TECHNISCHE UNIVERSITÄT WIEN

DIPLOMARBEIT

Mathematische Codierungsverfahren bei CDs, DVDs und der Warenkennzeichnung

von

David M. Fischer

Ausgeführt am

Institut für Diskrete Mathematik und Geometrie

unter der Anleitung von

Em.Univ.Prof. Dr.phil. Dietmar Dorninger

19. Januar 2018

Inhaltsverzeichnis

Einleitung

Zielgruppe dieser Diplomarbeit ist die Lehrerschaft von höheren Schulen (AHS, HTL, usw.). Die Arbeit soll auf didaktisch sinnvolle Weise dem Leser einen Überblick über die Codierung der CD sowie die Neuerungen bei der Entwicklung der DVD geben und beispielhaft den Einsatz von Codierung bei der Warenkennzeichnung, also im „täglichen Leben“ aufzeigen. Nach dem Lesen soll es einem Lehrer möglich sein, eine vorwissenschaftliche Arbeit zu betreuen oder einen projektorientierten Unterricht zu diesem Thema zu gestalten.

Es wird davon ausgegangen, dass ein grundlegendes Verständnis der Algebra vorhanden ist. Einzelne Begriffe werden wiederholt, auf Beweisführungen wird großteils verzichtet.

Nach der Motivation beginnt die Diplomarbeit mit der Wiederholung der grundlegenden algebraischen Begriffe, wie Zahlenmengen und das Rechnen mit Restklassen. Danach werden algebraische Strukturen, nämlich die abelsche Gruppe, der Ring, der Körper, der Vektorraum, der Polynomring und das Ideal definiert und erklärt. Es werden die Darstellungsmöglichkeiten der Information mithilfe von Vektoren bzw. Polynomen erläutert. Abschließend wird auf die für die Codierungstheorie wichtigste Struktur, das Galoisfeld, eingegangen. Es wird die Konstruktion zuerst allgemein, danach anhand eines Beispiels, erarbeitet.

Im nächsten Kapitel werden codierungstheoretische Begriffe festgelegt. Die wichtige Abgrenzung zur Kryptologie wird anfangs behandelt. Eine schematische Übersicht über die Informationsübertragung wird im darauf folgenden Abschnitt veranschaulicht. Anschließend werden grundlegende Begriffe der Codierungstheorie, wie das Alphabet, das Codewort, der Hammingabstand und weitere Begriffsbildungen, erklärt. Da diese Arbeit zu einem großen Teil von Computercodierungen handelt, wird dies noch in einem eigenen Abschnitt erklärt. Danach werden, dem Schema der Codierung folgend, die zwei grundlegenden Codierungstypen unterschieden. Es handelt sich hierbei um die Quellencodierung und die Fehlercodierung. Anschließend werden einige Eigenschaften von Codes, nämlich lineare Codes, zyklische Codes und Polynomcodes, systematische Codes und optimale Codes erklärt. Zum Schluss des Kapitels wird näher auf die (n, k)-Codierung und den RS-Code eingegangen.

Das dritte Kapitel, die Anwendungen, ist der Codierung der CD, der DVD und der Warenkennzeichnung gewidmet. Die Codierung der CD wird im ersten Abschnitt des Kapitels behandelt. Nach einem technischen Überblick über Aufbau und Funktionsweise der CD wird auf den Ablauf der CD-Codierung eingegangen. Dieser Ablauf umfasst acht Schritte, welche einzeln behandelt werden. Dabei wird der Quellencodierung (der Digitalisierung der

Musik), der Durchmischung der Information (Interleaving) und der Fehlercodierung (EF-Modulation) besondere Aufmerksamkeit geschenkt. Danach werden die Unterschiede einer CD gegenüber der DVD erläutert.

Im letzten Kapitel wird die Warenkennzeichnung behandelt. Beispielhaft werden fünf Kennzeichnungen angeführt. Es wird in einem Schema, das für alle Beispiele gleich bleibt, erklärt, was die Abkürzungen bedeuten und wie die Nummern aufgebaut sind. Anschließend werden die Prüfziffern anhand einer Beispielnummer errechnet. Danach ist eine Realisierung der Prüfziffernberechnung in der Programmiersprache Python angeführt.

Am Schluss dieser Arbeit sind, zum Nachschlagen algebraische Strukturen, ein Glossar der Abkürzungen und deren Erklärung, die in dieser Arbeit verwendet werden, sowie Codierungstafeln angehängt.

Danksagungen

Diese Diplomarbeit markiert das Ende meines Lehramtsstudiums. Ich möchte allen voran Prof. Dr. Dietmar Dorninger für seinen Einsatz bei der Betreuung meiner Diplomarbeit danken. Er hat mir dieses interessante Thema vorgeschlagen und mich mit der nötigen Literatur versorgt.

Nachfolgend möchte ich meiner Familie danken, die mir dieses Studium ermöglicht hat und mich stets in meinen Ausbildungen unterstützt hat, obwohl sie zum großen Teil selbst kein Hochschulstudium absolvieren konnten.

Abschließend möchte ich mich bei meiner Ehefrau Astrid bedanken, die mir nicht nur bei den sprachlichen und grammatischen Korrekturen geholfen hat, sondern mich auch bei der Diplomarbeit und beim Studium allgemein mental unterstützt hat.

Motivation

Wie weit die Mathematik in unserem täglichen Leben Anwendung findet, ist vielen Schülern nicht bewusst. Eine der typischen Fragen eines Pubertierenden ist: „Wozu braucht man das?“ Ich finde diese Frage sehr berechtigt. Die Zeit, die uns allen zur Verfügung steht, ist begrenzt und sollte weise genutzt werden. Dass die Mathematik und die Codierungstheorie im Besonderen diese Zeit verdient, ist uns ganz klar.

Das Morse-Alphabet hat schon hunderten Menschen das Leben gerettet. Das Symbol für „SOS“ (··· ___ ···) ist besonders als Klang wiedererkennbar. Die Braille-Schrift hat es blinden und sehbehinderten Menschen ermöglicht, den Fängen des Analphabetismus zu entkommen. All diese Alphabete erlauben uns Wissen zu erwerben und zu kommunizieren. Kommunikation ist eine der wichtigsten Aktivitäten, wenn nicht die Hauptaktivität unseres Lebens. Nirgendwo kann man mehr über die Welt und über sich selbst erfahren als in der Kommunikation mit anderen.

Aber auch im Bereich der Wirtschaft ist der Einkauf mit der Produktnummer (GTIN) revolutioniert worden. Durch die GTIN und ihre Darstellung als Strichcode können Produkte in kürzester Zeit in die Kasse eingelesen und verrechnet werden. Das spart Zeit und verringert den Aufwand. Um den Einlesevorgang störungssicher zu machen, benötigt man Prüfnummern, um sicherzustellen, dass die eingelesene Nummer auch tatsächlich die Nummer ist, die dem Produkt zugeordnet ist.

Aber auch High-Tech-Anwendungen sind ohne Codierungstheorie nicht denkbar. Die Menge an Daten, die wir auf einer CD, DVD oder eine Festplatte unterbringen können, könnte ganze Bibliotheken füllen. Ein E-Book-Reader lässt uns tausende Bücher auf einem Gerät lesen, das nicht dicker ist als ein Pamphlet. Eine CD lässt uns ein Konzert in einer Qualität erleben, als wären wir selbst anwesend. Durch die Möglichkeit, die CD in mehrere Titel zu unterteilen, wurde unsere Denkweise von Alben verändert.

Auf einer DVD ist ein Film codiert, der die Schärfe der analogen VHS bei Weitem übersteigt. Wenn man sich vorstellt, dass ein Film nicht nur Bildmaterial hat, sondern auch aus mehreren qualitätiv hochwertigen zweispurigen (im Falle von Dolby 7.1 sogar achtspurigen) Audioaufnahmen[1] und zuschaltbaren Untertiteln besteht, dann wird einem bewusst, wie viele Daten tatsächlich auf diese kleine Scheibe passen.

All diese Errungenschaften wären ohne die Codierungstheorie nicht möglich gewesen. Es lohnt sich also, Zeit in die Codierungstheorie zu investieren.

[1] Eine DVD verfügt meist über mehrere Tonspuren, um den Film in verschiedenen Sprachen sehen zu können.

Kapitel 1

Wichtige Begriffe aus der Algebra

Um in die Tiefe der Materie eindringen zu können, ist es wichtig, einige Begriffe aus der Algebra zu wiederholen. Ein Basisverständnis des Lesers über grundlegende Begriffe (z. B.: Funktion, Operation) wird jedoch vorausgesetzt. Zusätzlich findet der Leser im Kapitel 4.1 eine kompakte Sammlung der wichtigsten Definitionen, die für diese Arbeit gebraucht werden.

1.1 Zahlenmengen

Die Codierungstheorie nutzt meist die Menge der *ganzen Zahlen* $\mathbb{Z}$. Vielfach geht es darum, eine Zuordnung gewisser Schriftzeichen (z. B. Buchstaben, Zahlen, Satzzeichen) und Zahlen herzustellen. Oft werden andere Daten, wie etwa bei der CD die Frequenz und Amplitude, codiert. Dazu genügen ebenfalls die ganzen Zahlen. In beiden Fällen hat man ein Kontingent an Informationen (Symbole bzw. Frequenz/Amplitude), denen man eine Zahl zuordnet. Diese Zahlen können dann in beliebigen Systemen codiert werden.

Der Computer hat bekannterweise nur die Werte 0 und 1 zur Verfügung und baut aus diesen zwei Ziffern alle Zahlen auf (siehe 2.5.1). Die ganzen Zahlen reichen dazu immer aus. Daher bedient man sich der *Restklassenmenge* $\mathbb{Z}_m$. So gewährleistet man auch die Abgeschlossenheit bei Berechnungen.

Definition Die *Modulo-Funktion* ist eine Abbildung der ganzen Zahlen $\mathbb{Z}$ auf einer Restklassenmenge $\mathbb{Z}_m = \{0, 1, 2, \ldots, m-1\}$. Dabei gilt:[1]

$$x \mod m = a \iff \exists k \in \mathbb{N} : a + k \cdot m = x,\ x, m \in \mathbb{Z},\ a \in \mathbb{Z}_m$$

Beispiel

$$23 \mod 7 = 2 \iff 2 + 3 \cdot 7 = 23$$

1.2 Algebraische Strukturen

Um Zahlen mit Rechenoperationen verknüpfen zu können, braucht es eine algebraische Struktur. Diese besteht aus einer Menge A und einem oder mehreren Operationen $\circ_i$. In der Codierungstheorie wird meist der Restklassenkörper $\mathbb{Z}_m$, wo m eine Primzahlpotenz ist, verwendet, da eine Zuweisung des Alphabets (26 Zeichen und weitere Sonderzeichen) zu Zahlen keiner rationalen Zahlen bedarf. Die Operationen sind meist binäre, also zweistellige, Verknüpfungen, die in denselben Zahlenraum abbilden. Meist geht es hier um die Addition, Multiplikation oder Potenzierung.

Nun können bestimmte Regeln (Assoziativität, Kommutativität, Distributivität) gelten oder nicht gelten. Darüber hinaus können auch Elemente mit bestimmten Eigenschaften auftreten (z. B. neutrales Element, inverses Element). Ein wichtiger algebraischer Typ ist das *Galoisfeld*. Dazu einige aufbauende Definitionen.

Definition Für eine *abelsche Gruppe*[2] $(G, \circ)$ gilt: [4, S. 13]

- Assoziativgesetz: $x \circ (y \circ z) = (x \circ y) \circ z,\ \forall x, y, z \in G$
- es existiert ein neutrales Element e: $e \circ x = x,\ \forall x \in G$
- zu jedem x existiert ein inverses Element x^{-1}: $x \circ x^{-1} = e,\ \forall x \in G$
- Kommutativgesetz:[3] $x \circ y = y \circ x,\ \forall x, y \in G$

Mit obenstehender Definition lässt sich der Begriff des Rings definieren.

[1] Als Präfixschreibweise $mod(x, m) = a$.
[2] auch *kommutative Gruppe* genannt
[3] Gilt das Kommutativgesetz nicht spricht man von einer Gruppe.

Definition Für einen *Ring* $(R, +, \cdot)$ gilt:[4, S. 12]

- $(R, +)$ ist eine abelsche Gruppe
- $(R, \cdot)$ ist eine Halbgruppe
- Distributivgesetze: $x \cdot (y + z) = x \cdot y + x \cdot z$ und $(y + z) \cdot x = y \cdot x + z \cdot x,\ \forall x, y, z \in R$

Nun lässt sich der *Körper* definieren, der einen wesentlichen Begriff in der Codierungstheorie darstellt.

Definition Für einen *Körper* $(K, +, \cdot)$ gilt: [4, S. 13]

- $(K, +)$ ist eine abelsche Gruppe
- $(K \backslash \{0\}, \cdot)$ ist eine abelsche Gruppe
- Distributivgesetze: $x \cdot (y + z) = x \cdot y + x \cdot z$ und $(y + z) \cdot x = y \cdot x + z \cdot x,\ \forall x, y, z \in K$

Oft werden Codewörter als Tupel (Vektoren) dargestellt. Dazu sind zusätzliche Voraussetzungen zu erfüllen.

Definition Für einen *Vektorraum* $(V, +, \cdot)$ über dem Körper K gilt:[4, S. 14]

- $(V, +)$ ist eine abelsche Gruppe
- $\lambda \cdot (x + y) = \lambda \cdot x + \lambda \cdot y,\ \forall x, y \in V, \lambda \in K$
- $(\lambda + \mu) \cdot x = \lambda \cdot x + \lambda \cdot y,\ \forall x \in V, \lambda, \mu \in K$
- $(\lambda \cdot \mu) \cdot x = \lambda \cdot (\mu \cdot x),\ \forall x \in V, \lambda, \mu \in K$
- $1 \cdot x = x \cdot 1 = x,\ \forall x \in V, 1 \in K$

Bemerkung In unserem Fall werden wir einen Vektorraum V betrachten, dessen Elemente *Codewörtern* entsprechen. Diese haben eine bestimmten Länge n, die wir als *Blocklänge* bezeichnen.

Um die Berechnungen, insbesondere die Berechnungen durch den Computer, zu vereinfachen, werden die Vektoren als Polynome dargestellt (siehe 1.2.1).

Definition Sei $R[x]$ die Menge aller Polynome $p(x) := \sum_{i=0}^{n} a_i x^n, n \in \mathbb{N}_0$, über einem Ring R, für $p(x)$ schreiben wir auch kurz $(a_0, a_1, \dots, a_n)$. $R[x]$ wird zu einem Ring, wenn wir für $(a_0, a_1, \dots), (b_0, b_1, \dots) \in R[x]$ definieren:

$$(a_0, a_1, \dots) + (b_0, b_1, \dots) = (a_0 + b_0, a_1 + b_1, \dots) \tag{1.1}$$

$$(a_0, a_1, \dots) \cdot (b_0, b_1, \dots) = (c_0, c_1, \dots) \tag{1.2}$$

mit $c_i = \sum_{i=1}^{j} a_j b_{i-j}$. Üblicherweise wird in der Codierungstheorie der Polynomring über $\mathbb{Z}_m$ verwendet.

