Die Mathematik der Compact Disc

Jack H. van Lint

Jeder verwendet heute ganz selbstverständlich Compact Discs. Warum ist aber die Musikübertragung auf einer CD reiner als auf einer herkömmlichen Schallplatte? Die Antwort lautet, um einen populären Slogan abzuwandeln: There is mathematics inside! Genauer gesagt, ein Zweig der Diskreten Mathematik, nämlich die Theorie der Fehler-korrigierenden Codes. In diesem Artikel soll über die Anwendung solcher Codes auf den Design des Compact-Disc-Audio-Systems berichtet werden, das von Philips Electronics und Sony entwickelt wurde.

Wörter und Codes

Wir werden das folgende leicht verständliche mathematische Konzept verwenden. Betrachten wir zwei n-Tupel $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$ und $\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$, wobei die a_i und b_i aus einer Menge Q, genannt das *Alphabet*, stammen. Der *Hamming-Abstand* von $\boldsymbol{a}$ und $\boldsymbol{b}$ ist definiert durch

$$d(\boldsymbol{a}, \boldsymbol{b}) = \text{ Anzahl der } i \text{ mit } a_i \neq b_i.$$

Wir nennen $\boldsymbol{a}$ und $\boldsymbol{b}$ *Wörter der Länge n* über dem Alphabet Q.

Die Analogie dieses Begriffes „Wörter“ zur gewöhnlichen Sprache ist absichtlich. Nehmen wir an, wir lesen ein Wort (sagen wir, in einem Buch auf Englisch) und bemerken, dass das Wort zwei Druckfehler enthält. Dann heißt das in unserer Terminologie, dass das gedruckte Wort Hamming Abstand 2 zum korrekten Wort hat. Der Grund, warum wir die 2 Druckfehler erkennen, liegt darin, dass in der englischen Sprache nur ein einziges Wort mit so einem kleinen Abstand zum gedruckten (fehlerhaften) Wort existiert. Das ist auch schon das grundlegende Prinzip der Fehler-korrigierenden Codes: Entwirf eine Sprache (genannt *Code*) von Wörtern einer festen Länge über einem Alphabet Q, so dass je zwei Codewörter mindestens Abstand $2e + 1$ haben. Diese Größe $2e + 1$ heißt dann der Minimalabstand des Codes. Offensichtlich ist ein Wort, das höchstens e Fehler enthält, näher zum korrekten Wort als zu jedem anderen Codewort.

Ein einfaches Beispiel

Ein Beispiel solch codierter Information ist jedem von uns geläufig: Es sind die *Strich-Codes*, mit denen heute die meisten Produkte versehen sind. Zunächst wird jedes Produkt mit einer Folge von 12 Zahlen (aus 0 bis 9) indentifiziert. Jede Zahl

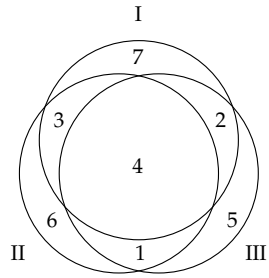
M. Aigner, E. Behrends (Hrsg.), *Alles Mathematik*, DOI 10.1007/978-3-658-09990-9_2

Abbildung 1. Strich-Code und Viking-Aufnahme vom Mars (Jet Propulsion Laboratory of the California Institute of Technology)

wird ersetzt durch ein Codewort mit sieben 0'en und 1'en. Auf dem Produkt werden 0 und 1 durch einen dünnen Strich dargestellt: 0 durch einen weißen Strich und 1 durch einen schwarzen Strich. Zum Beispiel wird 5 codiert durch 0110001, so dass auf dem Produkt ein weißer Strich erscheint, eine Einheit breit, gefolgt von einem schwarzen Strich, zwei Einheiten breit, einem weißen Strich, drei Einheiten breit, und einem abschließenden schwarzen Strich. Wird nun das Produkt an der Kasse des Supermarktes gescannt, so kann es passieren, dass ein Bit (d. h. eine 0 oder 1) falsch gelesen wird. Die Codewörter für die Zahlen und für die Produkte sind so gewählt, dass der Scanner den Fehler entdeckt und der Kassiererin ein Signal gibt, den Scanning Vorgang zu wiederholen. Jeder von uns hat dies sicher schon des öfteren erlebt.

