3.4 Codierung mit 0 und 1 ist überall

An dieser Stelle ist es vielleicht hilfreich, wenn wir uns die überragende Bedeutung von 0 und 1 vor Augen führen. Computer arbeiten mit Strom und es kommt darauf an, ob an bestimmter Stelle Strom fließt oder nicht. Irgendeine Unterscheidung von viel und wenig Strom findet nicht statt. Darum gibt es nur zwei Zustände, repräsentiert durch 0 und 1. Ein „Platz“ für 0 oder 1 ist ein *Bit*. Es ist die kleinste Information tragende Einheit, das *Informationsatom*. Dazu können Sie mehr in Kapitel 8 lesen.

Bei allen *digitalen* technischen Geräten kann man sich vorstellen, dass sie lange Ketten aus 0 und 1 verarbeiten. Digitale Bilder enthalten keine Farbpigmente, digitale Musik enthält keine Töne, beim digitalen Fernsehen kommen Ströme von 0 und 1 bei Ihnen zu Hause an. Das Internet ist ein erdumspannender, nicht endender Strom von 0-1-Ketten. Codierung ist überall, denn schließlich sehen Sie ja doch bunte Bilder im Film, hören Musik von CDs, arbeiten mit Ihrem Computer, ohne dass Sie Nullen und Einsen zu Gesicht bekommen.

3.4.1 Fehlerkorrigierende Codes

Vielleicht haben Sie sich schon einmal gefragt, warum die oft gespielten CDs so wenig kratzen und knacken, auch wenn die Oberfläche schon nicht mehr ganz unversehrt ist. Warum kommen die Texte und Bilder, die Sie mit Ihren Freunden und Kollegen austauschen, genauso an, wie sie abgeschickt wurden? Passieren denn beim Übertragen gar keine Fehler?

Doch, es passiert gar nicht selten, dass ein Bit falsch ankommt. Aber die Mathematiker haben Codes entwickelt, bei denen etliche Fehler selbsttätig korrigiert werden können. Das leuchtet zunächst gar nicht ein: Wie soll es denn noch richtig werden, wenn etwas beim Empfänger falsch ankommt?

Hamming-Code

Der erste fehlerkorrigierende Code wurde von Richard Hamming gleich zu Beginn des Computerzeitalters 1948 entwickelt. Er zeigt das Grundprinzip so deutlich, dass wir ihn uns genauer ansehen.

Ein wesentlicher Begriff der Codierungstheorie ist der folgende:

Die **Parität einer Bitfolge** ist 0, wenn die Anzahl der 1 in der Folge gerade ist.
Die Parität einer Bitfolge ist 1, wenn die Anzahl der 1 in der Folge ungerade ist.

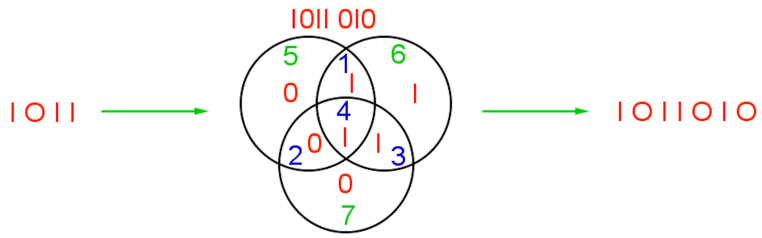

Abb. 3.9: Hamming-Code, ∼-Algorithmus

Beispiel: Die Parität von 11101011 ist 0, die Parität von 11101010 ist 1.

Übrigens schreibt man in der Handschrift gern die 1 in einer Bitfolge als einfachen Strich. Dann kann man gleich Bitfolgen von Dezimalzahlen unterscheiden.

Zu je vier eigentlich zu sendenden Bits der Nachricht werden beim Hamming-Algorithmus drei *Korrekturbits* berechnet und angehängt. Abb. 3.9 verdeutlicht das Vorgehen, wenn man „von Hand“ arbeitet:

1. Schreibe die Nachricht in die blauen Felder 1, 2, 3, 4.
2. Schreibe in die grünen die Parität der im zugehörigen Kreis stehenden Bits.
3. Hänge die Bits der Felder 5, 6, 7 an die Nachricht an.
4. Der Empfänger trägt die sieben Bits in die Felder ein und prüft, ob alles richtig ist.

Im Beispiel in Abb. 3.9 ist dargestellt, dass statt der Nachricht 1011 die Bitfolge 1011010 gesendet wird. Nur vier dieser sieben Bits tragen die eigentliche Information. Daher sagt man auch, der Hamming-Code habe einen *Informationsgehalt* von vier Siebenteln.

Dieser zusätzliche Aufwand lohnt sich aber, wie wir uns im Folgenden klarmachen.