Definition Ein *Ideal* I ist ein Unterring des Rings R für das gilt:

$$\forall r \in R : rI := \{ri | i \in I\} \subseteq I^4$$

Es kommt nicht nur bei der Verknüpfung von Elementen aus dem eigenen Raum kein raumfremdes Element heraus, sondern es ist sogar möglich, Elemente mit der Obermenge zu verknüpfen, ohne den Raum zu verlassen.

1.2.1 Darstellung der Information mithilfe von Vektoren und Polynomen

Wenn Information codiert wird, muss diese zum Fehlercodieren in Blöcke geteilt werden. Dies entspricht etwa unseren Wörtern. Im Gegensatz zu diesen wird allerdings die *Blocklänge* für jedes Codewort immer gleich gewählt. Diese Blöcke können nun als n-Tupeln (Vektoren) oder als Koeffizienten eines Polynoms n-ten Grades verstanden werden, wobei n der Blocklänge entspricht.

Beispiel Blocklänge $n = 4$, Information: $1, 4, 8, 5$

- Vektordarstellung: $(1, 4, 8, 5) \in \mathbb{Z}^4$
- Polynomdarstellung: $f(x) = x^4 + 4x^3 + 8x + 5;\ x, f(x) \in \mathbb{Z}$

1.3 Galoisfeld GF(p^m)

Um eine Codierung vornehmen zu können, ist es nötig, die erhobenen Daten (Audiosignal, Produktnummer, ...) in eine algebraische Struktur einzubetten. Eine sehr effiziente algebraische Struktur ist das Galoisfeld.

Ein Galoisfeld ist ein Restklassenkörper, insbesondere also endlich. Der Modul der Restklasse ist eine Primzahl ($p \in \mathbb{P}$) oder eine Potenz einer

[4] Ir gilt analog

Primzahl ($p^m, p \in \mathbb{P}, m \in \mathbb{N}$). Die Elemente des Galoisfeldes werden mithilfe eines irreduziblen Polynoms[5] gebildet. Dabei wird ein Element a zur Menge hinzugefügt und als Nullstelle dieses irreduziblen Polynoms definiert.

Bemerkung Ein Restklassenring $\mathbb{Z}_m$ ist genau dann ein Körper, wenn der Modul m eine Primzahl ist. Allgemein gilt: „Die Anzahl der Elemente eines jeden endlichen Körpers ist eine Primzahl*potenz*“. [6, S. 11]

1.3.1 Allgemeine Konstruktion von $GF(p^m)$

Ein Galoisfeld $GF(p^m)$ wird konstruiert, indem ein irreduzibles Polynom q in $\mathbb{Z}_p[x]$[6] vom Grad m benutzt wird, um

$$GF(p^m) = \mathbb{Z}_p[x]/(q) \tag{1.3}$$

zu bilden. [7, S. 281] Hierbei ist (q) das Ideal des Ringes $\mathbb{Z}_p[x]$, das vom Polynom q generiert wurde.

Eine zweite Möglichkeit der Konstruktion von $GF(p^m)$ ist, durch Adjunktion von a zu $\mathbb{Z}_p$, also

$$GF(p^m) = \mathbb{Z}_p(a) \tag{1.4}$$

Die Elemente des Galoisfeldes sind dann:

$$GF(p^m) = \{0, 1, a, a^2, \ldots, a^{p^m-2}\}. \tag{1.5}$$

Bemerkung Man beachte, dass $GF(2^2) \neq \mathbb{Z}_4$. Bei $\mathbb{Z}_4$ handelt es sich lediglich um einen Restklassenring, da nicht alle Elemente über ein inverses Element verfügen.

1.3.2 Konstruktion von GF(8)

Hier wird die zweite Konstruktionsart von $GF(8)$ über Adjunktion gewählt. Als irreduzible Polynome dritten Grades (da $8 = 2^3$) stehen $x^3 + x^2 + 1$ und $x^3 + x + 1$ zur Auswahl. Hier soll das Polynom $x^3 + x^2 + 1$ gewählt werden. Es wird nun ein Element a zu $\mathbb{Z}_p$ hinzugefügt, das als Nullstelle von $x^3 + x^2 + 1$ gewählt wird:

$$a^3 + a^2 + 1 = 0 \tag{1.6}$$

$$a^3 = a^2 + 1 \tag{1.7}$$

[5] irreduzibles Polynom: Polynom q, das nur die trivialen Teiler $(q, 1)$ hat.

[6] $\mathbb{Z}_p[x]$ ist die Menge aller Polynome über dem Restklassenring $\mathbb{Z}_p$.

Nun stehen bereits fünf Elemente zur Verfügung.

$$0 = 0 \tag{1.8}$$
$$a^0 = 1 \tag{1.9}$$
$$a^1 = a \tag{1.10}$$
$$a^2 = a^2 \tag{1.11}$$
$$a^3 = a^2 + 1 \tag{1.12}$$

Die übrigen drei Elemente werden wie folgt berechnet:

$$\begin{aligned} a^4 &= a \cdot a^3 = a(a^2+1) = a^3 + a = a^2 + 1 + a \\ &= a^2 + a + 1 && (1.13) \\ a^5 &= a \cdot a^4 = a(a^2 + a + 1) = \ldots \\ &= a + 1 && (1.14) \\ a^6 &= a \cdot a^5 = \ldots \\ &= a^2 + a && (1.15) \end{aligned}$$

Bei der Berechnung von a^7 fällt auf, dass $a^7 = a^0 = 1$ ist. Das letzte Element a^6 ist jenes, welches wir auch aufgrund der Gleichung (1.5), nämlich $a^{p^m-2} = a^{8-2}$, erwartet haben. Der Kreis hat sich also geschlossen und es liegen alle Elemente des Galoisfeld 8 vor.

Nun bleiben noch die Additions- und die Multiplikationtafeln aufzustellen. Dabei empfiehlt es sich, zwei verschiedene Darstellungen der Elemente zu wählen. Die Additionstafel ist leichter zu berechnen, wenn man die Polynomform[7], also die rechte Seite der Gleichungen (1.8) bis (1.15) verwendet. Für die Multiplikationstafel empfiehlt es sich, die Potenzdarstellung zu wählen, also die linke Gleichungsseite von (1.8) bis (1.15). Die geeignete Umrechnungstabelle liegt in Tabelle 1.1 auf Seite 11 vor.

Tabellen 1.2 und 1.3 zeigen die Additions- und Multiplikationstafeln von $GF(8)$ in der jeweils geeigneteren Darstellungsart.[8]

Bemerkung Es ist darauf zu achten, dass der Darstellungswechsel nicht zu einer zeilen- oder spaltenweise Umsortierung der Elemente führt.

Tipp Eine weitere tabellarische Vereinfachung bei der Addition ist die Binärschreibweise ($0 = 000, 1 = 001, a = 010, a + 1 = 011, a^2 = 100, \ldots$). Diese ist in Tabelle 1.1 zu sehen.

[7] oder noch kürzer, die Binärform.

[8] Dem interessierten Leser wird die Konstruktion sehr empfohlen.

Bemerkung Wird als irreduzibles Polynom $x^3 + x^2 + x$ gewählt, so erhält man einen zu $\mathbb{Z}_2[x]/(x^2 + x + 1)$ isomorphen Körper. [7, S. 282]

Polynom	Binär	Potenz
0	000	0
1	001	a^0
a	010	a
$a+1$	011	a^5
a^2	100	a^2
a^2+1	101	a^3
a^2+a	110	a^6
a^2+a+1	111	a^4

Tabelle 1.1: Umrechnungstafel der Darstellungen von $GF(8)$ mit Generatorpolynom $x^3 + x^2 + 1$

+	000	001	010	011	100	101	110	111
000	000	001	010	011	100	101	110	111
001	001	000	011	010	101	100	111	110
010	010	011	000	001	110	111	100	101
011	011	010	001	000	111	110	101	100
100	100	101	110	111	000	001	010	011
101	101	100	111	110	001	000	011	010
110	110	111	100	101	010	011	000	001
111	111	110	101	100	011	010	001	000

Tabelle 1.2: Additionstafel

$\cdot$	0	a^0	a^1	a^5	a^2	a^3	a^6	a^4
0	0	0	0	0	0	0	0	0
a^0	0	a^0	a^1	a^5	a^2	a^3	a^6	a^4
a^1	0	a^1	a^2	a^6	a^3	a^4	a^0	a^5
a^5	0	a^5	a^6	a^3	a^0	a^1	a^4	a^2
a^2	0	a^2	a^3	a^0	a^4	a^5	a^1	a^6
a^3	0	a^3	a^4	a^1	a^5	a^6	a^2	a^0
a^6	0	a^6	a^0	a^4	a^1	a^2	a^5	a^3
a^4	0	a^4	a^5	a^2	a^6	a^0	a^3	a^1

Tabelle 1.3: Multiplikationstafel

Kapitel 2

Allgemeine Einführung in die Codierungstheorie

2.1 Abgrenzung zur Kryptologie

Wenn mit Schülern über Codierung gesprochen wird, ist es gerade am Anfang wichtig zu betonen, dass Codierung und Verschlüsselung nicht das Gleiche sind. Die Kryptologie (griech. *versteckt, verborgen, geheim*) beschäftigt sich mit der Verschlüsselung zum Zweck der Geheimhaltung, die Codierungstheorie mit dem Übersetzen in ein für praktische Zwecke besser geeignetes Alphabet.

Codierung Codierungen sind Abbildungen, die den Zweck erfüllen, Information in Wörter über einem anderen Alphabet darzustellen. Das Codealphabet kann hierbei Eigenschaften aufweisen, die das Quellenalphabet nicht hat. Unser bekanntes Alphabet $\{A, \ldots, Z, a, \ldots, z,$ Sonderzeichen$\}$ lässt sich beispielsweise in Binärstrings abbilden, die von Computern verstanden werden. Wird der Buchstabe „G“ auf der Tastatur eingegeben, dann speichert der Computer den Buchstaben als „Unicode“ U+0047[1][2]. Das vom Computer verwendete Alphabet ist das Binärsystem $\mathbb{Z}_2 = \{0, 1\}$. Die bei den Codierungen verwendete Abbildung kann leicht umgekehrt werden, wenn man über die Zuordnungsvorschrift verfügt. Sie dient also nicht zur Geheimhaltung, sondern zur leichteren Übermittlung über einem Informationskanal.

Verschlüsselung Im Gegensatz dazu hat die Verschlüsselung den Zweck, die Decodierung nur jenen Menschen zu ermöglichen, die über eine Informa-

[1] Dies ist die Hexadezimaldarstellung $\{0 - 9, A - F\}$ des Bitstrings.
[2] UTF-8 und ASCII sind bekannte Standards, die Unicode implementiert haben.

tion, den Schlüssel, verfügen. Dazu werden unter anderem sogenannte *trap door functions* verwendet, also Abbildungen bei denen es schwer ist, eine Umkehrfunktion zu finden.

Ein Beispiel solcher Funktionen ist die Primzahlfaktorisierung. Wird eine Zahl gewählt, die das Produkt lediglich zweier (hinreichend großer) Primzahlen ist, ist es schwer, die Primzahlen aus der gegebenen Zahl zu errechnen. Es wird aber keine Mühe machen, zwei Primzahlen zu multiplizieren.

2.2 Informationsübertragung

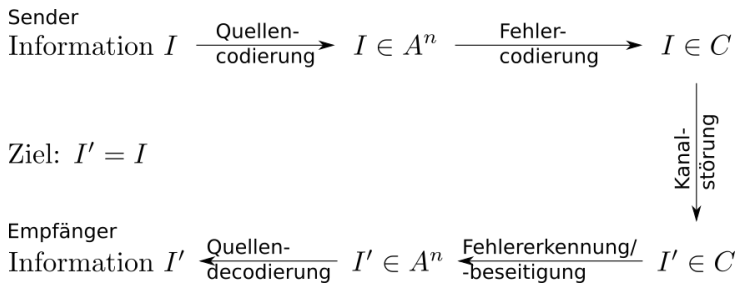


Abb. 2.1: Schema einer Informationsübertragung

Wenn eine Information, etwa Buchstaben, Frequenzen und Amplituden oder auch Farbpixel, übertragen werden, muss sie in Wörter mit Buchstaben aus einem Quellenalphabet A abgebildet werden. Das Quellenalphabet ist dabei abhängig vom Kanal, also der Übertragungsweise. Im Falle der CD ist es das Binärsystem $A = \mathbb{Z}_2 = \{0, 1\}$ und bei den meisten Warencodierungen das Ziffernsystem $A = \mathbb{Z}_{10} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

Die Information, also jeder Buchstabe oder jede Frequenz und Amplitude bzw. jede mögliche Farbe, wird dann einem *Codewort* mit einer bestimmten Blocklänge zugeordnet. Die Quellencodierung ist also eine Abbildung einer Information in Wörtern aus A^n, wobei n die Länge der Wörter ist (Blocklänge).

Ist die Information im Quellenalphabet abgebildet, wird die Funktion der Fehlercodierung angewendet. Sie bildet jedes Quellencodewort in ein Fehlercodewort ab. Dieses Codewort ist länger als das ursprüngliche Quellencodewort und so konstruiert, dass Fehler, die aufgrund des störanfälligen Kommunikationskanals auftreten können, erkannt und eventuell auch beseitigt werden können.

Nach der Übertragung der Codewörter über den Kommunikationskanal wird überprüft, ob Fehler gefunden wurden und ob diese reparabel sind.

Danach wird die Information im Quellenalphabet als Originalinformation (Text, Ton, Bild) dargestellt.

Beispiel Als Information sei das Wort „TUWIEN“ gegeben. Das Quellenalphabet soll auf einer Zuordnung der Buchstaben zu ihrer Position im Alphabet ($A = 1,\ B = 2,\ C = 3, \ldots, Z = 26$) beruhen. Die Quellencodierung von „TUWIEN“ ergibt also im Quellenalphabet $\mathbb{Z}_{26}$ (20, 21, 23, 9, 5, 15). Nun wird eine simple Fehlercodierung angewendet, die im besten Fall erlaubt, einen Fehler zu erkennen: Falls die Summe der Zahlen gerade ist, wird eine Null angehängt, sonst eine Eins. Die Information als Codewort (20, 21, 23, 9, 5, 15, 1) ist für die Übertragung bereit gemacht.

Das empfangene Codewort laute nun (20, 16, 23, 9, 5, 15, 1). Mittels Neuberechnung der Fehlerbits wissen wir, dass ein Fehler entstanden ist. Das empfangene Wort heißt nach der Decodierung „TPWIEN“. Die natürliche Sprache verfügt über genügend Redundanz, sodass eine Korrektur bei bekanntem Kontext ohne Probleme möglich ist. In anderen Fällen hilft die Grammatik (Satzstellung, Groß- und Kleinschreibung, Fälle) dabei, Fehler zu erkennen und zu beseitigen. Wiegen die Fehler allerdings schwerer, so muss eine Fehlercodierung angewendet werden, die es auch erlaubt, Fehler zu korrigieren. Dies benötigt im Allgemeinen jedoch längere Codewörter und führt damit zu einer geringeren Informationsdichte.

2.3 Grundlegende Begriffe

Um über Codierung sprechen zu können, ist es wichtig, einige Begriffe zu verstehen.

