ \alpha^5 x+\alpha^2 y &= \alpha^3. \end{aligned}$$

Erklären Sie ausserdem, warum die Lösung in $GF(2^3)$ nicht eindeutig wäre. Tipp: Setzen Sie die zweite Gleichung in die erste ein.

Aufgabe 4

Berechnen Sie die Ordnung der Elemente $1, \alpha, \alpha^2, \alpha^3, \alpha^4$ aus $GF(2^5)$. Was fällt Ihnen dabei auf?

Lösungen Tutortest Kapitel 4

Aufgabe 1 (K3)

Weil wir von der Modulo-2-Arithmetik ausgehen, wissen wir, dass 0 und 1 sicher Elemente dieses neuen Körpers sind. Wir führen ein neues Element α ein, das eine Lösung der Gleichung $x^2+x+1=0$ sei, also $\alpha^2+\alpha+1=0$. Ein endlicher Körper lässt sich ja konstruieren, indem man dieses Element α fortlaufend potenziert. Als Erstes folgt α^2. Unsere Ausgangsgleichung lässt sich umschreiben zu $\alpha^2=\alpha+1$. Dies ist die Polynomdarstellung von α^2. Wir haben also bis jetzt zwei neue Elemente gefunden, α und α^2. Als nächstes Element folgt

$$\alpha^3=\alpha\cdot\alpha^2=\alpha\cdot(\alpha+1)=\alpha^2+\alpha=\alpha+1+\alpha=1$$

Wir erhalten kein neues Element mehr. Alle weiteren Potenzen von α lassen sich auf die bisherigen vier Elemente reduzieren.
Unser Körper lautet also: $GF(4)=GF(2^2)=\{0,1,\alpha,\alpha^2=\alpha+1\}$.

Aufgabe 2 (K1)

1, α^3 und α^7. Die Lösung ergibt sich durch Einsetzen. Für z.B. $x=\alpha^3$ berechnet sich die Gleichung wie folgt:

$$\begin{aligned}&x^3+\alpha x^2+\alpha^2x+\alpha^{10}=\alpha^9+\alpha^7+\alpha^5+\alpha^{10}=\\&=(\alpha^3+\alpha)+(\alpha^3+\alpha^2+1)+(\alpha^2+\alpha)+(\alpha^2+\alpha+1)=0\end{aligned}$$

Aufgabe 3 (K3)

In $GF(2^4)$ ergibt sich $x=0$ und $y=\alpha$.
In $GF(2^3)$ folgt $0=0$, wenn man die erste Gleichung nach x auflöst und in die zweite Gleichung einsetzt. Die beiden Gleichungen sind also gleichbedeutend. Es existieren mehrere Lösungen.

Aufgabe 4 (K2)

Alle diese Elemente haben die gleiche Ordnung 31, sind also primitiv.

Tutortest Kapitel 5

Die folgenden Angaben könnten Ihnen nützlich sein in diesem Test. Weitere Tabellen finden Sie im Anhang A.

- Auszug aus den minimalen Polynomen des Galois-Körpers $GF(2^5)$

Element	**Minimales Polynom**
α	x^5+x^2+1
α^3	$x^5+x^4+x^3+x^2+1$

- Generatorpolynom des (15,7)-BCH Codes: $g(x) = x^8+x^7+x^6+x^4+1$

Aufgaben

Aufgabe 1

Sie müssen auf einem etwas schlechteren Kanal Daten übertragen. Dazu reichen Codes, die nur 1 oder 2 Fehler korrigieren können, nicht aus. Konstruieren Sie deshalb einen Code, der bis zu 3 Fehler korrigieren kann. Wählen Sie den Galois-Körper $GF(2^4)$. Geben Sie das Generatorpolynom an.

Aufgabe 2

Die BCH-Codes sind Teil der zyklischen Codes. Nennen Sie eine weitere Eigenschaft der BCH-Codes, die allgemeine zyklische Codes nicht erfüllen. Zählen Sie ausserdem zwei Gemeinsamkeiten auf.

Aufgabe 3

Wir möchten nun gerne grösse Blöcke auf einmal übertragen. Deshalb wollen wir einen BCH-Code konstruieren, der eine Informationslänge von 26 Bit aufweist. Der Code soll 1 Fehler korrigieren können und eine minimale Blocklänge aufweisen. Geben Sie das Generatorpolynom an.

Aufgabe 4

Sie empfangen mit dem (15,7)-BCH Code das Wort

$$d = x^7 + x^6 + x^5 + x^4 + 1$$

Zeigen Sie, dass d zwei Fehler enthält. Geben Sie das Fehlerpolynom und das korrekte Codewort zu d an.

Lösungen Tutortest Kapitel 5

Aufgabe 1 (K1)

Da $t = 3$ ist, müssen wir die minimalen Polynome der Elemente α, α^3 und α^5 miteinander mulitplizieren und das LCM davon nehmen. In diesem Fall ist das LCM gerade gleich das Produkt der drei Polynome, da keines ein Teiler eines anderen ist.
Das Generatorpolynom ergibt sich zu

$$g(x) = LCM(m_1(x), m_2(x), m_5(x)) = m_1(x) \cdot m_2(x) \cdot m_5(x) = ... = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1.$$

Aufgabe 2 (K4)

Gemeinsamkeiten:

- Beide werden durch ein Generatorpolynom beschrieben.
- Die Fehler lassen sich durch Syndrome bestimmen.

Im Gegensatz zu den allgemeinen zyklischen Codes in Kapitel 3, wo wir das Fehlersyndrom durch Division des empfangenen Wortes durch das Generatorpolynom erhalten haben ($r(x)/g(x)$), können wir bei den BCH-Codes einfach die Nullstellen des Generatorpolynoms α^i in das empfangene Wort $r(x)$ einsetzen: $r(\alpha^i)$.
Ein Pluspunkt der BCH-Codes ist, dass man direkt mit vorgegebenen Fehlereigenschaften einen passenden Code sehr einfach konstruieren kann, während es bei allgemeinen zyklischen Codes einiges schwieriger ist.

Aufgabe 3 (K3)

Die Blocklänge des Codes muss grösser als die Informationslänge sein, also ist $2^m - 1 > 26$. Als erster Galois-Körper ermöglicht uns also $GF(2^5)$ eventuell eine Konstruktion. Wir wählen $t = 1$. Daraus ergibt sich das Generatorpolynom $g(x) = x^5 + x^3 + x^2 + 1$. Die Kontrolle ergibt, dass $31 - 5 = 26$. Dieser (31,26)-BCH Code hat also tatsächlich eine Informationslänge von 26 Bit.

Aufgabe 4 (K1)

Wir gehen nach der auf Seite 71 beschriebenen Methode vor.

Die beiden Fehler-Syndrome ergeben sich zu

$$d(\alpha) = \alpha^4$$

$$d(\alpha^3) = \alpha^3 + \alpha + 1$$

Daraus folgt die Gleichung $x^2 + a \cdot x + \frac{a^3+b}{a} = 0$ zu

$$x^2 + \alpha^4 \cdot x + \alpha^{13} = 0$$

mit den beiden Wurzeln α^5 und α^8. Das Fehlerpolynom lautet damit $e(x) = x^5 + x^8$. Das korrekte Codewort ist also

$$c(x) = x^8 + x^7 + x^6 + x^4 + 1.$$