Einer der frühesten Erfolge von Fehler-korrigierenden Codes war die Qualität der Bilder von Satelliten, z. B. vom Mars. Wären diese Bilder ohne die Verwendung von Codes zur Erde übertragen worden, so wäre überhaupt nichts zu erkennen gewesen. Die Kontrollstationen hätten nur einen unverständlichen Bildsalat erhalten (in der Fachsprache: random noise).

Der erste effiziente Fehler-korrigierende Code wurde von R. W. Hamming 1948 in den Bell Laboratories entworfen. Eine binäre Folge (d. h. eine Folge von 0 und 1) wurde in Blöcke zu je 4 Bits zerlegt. Hammings Idee war es, jedem 4-Block drei *Korrekturbits* anzuhängen, mit Hilfe der folgenden Regel. Betrachten wir das folgende Diagramm:



Die 4 Bits (z. B. 1101) werden in die Teile mit den Nummern 1 bis 4 geschrieben. Die Korrekturbits kommen von den Teilen 5 bis 7. Die Regel ist, dass jeder Kreis eine *gerade* Anzahl von 1'en enthalten muss. Eine Konfiguration von 7 Bits, welche *einen* Fehler enthält, ergibt einen oder mehrere Kreise mit einer ungeraden Anzahl von 1'en. Das Bit, welches in all diesen Kreisen mit ungerader Parität enthalten ist, aber nicht in jener mit gerader Parität, muss dann das fehlerhafte Bit sein.

Wir sehen in diesem Beispiel, dass jedes Wort der Länge 7 genau 4 Informationsbits enthält. Daher sagen wir, dass der Hamming Code ein binärer $[7,4]$-Code ist mit *Informationsrate* $\frac{4}{7}$. Wir bemerken, dass die einfache Wiederholung der 4 Informationsbits einen $[8,4]$-Code ergibt, also einen Code mit Informationsrate $\frac{4}{8} = \frac{1}{2}$ (schlechter als im Hamming Code). Und dieser Code kann zwar einen Fehler entdecken, aber nicht korrigieren.

Von Musik zu Audiobits

Bevor wir uns den Codes zuwenden, wollen wir besprechen, wie Musik in eine Folge sogenannter Audiobits umgewandelt wird. Im CD-System wird das Analogsignal 44100 Mal pro Sekunde bemustert (in der Fachsprache „sampled"). Diese Rate von 44,1 kHz ist ausreichend, um Frequenzen bis zu 20000 Hz hören zu können. Die Muster werden uniform in 16 Bits zerlegt, und da wir Stereo-Musik empfangen wollen, so entspricht jedes Muster einer Folge von 32 Bits. So eine Folge von 32 Bits wird aufgefasst als 4 aufeinanderfolgende Bytes (ein Byte ist eine Folge von 8 Bits). Für den Codierungsvorgang werden diese Bytes als Elemente das Körpers $\mathbb{F}_{2^8}$ aufgefasst. Wer nicht weiß, was ein (mathematischer) Körper ist: In einem Körper gelten die üblichen Rechenregeln, d. h. wir können Bytes addieren, multiplizieren und dividieren, wie wir es gewohnt sind.

Genauso wie bei den Hamming-Codes wird die Folge der Bytes in Gruppen fester Länge zerlegt, an die dann Korrekturbytes angehängt werden. Im CD-System wird eine Folge von 24 Bytes in zwei Schritten zu einem Codewort der Länge 32 Bytes verwandelt. Die dabei verwendeten Codes sind sogenannte *Reed–Solomon-Codes* vom Typ [28,24] bzw. [32,28], die im nächsten Abschnitt erklärt werden. Schließlich wird noch ein Extrabyte hinzugefügt, welches die Kontroll- und Anzeigeinformation enthält. So kann der CD-Player (und der Hörer) erken-

nen, auf welcher Spur sich die CD gerade befindet. Also: Sechs Muster führen zu 33 Datenbytes, von denen jedes aus 8 Bits besteht.

Wie später erklärt wird, können diese Datenbytes noch nicht direkt auf die CD übertragen werden. Die Transformation in die Sequenz von Kanalbits (welche dann die Tonspur ergeben) wandelt 8 Datenbits in 17 Kanalbits um, und nach jeder Folge von 33 mal 17 Kanalbits wird eine Sequenz von 27 Synchronisationsbits eingefügt. Diese Synchronisationsbits identifizieren den Beginn der nächsten Codesequenz. Die Synchronisationszeichen sind natürlich so gewählt, dass sie auf keinen Fall in einer Codesequenz auftreten können, eine Verwechslung also unmöglich ist.