2.3.1 Alphabet

Als Alphabet A sei die Menge der Zeichen bezeichnet, die zur Bildung von Wörter benutzt wird. Dabei wird zwischen dem Quellenalphabet und dem Codealphabet unterschieden.

Beispiele

- Auf der CD und DVD wird das Binärsystem $A = \mathbb{Z}_2 = \{0, 1\}$ benutzt (siehe Tabelle 1.1).
- Die Brailleschrift besteht aus einer (3×2)-Matrix aus Punkten, die entweder erhoben, also tastbar, oder nicht erhoben sind (siehe Tabelle 4.1).

- Das Winkeralphabet besteht aus 2 Flaggen, die je im Nadirwinkel[3] $\alpha = 0°, 45°, 90°, 135°, 180°, 225°, 270°, 315°$ gehalten werden können.
- Das Hexadezimalsystem benutzt das Alphabet $A = \{0 - 9, A - F\}$, wobei $A = 10, B = 11, C = 12, D = 13, E = 14, F = 15$.

2.3.2 Code

„Ein Code entsteht durch die Abbildung eines Quellenalphabets in ein Codealphabet, wobei das Codealphabet ganz bestimmte gewünschte Eigenschaften aufweist, die das Quellenalphabet nicht hat.“ [2, S. 4] Die Abbildung selbst muss bijektiv sein. Das Codierungsverfahren ist dann die Abbildungsvorschrift.

Beispiele

- Berechnung des Dezimalwertes eines Binärstrings:

$$x = \sum_{i=0}^{n} z_i \cdot 2^i \tag{2.1}$$

$$= z_0 2^0 + z_1 2^1 + z_2 2^2 + \dots$$

$$(0,1,1,0): \quad x = 0 \cdot 2^0 + 1 \cdot 2 + 1 \cdot 2^2 + 0 \cdot 2^3 = 6 \tag{2.2}$$

- Berechnung des Dezimalwertes eines Hexadezimalstrings:

$$x = \sum_{i=0}^{n} z_i \cdot \ 16^i \tag{2.3}$$

$$= z_0 16^0 + z_1 16^1 + z_2 16^2 + \dots$$

$$(A,9,C,6): \quad x = 6 \cdot 16^0 + 12 \cdot 16^1 + 9 \cdot 16^2 + 10 \cdot 16^3 = 43462 \tag{2.4}$$

Bemerkung

- Um die Einstelligkeit zu bewahren, werden beim Hexadezimalsystem den Zahlen 10 bis 16 die Buchstaben A bis F zugeordnet.
- Bei der Berechnung eines Dezimalwertes ist stets von rechts nach links zu rechnen. Das hat den Vorteil, dass ohne Änderung der Reihenfolge Bits hinzugefügt werden können. (0, 0, 1, 0, 1) ist also das Gleiche wie (1, 0, 1), nämlich 5.

[3] Der Nadirwinkel ist der Winkel, der vom Lot, also senkrecht zum Boden, ausgehend gemessen wird.

2.3.3 Codewort

Ein Codewort ist ein n-Tupel aus $C \subset A^n$, wobei C die Menge aller Codewörter ist. Wenn, wie hier, die Länge gleich bleibt, spricht man von einem *Blockcode* der *Blocklänge* n. Im Folgenden ist immer von Blockcodes die Rede.

2.3.4 Hammingabstand

Wenn Codewörter mit gleicher Länge n definiert werden und A ein Körper ist, so können sie als Vektoren im n-dimensionalen Vektorraum A^n über A darstellt werden. Je zwei unterschiedliche Vektoren unterscheiden sich dabei in mindestens einer Stelle. Es kann nun eine Metrik definiert werden, die es erlaubt, den Abstand zwischen je zwei Vektoren zu ermitteln, indem man die Anzahl der unterschiedlichen Stellen zählt. Diese Zahl heißt *Hammingabstand* $h(\vec{c_i}, \vec{c_j})$:

$$\vec{c_i} = (a_{i1}, a_{i2}, \ldots, a_{in}) \in C \subseteq A^n \tag{2.5}$$

$$h(\vec{c_i}, \vec{c_j}) = |\,\{a_{it} | a_{it} \neq a_{jt};\ t = 1, \ldots, n\}\,| \tag{2.6}$$

Untersucht man den Hammingabstand aller Paare von Codewörtern, existiert ein kleinster Abstand. Es gibt also Paare von Codewörtern, deren Hammingabstand kleiner oder gleich ist, wie der Hammingabstand aller anderen Paare. Der Hammingabstand von zwei solchen Codewörtern wird als *Minimaldistanz* d bezeichnet. [2, S. 18]

$$d = min\,\{h(\vec{c_i}, \vec{c_j}) | \vec{c_i}, \vec{c_j} \in C\} \tag{2.7}$$

Beispiele

- Der Hammingabstand $h(\vec{x}, \vec{y})$ von $\vec{x} = (1, 0, 1, 1, 1)$ und $\vec{y} = (0, 0, 1, 1, 0)$ ist 2.

- Der Hammingabstand des sogenannten Paritätscheck-Code (siehe Tabelle 2.1) ist 2.

- Der Hammingabstand des dreifach Wiederholungscodes (siehe Tabelle 2.2) ist 3.

x	$f(x)$
$(0,0)$	$(0,0,0)$
$(0,1)$	$(0,1,1)$
$(1,0)$	$(1,0,1)$
$(1,1)$	$(1,1,0)$

Tabelle 2.1: Paritätscheck

x	$f(x)$
(0)	$(0,0,0)$
(1)	$(1,1,1)$

Tabelle 2.2: 3-fach Wiederholungscode

Satz 2.3.4 Ein Code kann genau dann jede Kombination von t oder weniger Fehlern entdecken bzw. korrigieren, wenn der Hammingabstand zwischen zwei beliebigen, verschiedenen Codewörtern mindestens $t+1$ bzw. $2t+1$ ist. [5, S. 95]
Beweis: Für $d = t+1$ gilt, dass bei einem Codewort mindestens $t+1$ Stellen geändert werden müssen, um ein anderes Codewort zu erhalten. Führt ein Fehler also zu einer Änderung von t Stellen, weiß man mit Sicherheit, dass es sich um kein gültiges Codewort handelt. Eine Fehlererkennung ist also möglich. Mathematisch ausgedrückt:

$$d \geq t+1, c_1 \in C: \quad h(c_1, c_2) < t+1 \Rightarrow c_2 \notin C \tag{2.8}$$

Um welches es sich allerdings handelt ist nicht klar, da es möglich wäre, dass nur mehr eine Stelle geändert werden müsste, um ein gültiges Codewort zu erhalten, das allerdings nicht dem ursprünglichen Codewort entspricht. Eine Fehlerkorrektur ist also nicht möglich.

Für $d = 2t+1$ gilt, dass bei einem Codewort mindestens $2t+1$ Stellen geändert werden, müssen um ein anderes Codewort zu erhalten. Führt ein Fehler also zu einer Änderung von t Stellen, weiß man mit Sicherheit, um welches ursprüngliche Codewort es sich gehandelt hat, wenn man von folgender Prämisse ausgeht:

„Das zum Empfangenen, eventuell fehlerbehafteten Wort nächstliegende Codewort ist dasjenige Codewort, welches von der Nachrichtenquelle ausgesandt wurde.“[5, S. 97]

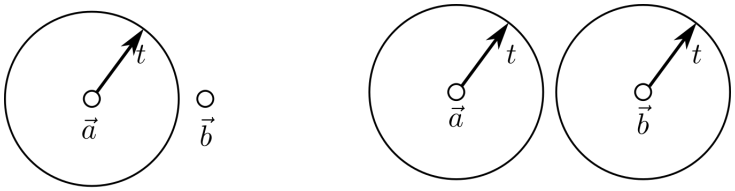
Denn der Abstand zum ursprünglichen Codewort ist lediglich t, der Abstand zu allen anderen Codewörtern ist größer oder gleich $t+1$. Damit ist nicht nur eine Fehlererkennung, sondern auch eine Fehlerkorrektur möglich. Mathematisch beschrieben:

$$d \geq 2t+1, c_1 \in C \subset A^n, c^* \in A^n:$$

$$h(c_1, c_1^*) < t \Rightarrow h(c_1^*, c_i) \geq t+1, \forall c_i \in C \tag{2.9}$$

$$\Rightarrow c_1^* := c_1 \tag{2.10}$$

Die Umkehrung der obigen Aussage wird aus der nachstehenden Abbildung 2.2 offensichtlich, wodurch Satz 2.3.4 bewiesen ist.



$t \leq h(\vec{a}, \vec{b}) - 1$ $\qquad$ $2t \leq h(\vec{a}, \vec{b}) - 1$

Abb. 2.2: Abstand zweier Punkte

2.4 Computercodierungen

Computer verwenden das Binärsystem. Es handelt sich hierbei um einen Code über dem Alphabet $\mathbb{Z}_2 = \{0, 1\}$, dem sogenannten *Binärcode*. Die Information wird meist in Blöcke mit gleichbleibender Länge geteilt. Nur zur besseren Lesbarkeit werden Codewörter von Computern oft in anderen Systemen, wie etwa dem Hexadezimalsystem[4], angegeben.

Beispiel Die MAC-Adresse (engl. *Media-Access-Control*) eines Netzwerkadapters ist seine eindeutige sechsstellige Identifikationsnummer, die üblicherweise hexadezimal geschrieben wird. Die sechs Stellen enthalten je zwei hexadezimale Ziffern, also einen Zeichenvorrat {00-FF}. Dabei werden je 8 Bit zu einem Byte zusammengefasst. Es stehen also weit mehr als 281 Billionen (16^{12}) verschiedene Adressen zur Verfügung. Tabelle 2.3 zeigt die Darstellungsarten der MAC-Adresse 00-80-41-ae-fd-7e, wobei statt A,...,F Kleinbuchstaben verwendet werden. Zu beachten ist, dass die HEX-Darstellung davon die kürzeste ist. Eine Umrechnungstabelle ist in Tabelle 4.2 im Anhang auf Seite 58 zu finden. Die Umrechnung der Hexadezimaldarstellung in die Dezimaldarstellung ist auf Seite 15 beschrieben.

HEX	BIN	DEC
00	0000 0000	0
80	1000 0000	128
41	0100 0001	65
ae	1010 1110	174
fd	1111 1101	253
7e	0111 1110	126

Tabelle 2.3: Darstellung einer MAC-Adresse

[4] hexa = sechs, decem = zehn, also ein System zur Basis 16.

2.5 Codierungstypen

Die grobe Unterscheidung zwischen Quellencodierung und Fehlercodierung, letztere wird auch Kanalkodierung genannt, wurde bereits in Kapitel 2.2 behandelt. Im Folgenden wird hier genauer auf die Aufgaben der beiden Codierungen eingegangen.

2.5.1 Quellencodierung

Die Quellencodierung ist eine injektive Funktion, die am Anfang der Informationsübertragung steht. Sie hat zur Aufgabe, Informationen aus Wörtern über einem Alphabet I in Wörter aus dem Quellenalphabet A abzubilden

$$\begin{aligned} f_C(x) : I &\to C \subset A^n \\ \vec{x} &\mapsto \vec{x}', \end{aligned} \tag{2.11}$$

wobei I Wörter über einem üblichen Alphabet einer Sprache sind, Audiosignale sind oder sonst eine Information sein kann. C ist die Menge aller möglichen Codewörter, A die Menge aller möglichen Elemente, aus denen Codewörter gebildet werden und n die Wortlänge. Mit dieser Zuordnung wird versucht, folgende drei Anforderungen zu erfüllen:[8, S. 2]

1. Zu bewerkstelligen, dass alle gesendeten Symbole mit ungefähr der gleichen Wahrscheinlichkeit auftreten,
2. eine optimale Ausnützung der Übertragungskapazität des Kanals sicherzustellen,
3. eventuell die Information zu komprimieren. (Beispiel: Matrikelnummern, Autokennzeichen, Bankomatkarten, ...)

ad 1. Wenn man eine von zehn möglichen Zuständen im Binärsystem codieren will, muss man zumindest eine Blocklänge von vier wählen[5]. Bei der Zuordnung der Zustände zu einem Codewort ist es daher nicht sinnvoll, der Reihe nach vorzugehen ($0 \to 0000, 1 \to 0001, 2 \to 0010, 3 \to 0011, \ldots, 9 \to 1000$), da das letzte Bit genau einmal den Wert 1 einnimmt. Besser wäre die Zuordnung in Tabelle 2.4 zu treffen. Zu beachten ist, dass an jeder Stelle sowohl 1 als auch 0 gleich oft vorkommen.

ad 2. Die Übertragungskapazität gibt die höchste Bitrate an, bei der Information gerade noch fehlerfrei übertragen werden kann. Für eine genaue Betrachtung des Begriffes „Kapazität“ sei auf [9, S. 199ff] verwiesen.

[5] $2^4 = 16 > 10 > 2^3 = 8$.

0	0000
1	0001
2	0010
3	0100
4	1000
5	1111
6	1110
7	1101
8	1011
9	0111

Tabelle 2.4: Binärzuordnung von zehn Zuständen

ad 3. Wenn der Zustand einer Ampel an die Zentrale übertragen wird, reicht es, den Zustand in Kurzform (R... Rot, RO... Rot/Orange, O... Orange, G... Grün, GB... Grün blinkend) zu übertragen.

Es folgen nun einige Beispiele für Quellencodierungen.

Beispiel Die Binärcodierung ist eine Darstellung von ganzen Zahlen $\mathbb{Z}$ als n-Tupel von Nullen und Einsen. Dabei werden die Ziffern mit den Potenzen in absteigender Reihenfolge multipliziert. Für die Dezimaldarstellung von (1, 1, 0, 0, 1) berechnet man damit:

$$x = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 \tag{2.12}$$

$$x = 16 + 8 + 1 = 25 \tag{2.13}$$

Allgemein gilt also

$$x = \sum_{i=0}^{n} z_i \cdot b^i, \tag{2.14}$$

mit b... Basis, z_i... Ziffer an der Stelle i. Mithilfe der Gleichung (2.14) können Zahlen vom Hexadezimal-, Oktal-, Binär- und jedem anderen System in Dezimaldarstellung umgerechnet werden.

Für die Berechnung einer Dezimalzahl a_0 als Tupel zur Basis b kann man folgendermaßen rekursiv vorgehen:

$$a_{n+1} = a_n / b \tag{2.15}$$

$$x_{n+1} = a_n \mod b \tag{2.16}$$

a_0 zur Basis b lautet dann $(x_m, x_{m-1}, \ldots, x_2, x_1)$.

Beispiel Für die Berechnung der Zahl 25 im Binärsystem ergibt sich folgende Rechnung:

$$25 : 2 = 12 \qquad 25 \mod 2 = 1 \tag{2.17}$$
$$12 : 2 = 6 \qquad 12 \mod 2 = 0 \tag{2.18}$$
$$6 : 2 = 3 \qquad 6 \mod 2 = 0 \tag{2.19}$$
$$3 : 2 = 1 \qquad 3 \mod 2 = 1 \tag{2.20}$$
$$1 : 2 = 0 \qquad 1 \mod 2 = 1 \tag{2.21}$$

Die Zahl 25 lautet im Binärsystem also $(1, 1, 0, 0, 1)$.