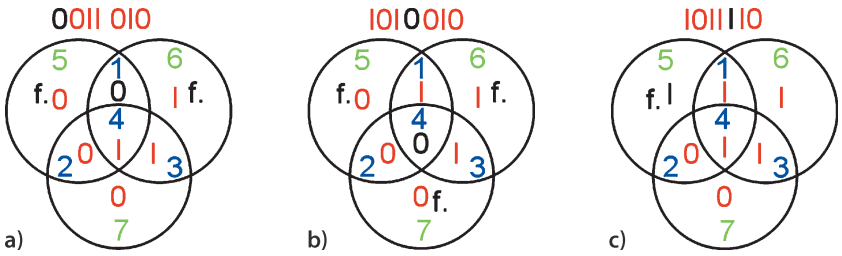

Abb. 3.10: Fehler in der Sendung

Wir betrachten die Fälle, bei denen beim Senden der Nachricht nur ein einziger Fehler auftritt. Dann gibt es drei Fehlertypen.

– **Typ 1**
In Abb. 3.10a ist *eins der ersten drei Bits falsch* übermittelt, hier schwarz in Feld 1 dargestellt. 0011010 wurde empfangen. Dann zeigt aber Bit 5 etwas Falsches an, denn die Felder 1, 2 und 4 haben nun nur eine 1, darum müsste in Feld 5 eine 1 stehen. Ebenso passt der Eintrag in Feld 6 nicht mehr. Aber in Feld 7 steht weiterhin das Richtige. Wenn der Empfänger also die Bitfolge prüft, merkt er, dass genau zwei Fehler aufgetreten sind, nämlich in Feld 5 und 6, darum muss, – es durfte ja nur ein Fehler beim Senden auftreten, – Feld 1 falsch sein. Da dort das Bit 0 angekommen ist, hätte es eine 1 sein müssen. Also korrigiert der Empfänger den Fehler und nimmt als Codewort nun 1011010 an. Ebenso können Einzelfehler in Feld 2 oder 3 korrigiert werden.

– **Typ 2**
In Abb. 3.10b ist *das vierte Bit falsch* übermittelt, hier schwarz in Feld 4 dargestellt. 1010010 wurde empfangen. Nun sind ebenfalls die

Felder 5 und 6 falsch, aber auch Feld 7. Hieraus schließt der Empfänger, dass das Bit in Feld 4 falsch angekommen ist. Er berichtigt es und nimmt als Codewort nun 1011010 an.

– **Typ 3**
In Abb. 3.10c ist *eins der Korrekturbits falsch* übermittelt, hier schwarz in Feld 5 dargestellt. 1011110 wurde empfangen. Von diesem Fehler sind die Felder 6 und 7 nicht berührt, ausschließlich Feld 5 zeigt etwas Falsches an. Daraus schließt der Empfänger, dass nur das Bit in Feld 5 selbst falsch angekommen ist. Er berichtigt es und nimmt als Codewort nun 1011010 an. Ebenso geht es bei Einzelfehlern in Feld 6 oder 7.

Wir haben gesehen: Der Hamming-Code kann Einzelfehler immer korrigieren. Wenn also in einer sehr langen Bitfolge in keinem Siebenerblock mehr als ein Übertragungsfehler auftritt, ist am Ende dennoch die ganze Folge vollständig richtig.

Hamming-Abstand

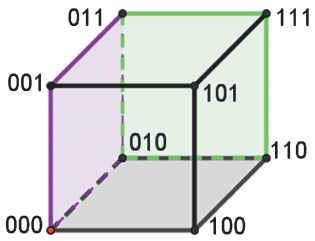

Abb. 3.11: Hamming-Abstand

Eine Verbesserung kann man schon dadurch erreichen, dass man gar nicht alle 0-1-Folgen der betrachteten Länge als Codewörter zulässt.

Bei zwei 0-1-Codewörtern gleicher Länge ist der **Hamming-Abstand** die Anzahl der unterschiedlich besetzten Plätze.

Abb. 3.11 zeigt an den Ecken des Würfels alle 0-1-Codewörter mit drei Stellen. Von einer Ecke zu einer Nachbarecke ändert sich stets nur ein Bit. Benachbarte Ecken haben also den Hamming-Abstand 1. Zwei flächig diagonal gegenüberliegende Ecken haben aber schon Hamming-Abstand 2, raumdiagonal ergibt sich Hamming-Abstand 3.

Wählt man als zulässige Codewörter jetzt nur {000, 110, 101, 011} aus, dann haben je zwei Hamming-Abstand 2. Das bewirkt, dass bei Übertragung eines einzigen falschen Bits sofort ein ungültiges Codewort entsteht und der Fehler gemerkt wird.

Bei der EAN-Codierung aus Abb. 3.5 haben die Codewörter mindestens Hamming-Abstand 2. Konstruiert man einen Code mit Hamming-Abstand mindestens 3, so kann man bei Auftreten eines einzigen Fehlers das *in der Nähe* gelegene gültige Codewort nehmen.

Andere fehlerkorrigierende Codes

Die Informatiker, die in diesem Buch übrigens nicht streng von den Mathematikern unterschieden werden, haben die Möglichkeiten noch erweitert. Manche Codes können wenigstens erkennen, ob in dem Bitpaket genau zwei Fehler aufgetreten sind, wenn auch deren Platz nicht ermittelbar ist. Dann kann die Sendung des Paketes noch einmal angefordert werden. Aber es gibt durchaus noch komfortablere fehlerkorrigierende Codes. Auch der Anwender hat z. B. in Brennprogrammen für CDs noch gewisse Steuerungsmöglichkeiten.