Nehmen wir alles zusammen, so erhalten wir folgendes Schema:

6 Muster $\longrightarrow 33 \times 8 = 264$ Datenbits $\longrightarrow 264 \times \frac{33\times 17+27}{33\times 8} = 588$ Kanalbits.

Eine Sekunde Musik führt also zu einer Folge von 4321800 Bits auf der Tonspur. Falls der CD-Player ein Bit mit der sehr kleinen Wahrscheinlichkeit von 10^{-4} falsch interpretieren würde, so würden immer noch hunderte Fehler pro Sekunde passieren!

Es gibt mehrere Gründe für Fehler auf einer CD. Es könnte Schmutz auf der CD sein, Luftblasen im Plastikmaterial, Ungenauigkeiten beim Druck, Fingerabdrücke, Kratzer, Oberflächenfehler. Es sei angemerkt, dass Fehler hauptsächlich hintereinander auftreten (sogenannte „burst errors“). Da wir Codes verwenden, die *zufällige* Fehler korrigieren, so erfordert dies ein systematisches Austauschen von Bytes, damit Bytes, die in den Codewörtern aufeinanderfolgen, nicht mehr benachbart sind auf der CD.

Reed–Solomon-Codes

Das Codierungssystem, das auf einer Compact Disc verwendet wird, hat den Fachnamen CIRC (Cross-Interleaved Reed–Solomon-Code). Der Inhalt dieses Abschnittes sind die mathematischen Prinzipien, die in der Fehlerkorrektur auf CDs eine Rolle spielen. Wie wir gesehen haben, besteht das Code-Alphabet auf der CD aus $2^8 = 256$ Buchstaben, welche den Körper $\mathbb{F}_{2^8}$ bilden. Um die etwas komplizierten Rechnungen in $\mathbb{F}_{2^8}$ zu vermeiden, illustrieren wir die Arbeitsweise der Reed–Solomon-Codes an einem Beispiel über einem Alphabet mit 31 Elementen, d. h. über dem Körper $\mathbb{F}_{31}$. Die Elemente von $\mathbb{F}_{31}$ sind $0, 1, 2, \ldots, 30$ und die Rechenoperationen werden wie üblich ausgeführt, wobei das Ergebnis jeweils modulo 31 reduziert wird. Das heißt: Wollen wir z.B. 6 und 9 multiplizieren, so erhalten wir $6 \times 9 = 54$ und dividieren anschließend das Resultat durch 31. Das „Produkt“ 6×9 in F_{31} ist dann der Rest 23

$$6 \times 9 = 54 = 31 + 23.$$

Wir schreiben kurz: $6 \times 9 = 23$ (modulo 31).

Als ein Beispiel betrachten wir ein Buch mit 200 Seiten, das beschrieben werden soll mit einer Druckgröße, die 3000 Symbole pro Seite erlaubt. Wir identifizie-

ren die Buchstaben a bis z mit den Zahlen 1 bis 26, den Zwischenraum mit 0, und benutzen 27 bis 30 für Punkt, Komma, Strichpunkt und ein weiteres Symbol.

Wir werden vom Drucker informiert, dass die Technik, die er benutzt, nicht allzu gut ist. Tatsächlich ist sie so schlecht, dass jedes Symbol auf einer Seite mit Wahrscheinlichkeit 0,1% inkorrekt ist. Das bedeutet, dass im Durchschnitt 3 Druckfehler pro Seite passieren, was eindeutig zu viel ist. Man bedenke nur, wie lästig schon ein gelegentliches Klopfen auf der Schallplatte ist. Um die Situation zu verbessern, verwenden wir Ideen aus der Codierungstheorie.

In einem ersten Ansatz zerlegen wir die Folge der Symbole in Gruppen zu je 4. Vor jeder Gruppe von 4 Symbolen werden zwei 2 Korrektursymbole eingefügt. Wir bezeichnen das resultierende Codewort mit $\boldsymbol{a} = (a_0, a_1, \ldots, a_5)$, wobei a_0 und a_1 die Korrektursymbole und a_2, a_3, a_4, a_5 die 4 Informationssymbole sind (alles in $\mathbb{F}_{31}$).