Beispiel Der ASCII-Code (ASCII, engl. *American Standard Code for International Interchange*) ist ein 1967 festgelegter Standard, der einen Buchstaben einem eindeutigen 7-Bit-String zuordnet. Er wurde später zu mehreren (länderspezifischen) 8-Bit-Codes erweitert.

Beispiel Ein Computer codiert Buchstaben im Standard *Unicode*, einer Erweiterung, die allen möglichen gängigen Schriftzeichen Rechnung trägt. Bekanntere Standards wie ASCII und UTF-8 sind in Unicode eingebettet und damit kompatibel.

Beim Unicode-Standard ist jedes Symbol (Buchstabe, Ziffer, Sonderzeichen, ...) einem sogenannten Codepunkt zugeordnet. Die *Blocklänge* eines Codepunktes von Unicode ist 6. Unicode verwendet als Alphabet das vierstellige Hexadezimalsystem und zwei vorangestellten Zeichen, nämlich (U+). Es würden somit etwa 17 Mio. Zeichen zur Verfügung stehen.[6] Ein Ausschnitt der Unicode-Tafel zur Codierung des Alphabets ist in Tabelle 4.1 auf Seite 57 zu finden.

2.5.2 Fehlercodierung

Die Fehlercodierung hat die Aufgabe, redundante Zeichen zur Information hinzuzufügen, um die Wahrscheinlichkeit eines Informationsverlustes bei Störung der Übertragung zu minimieren. Redundanz ist daher nicht als Störgeräusch oder Zufallszeichen zu verstehen, sondern als Verbreiterung der Information. Die Wahl der erlaubten Codewörter spielt hierbei eine entscheidende Rolle, da damit die Minimaldistanz mitbestimmt wird. Es ist auch wichtig, den Kanal zu kennen. Tritt eine Störung eher lokal auf (*Bündelfehler*), wie etwa bei der CD, so muss die Information gleichmäßig über das

[6] Tatsächlich stehen weit weniger Zeichen zur Verfügung, da Unicode einige Bereiche nicht zulässt.

Medium verteilt sein. Wirkt hingegen die Störung gleichmäßig über das Medium, so kann man die Redundanz nahe zur Information legen. Die Redundanz gewährleistet, dass eine Information, die verfälscht oder verloren geht, aus der übrigen Information zurückgewonnen werden kann.

Beispiel Ich rufe in einem Hotel an und würde gerne ein Zimmer reservieren. Die Dame an der Rezeption fragt nach meinem Namen. Ich sage: „FISCHER, Friedrich, Ida, SCHule, Emil, Richard.“ Ich habe gerade eine Codierung des üblichen Alphabets $\{A-Z, a-z\}$ zur Buchstabiertafel (siehe Tabelle 4.1, S. 57) getätigt. Des Weiteren wurde die Redundanz enorm erhöht, da im besten Fall der Buchstabe F (1 Buchstabe) zu Friedrich (8 Buchstaben) erweitert wurde. Allerdings handelt es sich hierbei um keinen Blockcode, da die einzelnen Buchstaben verschieden langen Codewörtern entsprechen.

Beispiel Bei dem *Paritäts-Check-Code* wird dem Code folgendermaßen ein weiteres Symbol angefügt:

$$f_c(x_0, x_1, \ldots, x_m) = \begin{cases} (x_0, x_1, \ldots, x_m, 0), & \sum x_i \equiv 0 \mod 2 \\ (x_0, x_1, \ldots, x_m, 1), & \sum x_i \equiv 1 \mod 2 \end{cases}$$

Dieser Code erlaubt über $\mathbb{Z}_2$ das Erkennen von bis zu einem Fehler, da sich bei jeder Änderung auch das Paritätsbit ändert. Zum Korrigieren von Fehlern bedarf es allerdings anderer Codierungen.

Beispiel Bei dem n-fachen *Wiederholungscode* wird die Information auf eine der folgenden Arten wiederholt. Allgemein:

$$f_1(x_0, x_1, \ldots, x_m) = (\underbrace{x_0, \ldots, x_0}_{\text{n-mal}}, \underbrace{x_1, \ldots, x_1}_{\text{n-mal}}, \ldots, \underbrace{x_m, x_m}_{\text{n-mal}}) \tag{2.22}$$

$$f_2(x_0, x_1, \ldots, x_m) = (\underbrace{x_0, x_1, \ldots, x_m}_{1}, \underbrace{x_0, x_1, \ldots, x_m}_{2}, \underbrace{\ldots}_{3\ldots n}) \tag{2.23}$$

Zahlenbeispiel ($n = 3$):

$$f_1 : (0, 1, 0) \mapsto (0, 0, 0, 1, 1, 1, 0, 0, 0) \tag{2.24}$$

$$f_2 : (0, 1, 0) \mapsto (0, 1, 0, 0, 1, 0, 0, 1, 0) \tag{2.25}$$

Dieser Code erlaubt das Erkennen von zwei Fehlern und die Korrektur von einem Fehler, da nach Satz 2.3.4 der Hammingabstand von zwei Codewörtern zum Erkennen von t Fehlern gleich $t+1$ und zur Korrektur von t Fehlern gleich $2t+1$ sein muss. In unserem Beispiel ist die Minimaldistanz 3. Allgemein gilt

ohne Beweis: Für eine Korrektur von t Fehlern muss ein $(2t+1)$-Wiederholungscode verwendet werden. [6, S. 5]

Der Nachteil des Wiederholungscodes ist seine Länge, da bei einem 3-fachen Wiederholungscode die Informationsdichte auf $\frac{1}{3}$ absinkt, bei einem n-fachen Wiederholungscode auf $\frac{1}{n}$.

Die zwei Fehlercodes, die bei der Audio-CD eingesetzt werden, sind ein Spezialfall von (n, k)-Codes (siehe 2.7), genannt (verkürzte) Reed-Solomon-Codes (siehe 2.8) und die EF-Modulation (siehe 3.1.5). Bevor diese behandelt werden, noch einige Eigenschaften von Codes.

2.6 Eigenschaften von Codes

2.6.1 Linearer Code

Sei A ein Körper, etwa $\mathbb{Z}_p, p \in \mathbb{P}$. Dann ist $(A^n, +, \cdot)$ ein Vektorraum.

Ein Code f_C heißt *Linearcode (linearer Code)*, wenn C ein Unterraum von A^n ist. Das heißt $(C, +, \cdot)$ ist ebenfalls ein Vektorraum mit

$$\vec{c_1} + \vec{c_2} \in C \tag{2.26}$$
$$\lambda \vec{c_1} \in C, \tag{2.27}$$

für alle $\vec{c_i} \in C$ und $\lambda \in A$.

Beispiel Ein Beispiel für einen linearen Code über $\mathbb{Z}_2$ ist folgender:

$$C = \{00000, 11010, 01101, 10111\} \tag{2.28}$$

Denn für jedes $c \in C$ gilt:

$$(00000) + c = c, \tag{2.29}$$
$$c + c = (00000) \tag{2.30}$$

und

$$(11010) + (01101) = (10111), \tag{2.31}$$
$$(11010) + (10111) = (01101), \tag{2.32}$$
$$(01101) + (10111) = (11010). \tag{2.33}$$

2.6.2 Zyklischer Code

Codes, deren Codewörter durch zyklische Verschiebung wieder ein Codewort ergeben, nennt man zyklische Codes. [2, S. 101] Mathematisch ausgedrückt: [4, S. 26]

$$c := (a_0, a_1, \ldots, a_{n-1}) \in C \Rightarrow (a_{n-1}, a_0, a_1, \ldots, a_{n-2}) \in C, \ \forall c \in C. \quad (2.34)$$

Tabelle 2.5 zeigt ein Beispiel. Zyklische Codes sind von großem Interesse für

$(0,0,0,0,0)$	$(0,0,1,0,1)$
$(0,0,0,0,1)$	$(0,1,0,1,0)$
$(0,0,0,1,0)$	$(1,0,1,0,0)$
$(0,0,1,0,0)$	$(0,1,0,0,1)$
$(0,1,0,0,0)$	$(1,0,0,1,0)$
$(1,0,0,0,0)$	$(1,1,1,1,1)$

Tabelle 2.5: Codewörter eines zyklischen Codes

die Codierungstheorie, da sie sehr schnell zu berechnen sind. Zur Berechnung ist es günstig, die Codes als Polynome darzustellen.

2.6.3 Polynomcode

Allgemein bildet man aus den Wörtern c eines Codes $C \subseteq A^n$ Polynome $c(x)$ vom Grad $\leq n-1$ durch

$$c(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1}. \quad (2.35)$$

Für den Beweis des unten stehenden Satzes benötigt man die Tatsache, dass die Multiplikation von zwei Polynomen vom Grad $\leq n-1$ wieder zu einem Polynom vom Grad $\leq n-1$ führt. Dies geschieht bei der Multiplikation im Allgemeinen nicht, da

$$grad(c_1(x) \cdot c_2(x)) = grad(c_1(x)) + grad(c_2(x)). \quad (2.36)$$

Um dem entgegenzuwirken, „fassen wir die Polynome als Elemente von $A[x]/(x^n - 1)$ auf, d. h. als Elemente des Polynomrings $A[x]$ faktorisiert nach dem von $x^n - 1$ erzeugten Hauptideal.“ [4, S. 26] Es wird also jedes Polynom modulo $(x^n - 1)$ gebildet.

Damit ist ein Mittel zur zyklischen Vertauschung geschaffen. Die Multiplikation mit x ergibt nämlich:

$$\begin{aligned} & x \cdot (a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1}) \mod (x^n - 1) \\ =& a_0 x + a_1 x^2 + a_2 x^3 + \ldots + a_{n-1} \\ =& a_{n-1} + a_0 x + a_1 x^2 + a_2 x^3 + \ldots + a_{n-2} x^{n-1}. \end{aligned} \quad (2.37)$$

Satz Jeder zyklische Linearcode ist ein Polynomcode.
Beweisskizze: [8, S. 16] Man ordnet jedem $c = (a_0, \ldots, a_n)$ ein Polynom $c(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}$ zu und zeigt:

1. Unter den Polynomen $\neq 0$ gibt es genau ein Polynom $g(x)$ von minimalem Grad.

2. Jedes $c(x), c \in C$, ist ein Vielfaches von $g(x)$

3. $grad(g(x)) = n - k$

2.6.4 Systematischer Code

Ein systematischer Code ist jener, bei dem die Prüfbits immer an der gleichen Stelle sind. Sie haben den Vorteil, dass die Codewörter des Codealphabets direkt, zumeist am Anfang oder Ende des Codewortes zu finden, ablesbar sind. Wenn also die Fehlerüberprüfung ergibt, dass kein Fehler vorliegt, muss keine Berechnung angestellt werden, um die Information aus dem Fehlercode zu extrahieren. Es werden lediglich die Fehlerbits weggestrichen.

Es sind sowohl alle Warenkennzeichnungen mit Prüfbits, als auch die RS-Codierung der CD, systematische Codes.

2.6.5 Optimaler Code

Es ist schon bei der Auswahl des Quellcodealphabets darauf zu achten, dass der mittlere Abstand aller Codewörter möglichst groß ist. Es existieren folgende zwei gegensätzliche Wünsche:

- So viel Information wie möglich auf dem Medium unterzubringen,

- die Fehlererkennungs-/ und Fehlerkorrekturrate möglicht groß zu machen.

Die größte Informationsdichte erreicht man natürlich durch Weglassen der Fehlercodierung. Die größte Fehlerkorrekturrate erreicht man, indem man eine Information möglichst oft wiederholt. Dass diese beiden Extrema nicht alltagstauglich sind, ist offensichtlich.

Werden in einer Fehlercodierung zu jedem Codewort der Länge k, $(n-k)$ Stellen an Redundanz hinzugefügt, so wird im Fall eines Linearcodes ein

k-dimensionaler Vektorraum in ein n-dimensionalen Vektorraum abgebildet.

$$f : A \to C;\ A \subset \mathbb{Z}_2^k, C \subset \mathbb{Z}_2^n \tag{2.38a}$$

$$f(\vec{x}) \mapsto \vec{x}' \tag{2.38b}$$

$$f(x_1, x_2, \ldots, x_k) = (x_1, x_2, \ldots, x_k, \underbrace{x_{k+1}, x_{k+2}, \ldots, x_n}_{(n-k)-Stellen}) \tag{2.38c}$$

Ein binärer (n, k)-Code, also ein Code, der zu k Informationsbits $n - k$ Prüfstellen hinzufügt, heißt *optimal*, wenn gilt:

- Er kann bis zu $n - k$ Fehler entdecken,
- er kann bis zu $\lfloor \frac{n-k}{2} \rfloor$ Fehler korrigieren.

Diese Eigenschaft erreicht man, wenn die Minimaldistanz eines (n, k)-Codes gleich $n - k + 1$ ist.

2.7 (n, k)-Code

Als (n, k)-Code über $GF(q)$, also ein Code, dessen Alphabet $A = GF(p^m)$ ist, wird ein Code bezeichnet, dessen Blocklänge k vor der Fehlercodierung und n nach der Fehlercodierung ist $(n > k)$. Es wird also ein Wort $w \in A^k$ auf ein Codewort $c \in C \subseteq A^n$ abgebildet.

Ist C ein Polynomcode so erfolgt die Berechnung einer Codierung mittels eines sogenannten *Generatorpolynoms* $g(x)$ vom Grad $n - k$. Dabei wird das Generatorpolynom mit den Polynomen, welche den Codewörtern entsprechen, multipliziert. Das Ergebnis ist das Codewort, das allerdings nicht systematisch ist, da man das ursprüngliche Wort nicht direkt herauslesen kann.

Sei A^n der Vektorraum über dem endlichen Körper $GF(q)$ und C ein linearer (n, k)-Code über $GF(p^m)$. Wie bereits in 2.6.1 angeführt, kann

$$C \subseteq GF(p^m)$$

als ein k-dimensionaler Untervektorraum von A^n aufgefasst werden.

Beispiel Sei C ein (n, k)-Polynomcode mit

$$g(x) = x^3 + x + 1.$$

als Generatorpolynom. Das Nachrichtenwort $w = (w_1 \ldots w_m)$ sei identifiziert mit dem Polynom $p_w(x) = w_1 + w_2 x + \ldots + w_m x^{m-1}$. Es lassen sich nun alle Nachrichtenwörter mit der Vorschrift

$$f_C(a) := p_a \cdot g(x), a \in C,$$

berechnen, allerdings erhält man damit keinen systematischen Code.

$$
\begin{aligned}
0000 : 0 \cdot g(x) &= 0 && \Rightarrow 0000000 && (2.39)\\
0001 : 1 \cdot g(x) &= 1 + x + x^3 && \Rightarrow 0001011 && (2.40)\\
&\vdots && \vdots \\
0101 : (x^2+1) \cdot g(x) &= x^5 + x^2 + x + 1 && \Rightarrow 0100111 && (2.41)\\
&\vdots && \vdots \\
1111 : (x^3+x^2+x+1) \cdot g(x) &= x^6 + x^5 + x^3 + 1 && \Rightarrow 1101001 && (2.42)
\end{aligned}
$$

Tabelle 2.6 zeigt die vollständige Zuordnung der Nachrichtenwörter zu ihren Codewörtern.