Betrachten wir das Word CODE. Dieses Wort korrespondiert zu $(a_2, a_3, a_4, a_5) = (3, 15, 4, 5)$. Die Codierungsregeln, um a_0 und a_1 zu berechnen, sind nun:

(i) $a_0 + a_1 + a_2 + a_3 + a_4 + a_5 = 0$ (modulo 31)

(ii) $a_1 + 2a_2 + 3a_3 + 4a_4 + 5a_5 = 0$ (modulo 31).

Daraus erhalten wir $a_0 = 3$ und $a_1 = 1$, so dass CODE als CACODE codiert wird.

Gehen wir zum anderen Ende und nehmen wir an, wir erhalten EPGOOF. Wenn wir die Korrektursymbole auslassen, so würde GOOF als gesendetes Word resultieren. Aber wir sehen sofort, dass ein Fehler passiert sein muß, denn EPGOOF korrespondiert zu $(a_0, a_1, \ldots, a_5) = (5, 16, 7, 15, 15, 6)$ und die Summe der a_i ist

$$5 + 16 + \ldots + 6 = 64 = 2 \quad \text{(modulo 31)},$$

was unserer Codierungsregel widerspricht. Wir vermuten, dass wahrscheinlich eines der Symbole a_i durch $a_i + 2$ ersetzt wurde. Der Wert des Fehler in a_i ist 2, das heißt wir werden $2i$ als Fehlerwert erhalten, wenn wir $a_0, \ldots, a_5$ in Gleichung (ii) einsetzen. Einsetzen ergibt

$$a_1 + 2a_2 + \ldots + 5a_5 = 16 + 14 + 45 + 60 + 30 = 165 = 10 \quad \text{(modulo 31)},$$

also ist $i = \frac{10}{2} = 5$, das heißt der Fehler passierte in der fünften Position. Anstelle von 6 sollte 4 stehen – und wir decodieren EPGOOF in GOOD! Wie schon gesagt, die arithmetischen Operationen in $\mathbb{F}_{2^8}$ sind ein wenig komplizierter, aber alle Reed–Solomon-Codes verwenden Codierungs- and Decodierungsregeln wie die Gleichungen (i) und (ii).

Wenn wir nun auch wissen, dass dieser Code einen Fehler in einer 6-Gruppe korrigieren kann, so müssen wir ehrlicherweise auf ein Problem eingehen, das auch auf der CD auftritt. Der Code hat eine Informationsrate von $\frac{4}{6} = \frac{2}{3}$. Da wir dieselbe Anzahl von Seiten derselben Größe benutzen wollen, so müssen wir also $1\frac{1}{2}$-mal so viele Symbole pro Seite unterbringen. Zu unserer Ernüchterung teilt uns aber der Drucker mit, dass die Druckfehlerwahrscheinlichkeit der $1\frac{1}{2}$-mal kleineren Druckgröße *doppelt so hoch* ist! Mit anderen Worten: Mit dem kleineren Font müssen wir im Durchschnitt 9 Fehler pro Seite *vor der Fehlerkorrektur* in Kauf

nehmen. Und nach der Korrektur? Ein Wort mit 6 Symbolen wird korrekt decodiert, falls es höchstens einen Fehler enthält. Die Wahrscheinlichkeit, dass mehr als ein Fehler passiert ist

$$\sum_{i=2}^{6} \binom{6}{i} (0.002)^i (0.998)^{6-i} \approx 0.00006\,.$$

Im Buch wird es einige Wörter geben, die zwei oder mehr Druckfehler enthalten, aber es werden im Schnitt nicht mehr als 10 Wörter im gesamten Buch sein – eine bemerkenswerte Verbesserung!.

Mit einer geringfügigen Veränderung wird die Situation dramatisch besser. Dazu verwenden wir einen etwas komplizierteren Reed–Solomon Code. Wir zerlegen die Symbole in Gruppen von 8, die wir mit $(a_4, a_5, \ldots, a_{11})$ bezeichnen. Davor fügen wir 4 Korrekturzeichen hinzu nach den folgenden Regeln:

(i) $a_0 + a_1 + a_2 + a_3 + \ldots + a_{11} = 0$ (modulo 31)

(ii) $a_1 + 2^k a_2 + 3^k a_3 + \ldots + 11^k a_{11} = 0$ (modulo 31) $(k = 1, 2, 3)$.

Die Codierung besteht also aus der Lösung von 4 Gleichungen mit den 4 Unbekannten a_0, a_1, a_2, a_3. Der Leser kann sich sofort wie oben überlegen, wie ein Fehler korrigiert wird. Nehmen wir an, es passierten zwei Fehler, mit den Abweichungen e_1 und e_2 in Positionen i und j. Einsetzen in die Gleichungen (i) und (ii) ergibt sofort

$$e_1 + e_2 \text{ und } i^k e_1 + j^k e_2 \quad (k = 1, 2, 3).$$

Daraus können wir e_1 und e_2 eliminieren und erhalten eine quadratische Gleichung mit i und j als Lösungen. Danach können wir e_1 und e_2 ermitteln und die beiden Fehler korrigieren. Der neue Code ist also ein 2-fehlerkorrigierender Code.

Glücklicherweise ergibt das kein neues Problem beim Drucken. Dieser neue Reed–Solomon-Code hat wiederum Informationsrate $\frac{2}{3}$ $(= \frac{8}{12})$ wie der vorige, so dass die Druckfehlerwahrscheinlichkeit wieder $= 0,2\%$ pro Symbol ist. Die Decodierung wird nur versagen, wenn ein gedrucktes Wort von 12 Symbolen mehr als zwei Fehler enthält. Die Wahrscheinlichkeit, dass dies passiert, ist

$$\sum_{i=3}^{12} \binom{12}{i} (0.002)^i (0.998)^{12-i} < 0.000002.$$

Das gesamte Buch hat 600.000 Symbole, also 75.000 Wörter der Länge 8. Da $75000 \times 0.000002 = 0,15$ ist, so ist es unwahrscheinlich, dass überhaupt ein Druckfehler auftritt! Das ist nun tatsächlich eine eindrucksvolle Verbesserung!

Auf einer CD ist die Informationsrate, wie im letzten Abschnitt erwähnt, $\frac{24}{32} = \frac{3}{4}$. Wie eben erläutert (am Beipiel des kleineren Fonts) wird die Fehlerwahrscheinlichkeit für ein Bit dadurch erhöht im Vergleich zur Situation, wenn keine Korrektur durchgeführt wird. Wie wir schon erwähnt haben, kann eine CD ohne

weiteres 500.000 Bitfehler enthalten. Allerdings passieren diese Fehler nicht zufällig (wie in unserem Beispiel des Buches), sondern meistens in „bursts“ (hintereinander). Es kann vorkommen, dass einige 1000 aufeinanderfolgende Symbole (z. B. durch einen Kratzer) fehlerhaft sind. Dem wird entgegengewirkt, indem benachbarte Informationssymbole in *verschiedenen* Codewörtern aufscheinen (genannt „Interleaving“). Eines muss man sich außerdem bei Design eines CD-Systems vor Augen halten: Der Speicher, der für die Bits zur Verfügung steht, kann nicht zu groß sein. Das impliziert natürlich Beschränkungen bei der Länge der Codewörter und beim Interleaving. Alle Berechnungen, die zur Fehlerkorrektur benötigt werden, passieren im Bruchteil einer Sekunde, und das ist der Grund, warum wir Musik hören, praktisch unmittelbar, nachdem der CD-Player eingeschaltet wird. In dem Buch-Beispiel sahen wir, dass ein längerer Code zwar bessere Korrekturleistung bringt, aber auch mehr Berechnungen benötigt.

Wie wir im letzten Abschnitt erwähnt haben, wird die Codierung auf der CD in zwei Schritten mit Hilfe eines [28,24]-Codes C_1 und eines [32,28]-Codes C_2 ausgeführt. Diese Codes sind ziemlich kurz und nicht besonders leistungsstark. Einer der Hauptgründe für die Effizienz dieses Codierungsschemas besteht darin, dass die beiden Codes *zusammenarbeiten*. Anstelle des tatsächlichen Schemas, das auf der CD benutzt wird, beschreiben wir eine sehr ähnliche Situation, in der zwei Codes C_1 und C_2 zusammenarbeiten – mit bemerkenswerten Ergebnissen. Die Idee ist, ein *Produkt* von Codes einzuführen. Angenommen, 24×28 Informationssymbole werden in einer rechteckigen Matrix mit 24 Zeilen und 28 Spalten hingeschrieben. Diese 24×28-Matrix wird als der linke obere Teil einer 28×32-Matrix A aufgefasst. Die verbleibenden Stellen von A werden definiert, indem man fordert, dass jede Spalte von A (Länge 28) im Code C_1 ist und jede Zeile von A (Länge 32) im Code C_2 ist. Aus den Rechenregeln folgt, dass dies möglich ist.