Wort	Codewort
0000	0000000
0001	0001011
0010	0010110
0011	0011101
0100	0101100
0101	0100111
0110	0111010
0111	0110001
1000	1011000
1001	1010011
1010	1001110
1011	1000101
1100	1110100
1101	1111111
1110	1100010
1111	1101001

Tabelle 2.6: Codierung eines (7,4)-Codes

Beispiel Ein Code, der systematisch ist und sich sehr schnell mittels Schieberegister durchführen lässt, ist der Folgende:

Sei $g(x) \in GF(p^n)[x]$ vom Grad $n-k$. Die Nachrichtenwörter sind wie oben identifiziert. $r_{g(x)}(p(x))$ bezeichnet den Rest bei der Division $p(x)/g(x)$. Die Codierungsvorschrift lautet mit $a \in A^k$

$$f_C(a) := p_a(x) \cdot x^{n-k} - r_{g(x)}(p_a(x) \cdot x^{n-k}).$$

Der erste Term ist vom Grad $\geq n-k$ und der zweite, der den Rest bezeichnet, vom Grad $< n-k$. Es ist also im Codewort das Nachrichtenwort direkt ablesbar, und die Fehlerkorrekturbits sind immer am Anfang des Codewortes zu finden (systematischer Code).

2.8 RS-Code

Der Reed-Solomon-Code, kurz RS-Code, wurde von den beiden Mathematikern *I. S. Reed* und *G. Solomon* entdeckt.[2, S. 142] Der Grundgedanke ist, dass ein Polynom vom Grad $m-1$ durch m Stützstellen eindeutig bestimmt ist. Wenn also ein Polynom vom Grad 1 (geometrisch interpretiert eine Gerade) zwei Stützstellen hat, ist es eindeutig definiert. Fügt man eine dritte Stützstelle hinzu, kann ein Fehler erkannt, jedoch nicht beseitigt werden.

Beispiel Information: (2, 3) soll codiert werden. Dazu versendet man eine Polynomfunktion vom Grad 1.

$$f(x) = a_1 x + a_0, \tag{2.43}$$

mit $f(0) = 2, f(1) = 3$. Daraus ergibt sich das zu dieser Nachricht eindeutige Polynom

$$f(x) = x + 2. \tag{2.44}$$

Nun fügt man ein weiteres Prüfbit hinzu: $f(2) = 4$. Es wird also das Wort

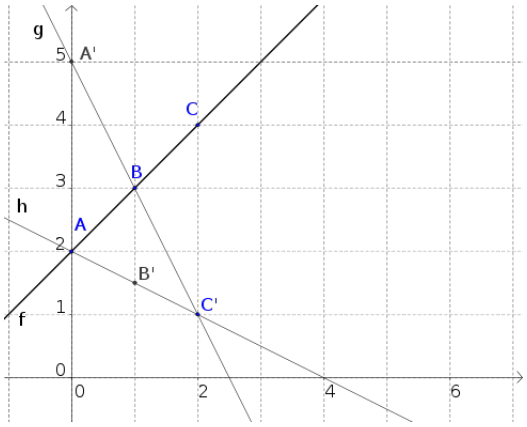


Abb. 2.3: Graphische Veranschaulichung des Beispiels

$(2, 3, 4)$ verschickt (Abb. 2.3: A,B,C). Beim Empfänger trifft das Wort $(2, 3, 1)$ ein (in Abb. 2.3: A,B,C'). Dieser weiß sofort, dass ein Fehler passiert ist, da die drei Punkte nicht auf einer Geraden liegen. Er kann jedoch nicht feststellen,

wo der Fehler liegt, da er nicht weiß, welche der möglichen Funktionen (Abb. 2.3: f, g, h) die richtige ist.

Um einen Fehler beheben zu können, bedarf es mindestens zwei zusätzlicher Stützstellen. Fügt man des Weiteren das Prüfbit $f(3) = 5$ hinzu, versendet also die Nachricht (2,3,4,5), wobei nur (2,3) die Information ist und (4,5) zur Fehlerkorrektion dient, ist es möglich, einen Fehler zu erkennen und diesen auch zu korrigieren. Allgemein gilt bei diesem Verfahren:

Satz [9, S. 403] Mit r zusätzlichen Prüfbits lassen sich

- r Übertragungsverluste kompensieren, wie nachstehend erläutert wird,
- r Übertragungsfehler erkennen,
- $\lfloor \frac{m}{2} \rfloor$ Übertragungsfehler korrigieren.

Ein Übertragungsverlust ist eine Verlust an Informationen, etwa $c_1 = (1, 0, 1, 0) \rightarrow c_2 = (1, 0, X, 0)$. Der Verlust ist immer erkennbar und die betroffene Stelle eindeutig. Beim Übertragungsfehler ist hingegen nicht klar, wo bzw. ob ein Fehler vorliegt. Wenn $c_1 = (1, 0, 1, 0)$ übertragen wird und $c_2 = (1, 0, 0, 0)$ beim Empfänger ankommt, dann wird der Fehler, wenn das Codewort c_2 existiert, ohne Fehlercode nicht erkannt.

Nachteil der obengenannten Methode ist, dass sowohl die Korrekturzahlen mit zunehmender Nachrichtenlänge stark wachsen als auch, dass die Berechnung des Polynoms stark anwächst.[7]

Beispiel Die Nachricht (4, 2, 10, 0, 4, 3, 7, 9, 12) soll codiert werden, wobei bis zu drei Fehler korrigierbar sein sollen. Es ist daher nötig, zu den neun Nachrichtenstellen mindestens sechs Prüfstellen hinzuzufügen. Das Polynom ($f(x) \in \mathbb{Q}[x]$), das sich aus den Nachrichtenzahlen errechnen lässt, ist vom

[7] Berechnung z. B. mittels der Interpolationsformel von Lagrange:[3, S. 67]

$$f(x) = \sum_{i=1}^{n} b_i \frac{q_i(x)}{q_i(\alpha_i)}$$

, mit

$$q_i(x) = \prod_{\substack{1 \leq j \leq n \\ i \neq j}} (x - \alpha_j), b_i = f(\alpha_i)$$

Grad 8 und lautet (vgl [9, S. 403f])

$$f(x) = \frac{43}{3360}x^8 - \frac{2123}{5040}x^7 + \frac{4109}{720}x^6 - \frac{7379}{180}x^5 + \frac{240703}{1440}x^4 - \frac{274991}{720}x^3 + \frac{139742}{315}x^2 - \frac{20491}{105}x + 4. \tag{2.45}$$

Durch Hinzunahme der sechs Korrekturstellen lautet das neue Codewort $(4, 2, 10, 0, 4, 3, 7, 9, 12, 844, 6992, 33202, 117108, 340669, 863709)$. Es ist offensichtlich, dass diese Form nicht alltagstauglich ist. In der Praxis wird daher als Grundlage ein Galoisfeld benutzt.

Definition Ein (n,k)-Code über $GF(p^m)$ der Länge $n = p^m - 1$ mit Generatorpolynom

$$g(x) = (x-1)(x-a)(x-a^2)\cdots(x-a^{n-k-1}),$$

heißt RS-Code. [4, S. 34] Dabei ist a ein primitives Element[8] von GF(p^m), p prim und $k = n - grad(g(x))$. Dieser Code ist optimal.

Verkürzung Bei der CD wird ein verkürzter RS-Code verwendet. Verkürzung heißt, dass ausgehend von einem (n, k)-Code C der Minimaldistanz d die ersten r Stellen in jedem Nachrichtenwort 0 gesetzt werden. Dadurch erhält man einen $(n-r, k-r)$-Code mit Minimaldistanz $\geq d$. Dieser Code verliert die Eigenschaft zyklisch zu sein, kann aber leicht implementiert werden.[3, S. 101]

[8] Ein primitives Element a erzeugt durch Potenzierung alle Elemente $\neq 0$ von $GF(p^m)$

Kapitel 3

Anwendungen

3.1 CD-Codierung

3.1.1 Technischer Überblick

Die CD ist eine etwa handgroße runde Plastikscheibe, die im Zentrum ein Loch hat. Auf der Plastikscheibe befindet sich eine reflektierende Schicht aus Polycarbonat. Auf dieser Schicht ist die Information (Audio-Signal) als Gruben (*pit*) bzw. Flächen (*land*) eingebettet. Die CD wurde so designed, dass sie leicht von einem erwachsenen Menschen in einer Hand gehalten werden kann. Technische Daten können der Tabelle 3.2 auf Seite 38 entnommen werden. Im Gegensatz zur Schallplatte, dem analogen Vorgänger der CD, wird die spiralförmig aufgetragene Information von Innen nach außen gelesen. Ein weiterer Unterschied besteht in der Rotationsgeschwindigkeit. Eine CD bewegt sich nach außen hin langsamer, es wird also die Bahngeschwindigkeit (= Lesegeschwindigkeit) konstant gehalten, nicht die Winkelgeschwindigkeit. Dies führt dazu, dass die Informationsdichte auf der CD gleich verteilt ist.

Abb. 3.1: Datenseite einer CD

Die Information, also der Strom an Nullen und Einsen, wird ausgelesen, indem ein Laser auf die Bahn aus Tälern und Ebenen gerichtet wird (siehe die

ganzseitige Abb. 3.2 auf Seite 33). Die Täler sind etwa 200nm tief (siehe Abb. 3.3), was einem Viertel der Wellenlänge des roten Lasers (780nm) entspricht. Das führt zu Interferenzerscheinungen, die die Intensität des zurückgeworfenen Lichtes vermindern. Die Photodiode registriert die Änderung der Intensität und setzt den Übergang als 1.

Wenn eine CD ausgelesen wird, muss der Datenstrom zur Decodierung in einen Puffer (*Buffer*) geladen werden. Darüber hinaus sind auf einer CD nicht nur Audioinformationen, sondern auch Metadaten (Zeit, Tracknummer, Länge des Tracks, ...) abgelegt. All diese Informationen müssen in den Puffer geladen und decodiert werden. Da der Puffer nur über einen begrenzten Speicherraum verfügt, ist es wichtig, den Informationsfluss, d. h. die Lesegeschwindigkeit, steuern zu können und konstant zu halten.

3.1.2 Ablauf der CD-Codierung

Bei einer Audio-CD benötigt man acht Schritte, bevor die CD mit dem Datenstrom beschrieben werden kann. [5, S. 98] Die Tabelle 3.1 mit den Schritten und ihren Auswirkungen auf die Wortlänge ist auf Seite 35 zu finden.

1. **Analog-Digital-Wandler:** Die Audioinformation wird digitalisiert (siehe 3.1.3).

2. **Scrambler:** Der Datenstrom wird in Wörter zu 8 Bits (1 Byte) und 24 dieser Bytes werden zu Blöcken (*frames*) zusammengefasst, wobei abwechselnd zwei Bytes pro Audiokanal entnommen werden. Diese Durchmischung der Kanäle wird als *Scrambling* bezeichnet.

3. **C_2-Encoder:** Die einzelnen Blöcke werden einer $(28, 24)$-Fehlercodierung, mittels eines verkürzten Reed-Solomon-Codes über $GF(256)$ unterzogen (siehe 2.7). Es werden also zu 24 Bytes vier weitere Kontrollbytes hinzugenommen. Damit sind nun zwei fehlerhafte Bytes korrigierbar und Fehler vom Ausmaß von vier Bytes erkennbar.

4. **Interleaver:** Um Bündelfehler (z. B. durch Kratzer) zu beseitigen, werden die Daten räumlich verteilt (*interleaving*) (siehe 3.1.4).

5. **C_1-Encoder:** Die 28 Byte langen Wörter werden einer zweiten (n, k)-Codierung unterzogen. Dieses Mal ein (32,28)-verkürzter RS-Code über $GF(256)$ (siehe 2.7).

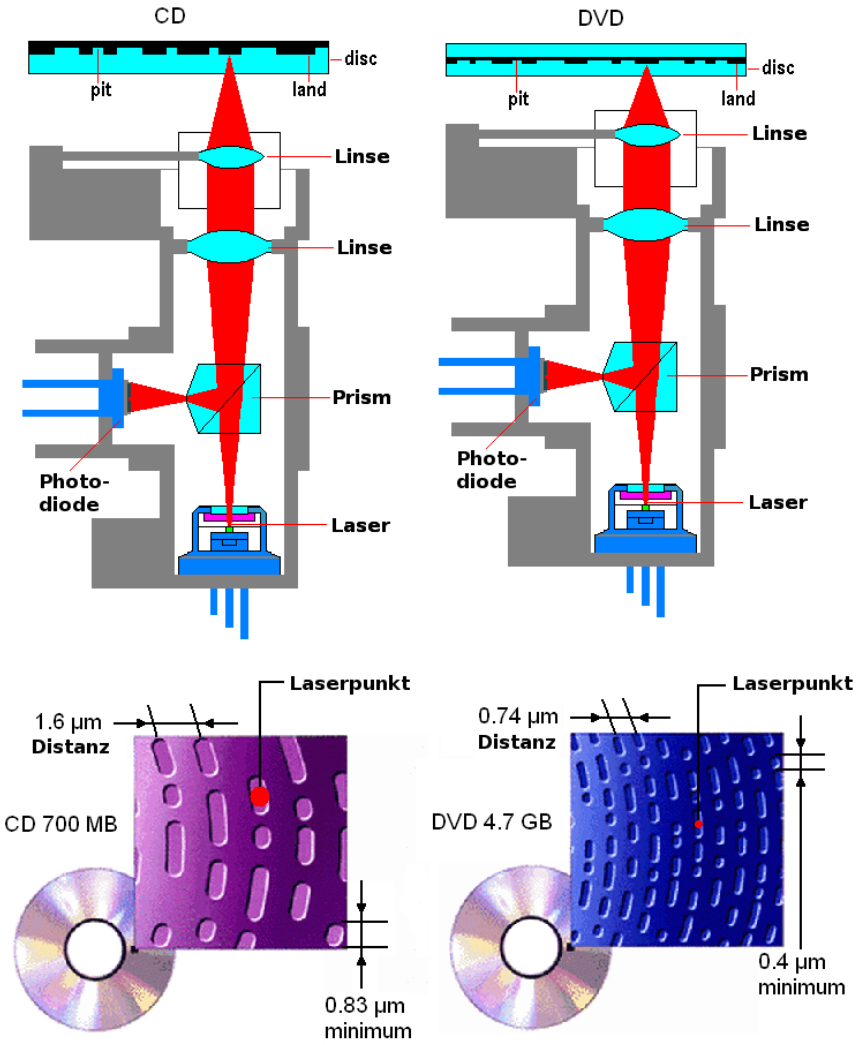


Abb. 3.2: Unterschiede zwischen CD/DVD, ©Wikimedia: Hoikka1

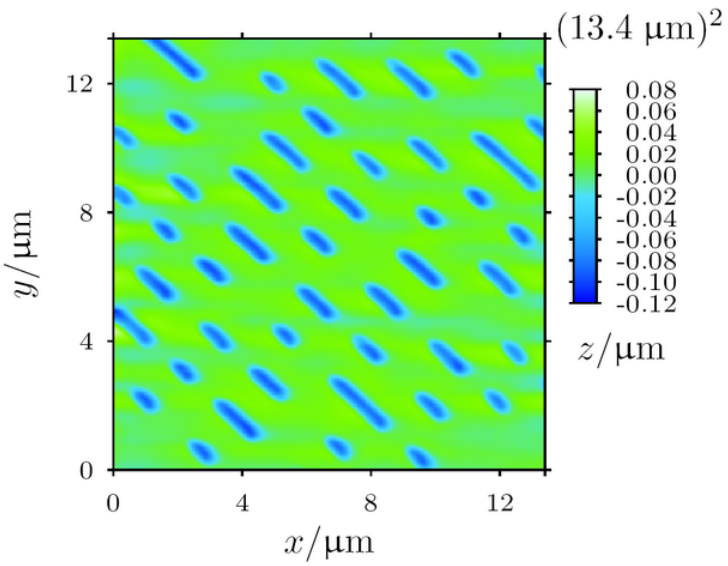


Abb. 3.3: Datenschicht mittels Rasterkraftmikroskop, ©Wikimedia: mrtz

6. **Informationszugabe:** Der Audioinformation werden Abspielinformationen hinzugefügt. Dabei wird jedem Wort 8 Bit (= 1 Byte) an Information vorangestellt. Diese werden mit P bis W bezeichnet (1. Bit: P, 2. Bit: Q, ..., 8. Bit: W). Der Q-Channel, also die Q-Bits mehrerer Wörter, enthält zum Beispiel das Inhaltsverzeichnis der CD. Ein Frame hat nun eine Länge von 33 Bytes.