Es ist nicht schwer sich zu überlegen, dass dieser „Produkt-Code“ Minimalabstand 25 hat und daher bis zu 12 Fehler in dem Codewort (welches jetzt eine 28×32-Matrix ist) korrigieren kann. Betrachten wir eine besonders schlechte Situation: Es treten 21 Fehler auf, und zwar 12 Spalten mit 1 Fehler, 1 Spalte mit 2 Fehlern, 1 Spalte mit 3 Fehlern, und eine Spalte mit 4 Fehlern. Wir können nicht erwarten, dass wir diese Situation in den Griff bekommen und tatsächlich ist die Decodierung der Spalten nicht möglich. Wir entscheiden uns, C_1 zu benutzen, um nur *einen* Fehler zu korrigieren. Das hat nun den Effekt, dass wir die beiden Spalten *erkennen*, welche zwei oder drei Fehler enthalten, da sie Abstand mindestens 2 zum nächsten Codewort in C_1 haben. Im Fall der Spalte mit 4 Fehlern ist es allerdings durchaus möglich, dass der Korrektur-Algorithmus glaubt, er hätte einen Fehler korrigiert, während in Wirklichkeit ein fünfter Fehler eingeführt wurde! Hier ist der Trick, der dies umgeht: Wir korrigieren die Spalten und erklären die zwei Spalten (mit 2 oder 3 Fehlern) als *gelöscht*. Das bedeutet, dass nach der Korrektur aller Spalten, 29 Spalten korrekt, 2 gelöscht sind und eine Spalte 5 Fehler hat. Nun betrachten wir die Zeilen, welche ja von C_2 herkommen. In jeder Zeile gibt es zwei gelöschte Stellen und in fünf der Zeilen ist ein Fehler. Da C_2 Minimalabstand 5 hat, können alle Zeilen richtig decodiert werden.

Die Compact Disc

Die Musik wird auf einer Compact Disc in Digitalform als eine 5 km lange spiralförmige Spur aufgezeichnet, welche aus einer Folge von sogenannten *Pits* besteht. Die Teile zwischen aufeinanderfolgenden Pits heißen *Lands*. Die Steigung der Spur ist 1,6μm, die Breite ist 0,6μm und die Tiefe der Pits ist 0,12μm. Die Abbildung zeigt eine Vergrößerung mehrerer paralleler Spuren.

Die Spur wird optisch von einem Laserstrahl gescannt. Jeder Land/Pit oder Pit/Land Übergang wird als 1 interpretiert, und alle anderen Bits als 0. Ein Kanalbit auf der Spur hat Länge 0,3 μm. So wird z. B. ein Pit der Länge 1,8 μm gefolgt von einem Land der Länge 0.9 μm übersetzt in die Folge 100000100. Der Durchmesser des Lichtstrahls ist ungefähr 1 μm. Wenn er auf ein Land fällt, so wird er fast völlig reflektiert. Da die Tiefe eines Pits ungefähr 1/4 der Wellenlänge des Lichtes im Material der Disc ist, so bewirkt Interferenz, dass weniger Licht von den Pits in die Öffnung des Objektives reflektiert wird.

Es gibt mehrere Anforderungen an die Folge der Bits, die in der CD beachtet werden müssen. Jedes Pit oder Land muss mindestens 3 Bits lang sein. Das wird gefordert, um zu vermeiden, dass der Lichtstrahl zwei aufeinanderfolgende Übergänge zwischen Pits und Lands gleichzeitig registriert, was eine sogenannte Intersymbol-Interferenz bewirken würde. Jedesmal wenn ein Übergang auftritt, wird die „Bit-Uhr“ im CD-Player synchronisiert. Da dies so rechtzeitig erfolgen muß, um einen Verlust an Synchronisation zu vermeiden, darf kein Pit oder Land länger als 3,3μm sein, d. h. zwei 1'en in der Sequenz sind durch höchstens 10 0'en getrennt. Die Einschränkung an die maximale Landlänge ist außerdem notwendig für den Mechanismus, der den Laser auf der Spur hält. Eine letzte Anforderung ist, dass der Niedrig-Frequenz Anteil des Signals auf ein Minimum beschränkt ist. Dies bedingt, dass die Gesamtlänge der Pits und Lands, von Beginn der Spur an, ungefähr gleich ist. Diese Bedingung wird für den Mechanismus benötigt, der entscheidet, ob etwas „hell“ oder „dunkel“ ist. (Das vermindert auch den negativen Effekt eines Fingerabdruckes auf der CD.) Als letztes betrachten wir nun die Konversion der Bytes (welche die Symbole in unseren Codewörtern sind) in die Sequenzen, welche endgültig auf die Disc übertragen werden. Diese Konversion wird EFM bezeichnet (= Eight-to-Fourteen-Modulation). Warum 14? Klarerweise