7. **EF-Modulator:** Die Codewörter werden einer EF-Modulation unterzogen (siehe 3.1.5). Jedes Byte (= 8 Bits) wird durch 14 Bits dargestellt. Zusätzlich wird zwischen jedem neuen Wort ein drei Bit langes Verbindungsglied gesetzt. Damit ist ein Frame nun $33 \times 17 = 561$ Bits lang.

8. **Synchronisationsbits:** Abschließend werden Synchronisationsbits hinzugefügt. Diese sind 24 Bits lang und stellen eine innewohnende Uhr dar, die zum Auslesen der Daten gebraucht wird. Es handelt sich hier um folgende Bitabfolge:

   100000000001000000000010

   Danach wird ein Verbindungsbit der Länge drei angehängt, womit schlussendlich 27 Bits hinzugefügt sind. Ein Frame ist nun 588 Bits groß. Die Synchronisationsbits markieren, am Anfang stehend, den Beginn eines

Frames.[1]

Schritt	Länge
A/D-Wandler	24 Byte
Scrambler	24 Byte
C_2-Encoder: (28,24)-Code	28 Byte
Interleaver	24 Byte
C_1-Encoder: (32,28)-Code	32 Byte
Informationszugabe	33 Byte
EF-Modulation	561 Bit
Synchronisationsbits	588 Bit

Tabelle 3.1: CD-Codierungsschema

3.1.3 Quellencodierung: Codierung der Audioinformation

Beim Digitalisieren des Audiosignals gibt es zwei wichtige Werte:

- Bitrate (bps): Anzahl der möglichen diskreten Niveaus, in die ein kontinuierliches Signal eingeordnet wird. Die Einheit bps heißt „Bit pro Sekunde“.

- Abtastrate (Hz): Anzahl der Abtastungen pro Zeiteinheit.

Die Frequenz f ist der Kehrwert der Periodendauer T.

$$f = \frac{1}{T}.$$

Um eine Frequenz von $1/s = 1Hz$ darstellen zu können, sind zumindest drei Abtastpunkte erforderlich, die einen Abstand von $0,5s$ haben (Abb. 3.4: A,B,C). Es wird daher eine Abtastrate von mindestens $0,5Hz$ benötigt, um eine Frequenz von $1Hz$ darzustellen. Allgemein gilt:

[1] 98 Frames bilden einen Sektor, die kleinste Adressierungseinheit einer Audio-CD. Bei einer Audio-CD werden durchschnittlich 75 Sektoren pro Sekunde ausgelesen. [9, S. 428]

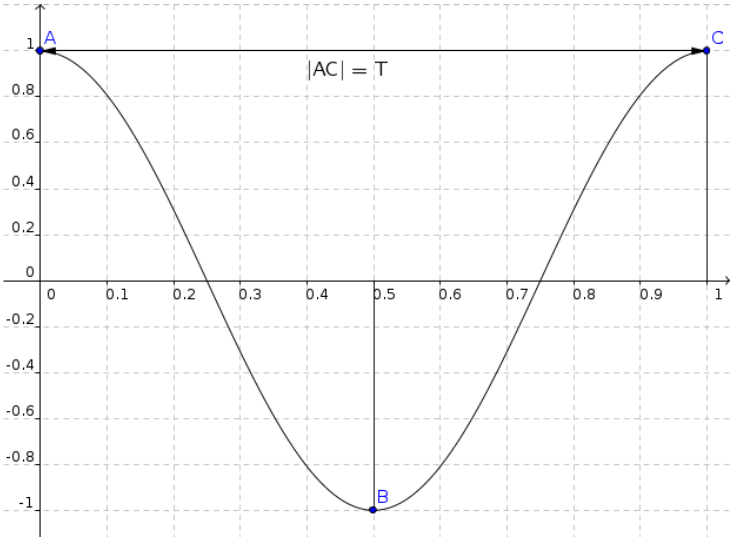


Abb. 3.4: Wellenlänge und Frequenz

Aus der Elektrotechnik Um eine bestimmte Frequenz darstellen zu können, benötigt man mindestens die doppelte Abtastfrequenz dieser Frequenz.

Bei der Audio-CD wird 4.4100 Mal pro Sekunde (pro Kanal) abgetastet (44,1 kHz).[2] Jeder abgetastete Wert wird einem von $2^{16} = 65536$ Niveaus zugeordnet. Pro Abtastung werden also zwei Bytes an Information erzeugt. [5, S. 91] Damit ergibt sich pro Kanal eine Bitrate von 88,2 kbps.

3.1.4 Interleaving

Auch bei der C_2-Codierung bestehen die Codewörter aus je 28 Bytes. Nun werden 28 Codewörter zusammengefasst und neu gruppiert. Das erste neue Wort besteht aus den ersten Bytes aller 28 alten Wörter. Das zweite neue Wort besteht aus den zweiten Bytes aller 28 alten Wörter usw. Die folgenden Gleichungen (3.1 a–d) zeigen die Codierungsvorschrift des Interleavings. Aus den Codewörtern $c_i \in C_{2_1}$ wird das Codewort $c_1^* \in C_{2_2}$ gebildet. Die Codewörter bestehen je aus 28 Bytes, die wie folgt neu organisiert werden:

$$f(c_i) : C_{2_1} \mapsto C_{2_2} \tag{3.1a}$$

$$c_i \to c_i^* \tag{3.1b}$$

$$c_i = (c_{i,1}, c_{i,2}, c_{i,3}, \ldots, c_{i,28}) \tag{3.1c}$$

$$c_i^* = (c_{1,i}, c_{2,i}, c_{3,i} \ldots, c_{28,i}) \tag{3.1d}$$

[2] Der Mensch hört im Frequenzspektrum 20–20.00Hz, also etwa halb so viel wie die Abtastrate.

Beispiel Nachstehende Gleichungen (3.2a–d) zeigen ein Beispiel mit Codewörtern der Länge drei. Daher muss auch immer eine Gruppe von drei Wörtern zum Interleaving zusammengefasst werden.

$$c_1 = (1, 1, 1) \tag{3.2a}$$
$$c_2 = (0, 0, 1) \tag{3.2b}$$
$$c_3 = (1, 0, 0) \tag{3.2c}$$
$$c_1^* = (1, 0, 1), c_2^* = (1, 0, 0), c_3^* = (1, 1, 0) \tag{3.2d}$$

3.1.5 Fehlercodierung: EF-Modulation

EF steht für *eight to forteen*. Es werden also die Informationen nicht in 8 Bits (= 1 Byte), sondern in 14 Bits gespeichert. Die Zuordnung ist so gewählt, dass zwei Voraussetzungen erfüllt werden.

- Zwischen je zwei Einsen stehen mindestens zwei Nullen.
- Zwischen je zwei Einsen stehen maximal zehn Nullen.

Damit diese Voraussetzungen auch zwischen zwei Codewörtern gilt, werden zwischen ihnen immer eine von drei möglichen „Verbindungsbits“ eingeführt. Diese sind gemäß der obigen Voraussetzung: [9, S. 427]

$$000, 001, 010, 100.$$

Die EF-Modulation hat zwei bedeutende Vorteile:

1. Die Minimaldistanz der Codewörter wird vergrößert, was zu einer besseren Fehlerkorrektur führt.
2. Auf physikalischer Ebene bedeutet 0 *„kein Wechsel“* und 1 *„Wechsel“* (von Pit auf Land bzw. umgekehrt). Durch die weite Streuung der Einsen ist die Konstruktion der Täler und Ebenen so zu bewerkstelligen, dass dies in einem kleineren Maßstab möglich ist als im Falle von nahe beieinander liegenden Erhebungen und Vertiefungen. Dadurch kann eine höhere Dichte an Information erreicht werden.

Die genaue Zuordnung der Audioinformationen zu deren Codewörtern ist in den sogenannten „Rainbow Books“[3] beschrieben.

[3] Die Audio-CD ist im Red Book

3.2 Entwicklungen der DVD

	CD	DVD
Durchmesser	120mm	120mm
Durchmesser (Loch)	7,5mm	7,5mm
Lichtdurchlässige Schicht	1,2mm	2x0,6mm
Lesegeschwindigkeit (1x)	1,2m/s	3,49 m/s
Abstand zweier Spurrinnen	$1,6\mu m$	$0,74\mu m$
Tiefe	$\approx 0,25 \cdot \lambda_{Laser}$	$\approx 0,25 \cdot \lambda_{Laser}$
Laserwellenlänge	780nm	635/650nm
kleinste Pit-Länge	$0{,}833\mu m$	$0,4\mu m$

Tabelle 3.2: Vergleich der Eckdaten von CD und DVD [15]

Die DVD (engl. *digital video disc*) ist der direkte Nachfolger der CD und wurde 1995 von Philips und Sony kreiert. [15] Sie war ursprünglich dafür konzipiert, Videos zu speichern, wurde dann aber auch, wie die CD, als Medium für andere Dateien genutzt.

Typen Es gibt zwei DVD-Typen: Die *Single-Layer*-DVD verfügt über eine Informationsschicht, die *Dual-Layer*-DVD verfügt über zwei Informationsschichten. Im Folgenden ist, wenn nicht anders angeführt, stets von der Single-Layer-DVD die Rede.

Laser Eine DVD verfügt über mehr Speicherplatz (4,7 Gigabyte[4]) als eine CD. Dies rührt daher, dass der verwendete Laser eine kürzere Wellenlänge aussendet. Dadurch ist der Laserspot kleiner und kann so kleinere Täler (*Pits*) bzw. Ebenen (*Lands*) auflösen. Dies führt zu einer höheren Datendichte.

Anforderungen Die Filmindustrie gab beim Designen der DVD eine Standard-Spielfilmlänge von 135 Minuten vor, die auf eine DVD zu passen hätte. Die Spezifikationen der Video-DVD, die daraufhin getroffen wurden, sind jedoch proprietär. Die Lizenz kostet über 4.000 Euro und ist mit einer Geheimhaltungsklausel verbunden. [18]

Abspielinformationen Im Gegensatz zur Audio-CD beherbergt die Video-DVD nicht nur Audioinformationen. Sie verfügt über eine Videospur,

[4] Dual-Layer-DVDs verfügen sogar über 8,5 Gigabyte.

die verschiedene Auflösungen und zwei mögliche Bildfrequenzen haben kann. Darüber hinaus verfügt sie über mehrere Audiospuren (Mehrsprachigkeit) mit zwei bis acht Kanälen[5], die eine Abtastrate von 48kHz bzw. 96kHz und eine Bitrate von 2^{16} bzw. 2^{24} möglichen Niveaus erreicht. Des Weiteren gibt es die Möglichkeit, eine von bis zu 32 Untertiteln anzeigen zu lassen und eine Menüstruktur zu speichern. [19]

(8,16)-Modulation Aufgrund der höheren Datendichte ist die DVD anfälliger für Kratzer und Schmutz. Um diesem Umstand entgegenzuwirken, wurde die Fehlercodierung von einer *eight-to-forteen*-Modulation (siehe 3.1.5) auf eine *eight-to-sixteen*-Modulation erhöht. [15] Die genaue Zuordnung bei der Video-DVD ist urheberrechtlich geschützt und wie oben erwähnt nur mit Bezahlung und Unterschrift einer Geheimhaltungsklausel einsehbar.

Reed-Solomon-Product-Code Der RSPC ist eine Weiterentwicklung des RS-Codes, der speziell für die DVD entwickelt wurde. [15] Auch dieser Code ist urheberrechtlich geschützt.

3.3 Warenkennzeichnung

Die Kennzeichnung von Waren durch Strichcodes und der zugrundeliegenden Nummer ist im alltäglichen Leben, insbesondere beim Einkaufen, von besonderer Wichtigkeit. Früher mussten die Verkäufer alle Preise im Kopf haben und diese mittels Kopfrechnung, Zettel oder einer Kassa aufsummieren. Heutzutage haben alle Produkte eine Artikelnummer, die nicht nur im Geschäft einzigartig ist, sondern global eindeutig ist. Damit dies möglich wird, ist eine 13- bis 14-stellige Zahl zur Kennzeichnung erforderlich. Die Zahl muss nicht von Hand eingegeben werden, sondern wird als Strichcode dargestellt, der mit einem Lesegerät von einem Computer eingelesen werden kann.

Der Frage, warum die meisten Strichcodes lediglich über ein Prüfbit verfügen und daher nur einen Fehler erkennen können, möchte ich vorweg beantworten. In den Situationen des Einkaufs ist es kein Problem, den Code ein zweites Mal einzuscannen, wenn er nicht oder falsch eingelesen wurde. Die Fehlerbeseitigung hingegen erfordert mehr als nur eine zusätzliche Ziffer und damit mehr Platz auf der Verpackung.

[5] Stereo bis Dolby Surround 7.1 sind damit möglich.

3.3.1 GTIN (ehem. EAN)

Abb. 3.5: GTIN und Strichcode

Die GTIN (engl. *Global Trade Item Number*) ist eine globale Identifikationsnummer von Waren und Paketen. Sie ist der Nachfolger der EAN[6], die 2009 abgelöst wurde. Der Code wird üblicherweise als Strichcode dargestellt (siehe Abb. 3.5)

Aufbau Durch die stetige Zunahme an Warenobjekten ist die GTIN von acht auf nunmehr 14 Stellen erhöht worden. Die Abwärtskompatibilität wird erhalten, indem Nullen vorangestellt werden.

Die erste Ziffer ist eine Füllziffer, die zweiten zwei sind Ländernummern, die nächsten fünf kennzeichnen den Hersteller, die darauf folgenden fünf stehen für den Artikel und die letzte Ziffer ist die Prüfziffer. Im Folgenden wird die Prüfziffer allgemein, dann anhand eines Beispiels berechnet.

z_{13}	z_{12}	z_{11}	z_{10}	z_9	z_8	z_7	z_6	z_5	z_4	z_3	z_2	z_1	z_P
0	4	0	1	3	7	5	2	0	1	9	1	0	3
	Land		Hersteller					Artikel					Prüfziffer

Tabelle 3.3: GTIN

Prüfziffernberechnung Die Prüfziffer wird berechnet, indem die Ziffern abwechselnd mit 3 und 1 multipliziert werden und anschließend die Produkte addiert werden. Die Differenz zur nächstgrößeren Zahl, die durch 10 teilbar ist, ist die Prüfziffer. Formell:

$$z_P = 10 - \left(\sum_{i \text{ ungerade}} 3z_i + \sum_{i \text{ gerade}} z_i \right) \mod 10. \tag{3.3}$$

[6] EAN, engl. *European Article Number*

Die Prüfziffer der Beispielzahl in Tabelle 3.3 wird in Tabelle 3.4 berechnet. Die Minimaldistanz des GTIN-Code ist 2, da sich bei einem Fehler das Prüfbit automatisch mitverändert. Insbesondere gibt es bei den ungeraden Zahlen, die mit 3 multiplizierten werden, keine Wiederholung der Einerstelle (siehe (3.5)).

$$(1,2,3,4,5,6,7,8,9,10)\cdot 3 \mod 10 = (3,6,9,2,5,8,1,4,7,0), \tag{3.4}$$

$$\tag{3.5}$$

Der GTIN-Code ist also in der Lage, einen Fehler zu erkennen.

Man beachte, dass eine Vertauschung von zwei Ziffern an geraden bzw. ungeraden Stellen nicht erkannt wird.

Abschließend ist der Python-Quellcode, mit dem eine Prüfziffer berechnet werden kann, angegeben.

```
#!/bin/python
# coding=latin-1
# Python GTIN-Prüfnummerngenerierer

print "Python GTIN-Prüfnummerngenerierscript"

# 1. Überprüfe Eingabe
while True:
    gtin=raw_input("GTIN-Nummer (ohne Prüfziffer): ")
    if gtin.isdigit():
            break
    print "Fehlerhafte Eingabe"

# 2. Prüfziffer berechnen
num = 0
for i in range(len(gtin)):
    if i % 2 == 0: # für gerade Positionen
        num += 1*int(gtin[-i]) # -i heißt rückwärts zählend
    if i % 2 == 1: # für ungerade Positionen
        num += 3*int(gtin[-i])
pz = (10 - num) % 10

# 3. Ausgabe
gtin_pz = gtin + str(pz)

print
print "GTIN Prüfziffer lautet %s" %(pz)
```

Ziffer	0	4	0	1	3	7	5	2	0	1	9	1	0	z_P
Multiplikator	3	1	3	1	3	1	3	1	3	1	3	1	3	z_P
Produkt/Summe	0	4	0	1	9	7	15	2	0	1	27	1	0	67
$z_P = 3$	$z_P = 10 - 67 \mod 10$													

Tabelle 3.4: Berechnung der GTIN-Prüfziffer

```
print "GTIN lautet: %s" %(gtin_pz)
```

3.3.2 Banknoten

Abb. 3.6: 20-Euro-Banknote mit markierter Seriennummer

Gerade beim Geld ist es wichtig, sichergehen zu können, dass es sich nicht um Fälschungen handelt. Dazu gibt es viele ausgefeilte optische und haptische Sicherheitsmerkmale. Einen gewissen Beitrag zur Sicherheit leistet auch die Mathematik, durch die Seriennummer, die jeder Geldschein auf der Rückseite aufweist (siehe Abb. 3.6). [17] Ein Beispiel ist in Tabelle 3.5 zu finden.

b_1	b_2	z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8	z_9	z_P
R	B	2	0	3	8	1	6	6	7	4	4

Tabelle 3.5: Beispiel einer Eurobanknotenseriennummer

Aufbau 2013 wurde eine zweite Serie von Euro-Banknoten in Druck gegeben. Diese wird im Folgenden behandelt. Die Seriennummer besteht aus zwei Buchstaben und zwölf darauf folgenden Ziffern. Der erste Buchstabe

zeigt die Druckerei an[7], der zweite „dient zur Kennzeichnung der Banknote innerhalb der Ausgaben der Druckerei“ [17]. Danach befindet sich eine 10-stellige Seriennummer, wobei die letzte Ziffer die Prüfziffer ist.

Prüfziffernberechnung Um die Prüfziffer berechnen zu können, müssen als erster Schritt die Buchstaben einer Zahl zugeordnet werden. Dies geschieht nach dem einfachen Schema $A = 1, B = 2, \ldots Z = 26$. In Tabelle 3.5 ergeben sich also die Zahlen 18 und 2. Danach wird die Quersumme über alle Ziffern gebildet und zur Summe der Buchstabenpositionen addiert. Die sich ergebende Zahl wird modulo 9 berechnet und von 7 abgezogen. Das Ergebnis ist die Prüfziffer, formell:

$$z_P = 7 - (\sum_{i=1}^{2} f(b_i) + \sum_{i=1}^{9} z_i) \mod 9,$$

wobei $f(b_i)$ Quersumme der oben genannte Zuordnung der Buchstaben zu Zahlen ist.[8] Tabelle 3.6 zeigt die Berechnung des Beispiels.

Die Minimaldistanz bei der Euro-Banknoten-Kennzeichnung ist 2, da jede Veränderung der Seriennummer eine Änderung der Prüfziffer nach sich ziehen würde. Damit kann die Kennzeichnung einen Fehler erkennen. Fehler bei den Buchstaben werden zum einen erkannt, wenn der Buchstabe nicht zugeordnet ist, zum anderen wenn die Kontrollnummer falsch ist.

Weitere Sicherheitsmaßnahme Wenn die 10-stellige Zahl modulo 9 berechnet wird, ergibt sich eine weitere Kontrollnummer, die für das Buchstabenpaar eindeutig vorgegeben ist. Zum Beispiel ist der Druckerei und Ausgabe mit den Buchstaben „RB“ die Kontrollnummer 5 zugeordnet.

$$2038166744 \mod 9 = 5 \tag{3.6}$$

Eine Möglichkeit, die Prüfziffer mittels Python zu berechnen, lautet wie folgt:

```
#!/bin/python
# coding=latin-1
# Python Eurobanknoten-Prüfnummerngenerierer
import string

print "Python Eurobanknoten-Prüfnummerngenerierscript"
```

[7] Es sind nicht alle 26 Buchstaben vergeben.

[8] Für Buchstabe R gilt: $f(R) = (1 + 8) \neq 18$!

```
# 1. Überprüfe Eingabe
while True:
    sn=raw_input("Seriennummer (ohne Prüfziffer): ")

    # Überprüfe Syntax
    # Sind 1./2.Symbole Buchstaben?
    if sn[0:2].isalpha():
    # Sind die restlichen Symbole Ziffern?
        if sn[2:].isdigit():
    # Sind genau 11 Symbole vorhanden?
            if len(sn) == 11:
                break
    print "Fehlerhafte Eingabe"

print "Generiere Prüfnummer für %s" %(sn)

# 2. Prüfziffer berechnen
num = 0
for i in range(len(sn)):
    if sn[i].isalpha():
    # Zahl des Buchstaben (A=1, B=2, ...)
    # +1 da Funktion von 0 wegzählt
        numzahl= string.lowercase.index(sn[i].lower())+1
        quersumme = sum([int(j) for j in str(numzahl)])
        num += quersumme
    if sn[i].isdigit():
        num += int(sn[i])
pz = (7 - num) % 9

# 3. Ausgabe
sn_pz = sn + str(pz)
print "Prüfziffer der Seriennummer lautet %s" %(pz)
print "Seriennummer lautet: %s" %(sn_pz)
```

R	B	2	0	3	8	1	6	6	7	4	z_P
1+8	2	2	0	3	8	1	6	6	7	4	z_P
Summe: $7 - (48 \mod 9) = 4$											4

Tabelle 3.6: Berechnung der Eurobanknoten-Prüfziffer

3.3.3 IBAN

IBAN (engl. *International Bank Account Number*) ist die international standardisierte Notation für Bankkontonummern.[13]

Aufbau Der IBAN-Code besteht aus maximal 34 Zeichen, wovon die ersten zwei den Ländercode darstellen (z. B. AT für Österreich), die zweiten zwei die Prüfsumme, und die verbleibenden 30 Stellen für Kontodaten zur Verfügung stehen. In Österreich bestehen die Kontodaten aus 15 Stellen, wovon fünf Stellen der Bankleitzahl und elf Stellen der Kontonummer zufallen. Kontonummern die kürzer als zehn Stellen sind, werden mit vorangestellten Nullen (zwischen Bankleitzahl und Kontonummer) gefüllt. Ein Beispiel zeigt Tabelle 3.7.

z_1 z_2	z_3 z_4	z_5 z_6 z_7 z_8 z_9	z_{10} z_{11} z_{12} z_{13} z_{14} z_{15} z_{16} z_{17} z_{18} z_{19} z_{20}
A T	X Y	1 2 3 4 5	0 0 0 1 2 3 4 5 6 7 8
Land	Prüf-ziffern	Bankleitzahl	Kontonummer

Tabelle 3.7: IBAN

Prüfziffernberechnung Die Prüfziffern XY ergeben sich aus der Berechnung

$$XY = 98 - (z \mod 97),$$

wobei z die Kontoidentifikation (BLZ und Kontonummer) mit vorangestelltem Ländercode und 00 (als temporäre Prüfziffer) ist. Der Ländercode ist mit der Vorschrift $A = 10, B = 11, C = 12, \ldots, Z = 36$ codiert. Für Österreich (AT) ist der Code also 1029. Im Beispiel der Tabelle 3.7 gilt also:

$$z = \underbrace{12345}_{\text{BLZ}}\underbrace{00012345678}_{\text{Kontonummer}}\underbrace{1029}_{\text{AT}}\underbrace{00}_{\text{Prüfziffern}} \tag{3.7}$$

$$XY = 98 - (123450012345678102900 \mod 97) = 16 \tag{3.8}$$

Der IBAN lautet also: AT16123500012345678.

Die Prüfziffern gewähren, dass eine Änderung einer Ziffer der IBANs eine Änderung der Prüfziffer bewirkt. Damit ist die Minimaldistanz der IBAN 2, was nach dem Satz im Kapitel 2.3.4 dazu führt, dass ein Fehler erkannt werden kann.

Folgend ist eine Möglichkeit angegeben, die Berechnung mittels Python zu realisieren:

```
#!/bin/python
# coding=latin-1
# Python IBAN-Prüfnummerngenerierer
import string

# Ländercode lt ISO 3166-1 alpha-2
land= ["AD","AE","AF","AG","AI","AL","AM","AO","AQ","AR",
        "AS","AT","AU","AW","AX","AZ","BA","BB","BD","BE",
        "BF","BG","BH","BI","BJ","BL","BM","BN","BO","BQ",
        "BR","BS","BT","BV","BW","BY","BZ","CA","CC","CD",
        "CF","CG","CH","CI","CK","CL","CM","CN","CO","CR",
        "CU","CV","CW","CX","CY","CZ","DE","DJ","DK","DM",
        "DO","DZ","EC","EE","EG","EH","ER","ES","ET","FI",
        "FJ","FK","FM","FO","FR","GA","GB","GD","GE","GF",
        "GG","GH","GI","GL","GM","GN","GP","GQ","GR","GS",
        "GT","GU","GW","GY","HK","HM","HN","HR","HT","HU",
        "ID","IE","IL","IM","IN","IO","IQ","IR","IS","IT",
        "JE","JM","JO","JP","KE","KG","KH","KI","KM","KN",
        "KP","KR","KW","KY","KZ","LA","LB","LC","LI","LK",
        "LR","LS","LT","LU","LV","LY","MA","MC","MD","ME",
        "MF","MG","MH","MK","ML","MM","MN","MO","MP","MQ",
        "MR","MS","MT","MU","MV","MW","MX","MY","MZ","NA",
        "NC","NE","NF","NG","NI","NL","NO","NP","NR","NU",
        "NZ","OM","PA","PE","PF","PG","PH","PK","PL","PM",
        "PN","PR","PS","PT","PW","PY","QA","RE","RO","RS",
        "RU","RW","SA","SB","SC","SD","SE","SG","SH","SI",
        "SJ","SK","SL","SM","SN","SO","SR","SS","ST","SV",
        "SX","SY","SZ","TC","TD","TF","TG","TH","TJ","TK",
        "TL","TM","TN","TO","TR","TT","TV","TW","TZ","UA",
        "UG","UM","US","UY","UZ","VA","VC","VE","VG","VI",
        "VN","VU","WF","WS","YE","YT","ZA","ZM","ZW"]


print "Python IBAN-Prüfnummerngenerierscript"

# 1. Überprüfe Eingabe
# 1.1. Bankleitzahl
while True:
    blz=raw_input("Bankleitzahl (BLZ): ")
    if blz.isdigit():
```

```
        if len(blz) == 5:
            break
    print "Fehlerhafte Eingabe"

# 1.2. Länge der Kontonummer und fülle mit Nuller
while True:
    kn=raw_input("Kontonummer: ")
    if kn.isdigit():
        if len(kn) <= 11:
                kn = kn.zfill(11)
                break
    print "Fehlerhafte Eingabe"

# 1.3. Ländercode
while True:
    ld=raw_input("Ländercode (2 Buchstaben): ")
    if len(ld) == 2:
        ld = ld.upper()
        if ld in land:
            ld_num = ''
            for i in range(len(ld)):
                a = ld[i].lower()
                ld_num += str(string.lowercase.index(a)+10)
            break
    print "Fehlerhafte Eingabe"

# 2. Prüfziffer berechnen
# XY= 98-(num mod 97)
num = int(blz+kn+str(ld_num)+'00')
num = num % 97
num = 98-num
XY = str(num).zfill(2)

# 3. Ausgabe
IBAN = ld + str(XY) + str(blz) + str(kn)

print
print "Prüfziffer lautet: %s" %(XY)
print "IBAN lautet: %s" %(IBAN)
```

3.3.4 ISBN-10

Abb. 3.7: ISBN und Strichcode

Die Internationale Standardbuchnummer (kurz ISBN) wird zur eindeutigen Identifikation von Büchern und anderen Veröffentlichungen genutzt.

z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8	z_9	z_P
0	1	4	1	0	3	6	1	4	1
Land	Verlag				Artikelnr.				Prüfziffer

Tabelle 3.8: ISBN-10

Aufbau Die ISBN-10 besteht aus zehn Zeichen, wovon das erste die Ländererkennung ist, das zweite bis fünfte Zeichen die Verlagsnummer, das sechste bis neunte Zeichen die Artikelnummer sind und das zehnte die Prüfziffer ist. Mögliche Zeichen sind 0–9, wobei als Prüfziffer ebenfalls 10 zugelassen ist. Um die Einstelligkeit zu bewahren, wird 10 gemäß des römischen Alphabets als X geschrieben. Ähnlich wie beim GTIN-Code ist es möglich, dass die Herstellerzeichen zugunsten der möglichen Artikelzahl kürzer gewählt werden. Ein Beispiel einer ISBN-10 Nummer ist in Tabelle 3.8 zu sehen.

Prüfziffernberechnung Die Prüfziffer z_P ergibt sich aus der Multiplikation der Ziffer mit seiner Position (von rechts nach links gezählt). Die Quersumme der sich ergebenden Zahlen wird modulo 11 berechnet.