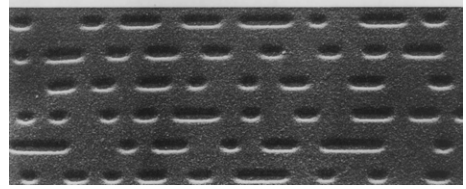


Abbildung 2. Vergrößerung mehrerer paralleler Spuren einer CD

wird eine Zahl benötigt, die den obigen Anforderungen genügt und so klein wie möglich ist.

Ein altbekanntes Problem in der Kombinatorik ist die Bestimmung der Anzahl der Folgen von 0 und 1 der Länge n, so dass keine zwei aufeinanderfolgenden 1'en auftreten. Nennen wir diese Zahl F_n. Eine Folge, die mit 1 endet, muß eine 0 an vorletzter Stelle haben. Davor kann jede Folge mit $n-2$ Symbolen sein (ohne aufeinanderfolgende 1'en). Endet aber die Folge in 0, so kann jede zulässige $(n-1)$–Folge davor auftreten. Wir erhalten somit

$$F_n = F_{n-1} + F_{n-2}.$$

Da $F_0 = 1$, $F_1 = 2$, so erhalten wir $F_2 = 3$, $F_3 = 5$, $F_4 = 8$ und allgemein die berühmte Folge der *Fibonacci-Zahlen*.

In unserem Fall ist die Sache nur unwesentlich schwieriger. Es sei $a(n)$ die Anzahl der Folgen von 0 und 1 der Länge n, in denen zwischen zwei 1'en mindestens zwei 0'en auftreten, aber niemals mehr als 10 aufeinanderfolgende 0'en. Für diese Größe $a(n)$ kann eine ähnliche Rekursion wie für die Fibonacci Zahlen aufgestellt werden. Wir müssen nun die $2^8 = 256$ möglichen Bytes (0,1-Wörter der Länge 8) in Folgen übersetzen, welche den technischen Anforderungen, wie eben geschildert, genügen. Es ergibt sich, dass $n = 14$ der kleinste Wert ist, für den $a(n) > 256$ ist, nämlich $a(14) = 267$.

Wir müssen noch ein letztes Problem meistern. Falls zwei Folgen der Länge 14 (welche beide den Anforderungen genügen) aneinanderhängen, so kann es passieren, dass die Folge der 28 Symbole die Bedingungen verletzt. Nach Entfernung der 11 schwierigsten Folgen aus den 267 ergibt sich, dass es möglich ist, zwei 14-er Sequenzen mit Hilfe von 3 „Merging Bits" zu vereinigen, wobei wir einige Freiheit in der Wahl dieser 3 Bits haben. Diese werden dann so gewählt, dass die Gesamtlänge der Pits und Lands wie gefordert ungefähr gleich ist.

Wir haben gezeigt, dass jedes Byte in einem Codewort in 17 Kanalbits auf der CD umgewandelt wird. Wenn die CD gelesen wird, so werden diese Bits zurück verwandelt in Bytes. Dann folgt De-Interleaving, Fehlerkorrektur, Digital-zu-Analog Konversion und schließlich hören wir Musik, mit Dank an Mozart und anderen – und an *Mathematik*.

Literatur

Die Fachliteratur ist für Laien nur schwer zugänglich. Eine Einführung in die Codierungstheorie findet sich im folgenden Band:

Ralph-Hardo Schulz: *Codierungstheorie – eine Einführung*. Vieweg Verlag 1991, 2. Auflage 2003.

http://www.springer.com/978-3-658-09989-3

Alles Mathematik
Von Pythagoras zu Big Data
Aigner, M.; Behrends, E. (Hrsg.)
2016, XI, 472 S. 253 Abb., 117 Abb. in Farbe., Softcover
ISBN: 978-3-658-09989-3