Formell wird die Prüfziffer wie folgt berechnet:

$$z_P = \sum_{i=1}^{9} (10 - i) \cdot z_i \mod 11.$$

Tabelle 3.9 zeigt die Berechnung der ISBN-10-Prüfziffer aus Tabelle 3.8. Die

Ziffer	0	1	4	1	0	3	6	1	4	z_P
Multiplikator	9	8	7	6	5	4	3	2	1	z_P
Produkt/Summe	0	8	28	6	0	12	18	2	4	78 mod 11
	$z_P = 1 = 78 \mod 11$									

Tabelle 3.9: Berechnung der ISBN-Prüfziffer

ISBN-10 ist in der Lage, einen Fehler zu erkennen da die Minimaldistanz auch hier 2 ist. Hinzu kommt, dass die ISBN-10 Vertauschungsfehler erkennen kann, da jede Ziffer mit einem anderen Vorfaktor mulipliziert wird. Eine Art, die Prüfziffer mittels Python zu berechnen, folgt.

```
#!/bin/python
# coding=latin-1
# Python ISBN-Prüfnummerngenerierer

print "Python ISBN-Prüfnummerngenerierscript"

# 1. Überprüfe Eingabe
while True:
    isbn=raw_input("ISBN-Nummer (ohne Prüfziffer): ")
    if isbn.isdigit():
        if len(isbn) == 9:
            break
    print "Fehlerhafte Eingabe"

# 2. Prüfziffer berechnen
# z_P = sum(i=1-9) (10-i) z_i mod 11
num = 0
for i in range(9):
    num += int(isbn[i])*(9-i)

num = num %11

if num == 10:
    num = 'X'

# 3. Ausgabe
isbn_pz = isbn + str(num)

print
```

```
print "ISBN Prüfziffer lautet %s" %(num)
print "ISBN lautet: %s" %(isbn_pz)
```

ISBN-13 2007 löste die ISBN-13 die ISBN-10 ab. Die ISBN-13 ist nichts Anderes als ein GTIN-Code mit 13 Stellen, wovon die ersten Stellen entweder 978 oder 979-1 bis 979-9 sind. [9, S. 333]. Leider verliert die ISBN-13 damit die Eigenschaft, Vertauschungsfehler zu erkennen. Die Prüfziffer ist ebenfalls eine andere.

3.3.5 IMEI-Nummer

Die IMEI (engl. *International Mobile Equipment Identity*) ist, ähnlich wie die MAC-Adresse bei Netzwerkgeräten, die eindeutig zuordenbare Nummer eines Mobiltelefons.

Aufbau Die IMEI-Zahl besteht aus 15 Ziffern, wovon die ersten acht den *Type Allocation Code* (TAC) bilden, die nächsten sechs die Seriennummer (SNR) und die letzte Ziffer die *Check Digit* (CD), also die Prüfziffer ist. Tabelle 3.10 zeigt das Schema der Aufteilung.

z_1 z_2 z_3 z_4 z_5 z_6 z_7 z_8	z_9 z_{10} z_{11} z_{12} z_{13} z_{14}	z_P
3 5 9 7 7 8 0 4	1 2 3 4 5 6	X
TAC	SNR	CD

Tabelle 3.10: IMEI

Prüfziffernberechnung Die Prüfziffer z_P ergibt sich aus der Berechnung

$$z_P = \sum_{i \text{ ungerade}} z_i + \sum_{i \text{ gerade}} 2z_i \mod 10.$$

Tabelle 3.11 zeigt die Berechnung der IMEI-Nummer aus Tabelle 3.10.

Ziffer	3	5	9	7	7	8	0	4	1	2	3	4	5	6	z_P
Multiplikator	1	2	1	2	1	2	1	2	1	2	1	2	1	2	z_P
Produkt/Summe	3	10	9	14	7	16	0	8	1	4	3	8	5	12	100
	$z_P = 0 = 100 \mod 10$														

Tabelle 3.11: Berechnung der IMEI-Prüfziffer

Der IMEI-Code erlaubt mit Minimaldistanz 2 einen Fehler zu erkennen. Nachfolgend ein Skript zur Prüfziffernberechnung in Python.

```
#!/bin/python
# coding=latin-1
# Python IMEI-Prüfnummerngenerierer

print "Python IMEI-Prüfnummerngenerierscript"

# 1. Überprüfe Eingabe
while True:
    imei=raw_input("IMEI-Nummer (ohne Prüfziffer): ")
    if imei.isdigit():
        if len(imei) == 14:
            break
    print "Fehlerhafte Eingabe"

# 2. Prüfziffer berechnen
# z_P = sum(i ungerade)z_i + sum(i gerade) 2z_i mod 10
num = 0
for i in range(14):
    print i
    # i+1: python zählt von 0 weg
    if (i+1) % 2 == 1: # ungerade
        num += int(imei[i])
        print "ungerade", imei[i]
    if (i+1) % 2 == 0: # gerade
        num += int(imei[i])*2
        print "gerade", int(imei[i])*2
num = num % 10

# 3. Ausgabe
imei_pz = imei + str(num)

print
print "IMEI Prüfziffer lautet %s" %(num)
print "IMEI lautet: %s" %(imei_pz)
```

3.3.6 Eierkennzeichnung

Seit 2004 sind Eier in der EU kennzeichnungspflichtig. Die Kennzeichnung erlaubt dem Konsumenten, genau festzustellen, woher die Eier stammen. [16] Ein Prüfbit gibt es in diesem Fall nicht.

Abb. 3.8: Eierkennzeichnung, ©Thomas R. Schwarz

2	AT	4755278	2
Haltung	Land	Betrieb	Freiwillige Angaben

Tabelle 3.12: Eierkennzeichnung – Beispiel

Aufbau Die Eierkennzeichnung besteht aus drei Blöcken. Tabelle 3.12 zeigt ein Beispiel. Der erste Block ist eine Ziffer von 0–3, die über die Haltungsart Auskunft gibt. Tabelle 3.13 zeigt die möglichen Werte.

0	Ökologische Erzeugung
1	Freilandhaltung
2	Bodenhaltung
3	Käfighaltung (in Österreich seit 2009 verboten) bzw. Kleingruppenhaltung

Tabelle 3.13: Eier-Code – Haltungsart

Nach der Haltungsziffer folgt der zweistellige Ländercode (siehe Tabelle 3.14). Danach identifiziert ein fünfstelliger Code den Betrieb (LFBIS-Nummer[9]).

AT	Österreich	FI	Finnland	LU	Luxemburg
BE	Belgien	FR	Frankreich	NL	Niederlande
DE	Deutschland	GR	Griechenland	PT	Portugal
DK	Dänemark	IE	Irland	SE	Schweden
ES	Spanien	IT	Italien	UK	Vereinigtes Königreich

Tabelle 3.14: Eier-Codes der Registrierungsmitgliedstaaten

[9] `http://www.statistik.at/ovis/pdf/index.html` (Stand: 5.10.2017)

Dabei steht in Österreich die erste Ziffer für das Bundesland (siehe Tabelle 3.15).

1	Burgenland		6	Steiermark
2	Kärnten		7	Tirol
3	Niederösterreich		8	Vorarlberg
4	Oberösterreich		9	Wien
5	Salzburg			

Tabelle 3.15: Eier-Code – Bundesland

Danach folgen freiwillige Angaben. Diese sind etwa:

- Letztes empfohlenes Verkaufsdatum
- Legedatum
- Futtermittel und -menge der Hennen

Kapitel 4

Anhang

4.1 Anhang A: Algebraische Strukturen

Dieser Teil dient als schlagwortartige Nachschlagemöglichkeit.

Halbgruppe $(H, \circ)$

- Assoziativgesetz: $x \circ (y \circ z) = (x \circ y) \circ z, \ \forall x, y, z \in H$

Gruppe $(G, \circ)$

- $(G, \circ)$ ist eine Halbgruppe
- neutrales Element e: $e \circ x = x, \ \forall x \in G$
- inverses Element x^{-1}: $x \circ x^{-1} = e, \ \forall x \in G$

Eine Gruppe heißt *kommutativ* oder *abelsch*, wenn das *Kommutativgesetz* gilt:

$$x \circ y = y \circ x, \ \forall x, y \in G$$

Ring $(R, +, \cdot)$

- $(R, +)$ ist eine abelsche Gruppe
- $(R, \cdot)$ ist eine Halbgruppe
- Distributivgesetze: $x \cdot \ (y + z) = x \cdot \ y + x \cdot \ z$,
  $(x + y) \cdot \ z = x \cdot \ z + y \cdot \ z, \forall x, y, z \in \ R$

Ist die Multiplikation kommutativ, so handelt es sich um einen *kommutativen Ring*. Existiert ein neutrales Element 1 gegenüber $\cdot$, so heißt der Ring, *Ring mit Einselement*. Ist ein kommutativer Ring R mit Einselement $1 \neq 0$ nullteilerfrei, das heißt es gibt kein Element $x, y \neq 0$ mit $x \cdot y = 0$, so nennt man diesen *Integritätsring*.

Körper $(K, +, \cdot)$

- $(K, +, \cdot)$ ist ein kommutativer Ring mit Einselement
- $0 \neq 1$
- $(K \backslash \{0\}, \cdot)$ ist eine abelsche Gruppe

Vektorraum $\mathbf{(V, +, \cdot)}$ über dem Körper K

- $(V, +)$ ist eine abelsche Gruppe
- $\lambda \cdot (x + y) = \lambda \cdot x + \lambda \cdot y, \ \forall x, y \in V, \lambda \in K$
- $(\lambda + \mu) \cdot x = \lambda \cdot x + \lambda \cdot y, \ \forall x \in V, \lambda, \mu \in K$
- $(\lambda \cdot \mu) \cdot x = \lambda \cdot (\mu \cdot x), \ \forall x \in V, \lambda, \mu \in K$
- $1 \cdot x = x \cdot 1 = x, \ \forall x \in V, 1 \in K$

Modulo-Funktion

$$x \mod m = a \iff \exists k \in \mathbb{N} : a + k \cdot m = x, \ x \in \mathbb{Z}, a \in \mathbb{Z}_m$$

Restklassen

$$\mathbb{Z}_m = \{x | x = n \mod m, n \in \mathbb{N}\}$$

4.2 Anhang B: Glossar

Im Folgenden sind einige Abkürzungen aufgelistet, die in dieser Arbeit oft verwendet wurden:

CD Compact Disc

DVD Digital Video/Versatile Disc

EAN European Article Number

EF Eight to Fourteen

GTIN Global Trade Item Number

IBAN International Bank Account Number

IMEI International Mobile Equipment Identity

ISBN International Standard Book Number

RS Reed Solomon

4.3 Anhang C: Codierungstafeln

	Buchstabiertafel [11]	Braille	Morse	Unicode [12]
A	Anton	⠁	·_	U+0041
Ä	Ärger			U+00C4
B	Berta	⠃	_···	U+0042
C	Cäsar	⠉	_·_·	U+0043
D	Dora	⠙	_··	U+0044
E	Emil	⠑	·	U+0045
F	Friedrich	⠋	··_·	U+0046
G	Gustav	⠛	__·	U+0047
H	Heinrich	⠓	····	U+0048
I	Ida	⠊	··	U+0049
J	Julius	⠚	·___	U+004A
K	Konrad	⠅	_·_	U+004B
L	Ludwig	⠇	·_··	U+004C
M	Martha	⠍	__	U+004D
N	Nordpol	⠝	_·	U+004E
O	Otto	⠕	___	U+004F
Ö	Österreich			U+00D6
P	Paula	⠏	·__·	U+0050
Q	Quelle	⠟	__·_	U+0051
R	Richard	⠗	·_·	U+0052
S	Siegfried	⠎	···	U+0053
Sch	Schule			
T	Theodor	⠞	_	U+0054
U	Ulrich	⠥	··_	U+0055
Ü	Übel			U+00DC
V	Viktor	⠧	···_	U+0056
W	Wilhelm	⠺	·__	U+0057
X	Xaver	⠭	_··_	U+0058
Y	Ypsilon	⠽	_·__	U+0059
Z	Zeppelin	⠵	__··	U+005A

Tabelle 4.1: Verschiedene Codierungen des Alphabets

HEX	BIN	DEC
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
a	1010	10
b	1011	11
c	1100	12
d	1101	13
e	1110	14
f	1111	15

Tabelle 4.2: Umrechnungstafel

Index

Literaturverzeichnis

[1] T. Borys *Codierung und Kryptologie*,
1. Auflage (Vieweg+Teubner Verlag, Wiesbaden 2011)

[2] W. Dankmeier *Grundkurs Codierung – Verschlüsselung, Kompression, Fehlerbeseitigung*,
3. Auflage (Vieweg Verlag, Wiesbaden 2006)

[3] G. Dorfer und G. Eigenthaler: *Algebra für Lehramt*,
(TU Wien, SS 2013)

[4] G. Dorfer: *Fehlerkorrigierende Codes – Vorlesungsskript*,
(TU Wien, SS 2013)

[5] D. Dorninger: *Algebraische Codierungstheorie und Compact Discs*,
1. Auflage (Birkhäuser Verlag, Basel 1996)

[6] D. Dorninger: *Codierung und Chiffrierung*,
Heft 144 (Schriftreihe zur Lehrerbildung im berufsbildenden Schulwesen, Pädagogisches Institut des Bundes in Wien, 1991)

[7] D. Dorninger, W. Müller: *Allgemeine Algebra und Anwendungen*,
(Teubner Verlag, Stuttgart 1984)

[8] D. Dorninger, C. Fabianek, A. Traxler: *Algebraische Methoden in den Computerwissenschaften*,
(TU Wien, SS 2000)

[9] D. W. Hoffmann: *Einführung in die Informations- und Codierungstheorie*,
(Springer Verlag, Berlin 2014)

[10] Johannes Bloemer: *Algorithmische Codierungstheorie – Skript zur Vorlesung*,
(Uni Paderborn, SS 2007)

[11] Buchstabiertafel – Wikipedia:
https://de.wikipedia.org/wiki/Buchstabiertafel (15.09.2017)

[12] Unicode (latin-1):
http://www.unicode.org/charts/PDF/U0000.pdf (20.09.2017)

[13] IBAN – Wikipedia:
https://de.wikipedia.org/wiki/IBAN (22.09.2017)

[14] Prüfziffernberechnung GTIN: https://www.gs1.at/fileadmin/user_upload/Algorithmus_GS1_Pruefziffernberechnung.pdf (22.09.2017)

[15] DVD – Digital Video/Versatile Disc:
http://www.elektronik-kompendium.de/sites/com/0507171.htm (06.10.2017)

[16] Richtlinie 2002/4/EG der Kommission (Amtsblatt der Europäischen Gesellschaften):
http://eur-lex.europa.eu/legal-content/DE/TXT/PDF/?uri=CELEX:32002L0004&from=DE (06.10.2017)

[17] Bedeutung der EURO Kontrollnummer:
http://www.geldschein.at/euro-banknoten/euro_seriennummer.html (06.10.2017)

[18] DVD FLLC – Licence:
http://www.dvdfllc.co.jp/license/l_howto.html (20.10.2017)

[19] DVD-Video – Wikipedia:
https://en.wikipedia.org/wiki/DVD-Video (20.10.2017)