Prof. Dr. Jürgen Dassow
Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik

Codierungstheorie
und
Kryptographie

Wintersemester 2008

Inhaltsverzeichnis

Kapitel 1

Definition und Charakterisierung von Codes

1.1 Definition von Codes

Gegeben sei eine Menge A von zu codierenden Objekten. Um die Codierung vorzunehmen, ist sicher erst einmal jedem Element aus A ein Element des Codes C zuzuordnen. Offensichtlich muss die dadurch definierte Funktion eineindeutig sein, wenn wir eine eindeutige Decodierung erwarten. Dies bedeutet nur, dass die Mächtigkeiten der Mengen A und C übereinstimmen müssen.

Diese Forderung reicht aber nicht aus, wenn wir Nachrichten übermitteln wollen, die aus Folgen der zu codierenden Objekte aus A bestehen. Um dies zu sehen, betrachten wir die Mengen

$$A = \{A_1, A_2, A_3\} \quad \text{und} \quad C_0 = \{a, ba, ab\},$$

deren Elemente vermöge

$$A_1 \leftrightarrow a, \quad A_2 \leftrightarrow ba, \quad A_3 \leftrightarrow ab$$

eineindeutig zugeordnet werden können. Empfangen wir die Information aba, so ist nicht zu erkennen, ob die Nachricht A_1A_2 oder A_3A_1 gesendet wurde.

Die folgende Definition berücksichtigt beide genannten Aspekte.

Definition 1.1 *Eine eineindeutige Funktion $\phi : A \to C$ ist eine Codierung der Menge A durch die nichtleere Sprache C über dem Alphabet X, wenn die homomorphe Erweiterung*[1] *ϕ^* von ϕ auf A^* eine injektive Funktion von A^* in X^* ist.*

Eine nichtleere Sprache C (über X) heißt Code, wenn C Wertevorrat einer Kodierung ist.

Die obige Definition bindet den Begriff Code an den einer Codierung. Eine Feststellung, ob eine Sprache C ein Code ist, erfordert daher das Auffinden einer zu codierenden Menge A. Hierfür eignet sich aber jede zu C gleichmächtige Menge, so dass die Wahl von A intuitiv bedeutungslos ist. Der nachfolgende Satz gibt eine Bedingung an, die äquivalent zur Codedefinition ist, aber nur die Menge C selbst betrifft.

[1] Für eine Abbildung $\alpha : A \to X^*$, die eine Menge A auf Wörter über dem Alphabet X abbildet, ist die homomorphe Erweiterung $\alpha^* : A^* \to X^*$ durch $\alpha(\lambda) = \lambda$ und $\alpha(a_1a_2 \dots a_n) = \alpha(a_1)\alpha(a_2) \dots \alpha(a_n)$ für $a_i \in A$, $1 \le i \le n$, definiert.

Satz 1.1 *Eine nichtleere Sprache C ist genau dann ein Code, wenn für beliebige*

$$x_{i_1}, x_{i_2}, \ldots, x_{i_n}, x_{j_1}, x_{j_2}, \ldots, x_{j_m} \in C,\ n \geq 1, m \geq 1$$

gilt, dass

$$x_{i_1}x_{i_2}\ldots x_{i_n} = x_{j_1}x_{j_2}\ldots x_{j_m} \quad \textit{impliziert} \quad x_{i_1} = x_{j_1}. \tag{1.1}$$

Beweis. Wir nehmen zuerst an, C sei ein Code. Dann gibt es eine Menge A und eine Kodierung $\phi : A \to C$. Es seien

$$a_{i_t} = \phi^{-1}(x_{i_t}) \quad \text{und} \quad a_{j_s} = \phi^{-1}(x_{j_s})$$

für $1 \leq t \leq n$ und $1 \leq s \leq m$. Dann gilt

$$\phi^*(a_{i_1}a_{i_2}\ldots a_{i_n}) = x_{i_1}x_{i_2}\ldots x_{i_n} = x_{j_1}x_{j_2}\ldots x_{j_m} = \phi^*(a_{j_1}a_{j_2}\ldots a_{j_m}).$$

Aus der Injektivität von ϕ^* folgt nun

$$a_{i_1}a_{i_2}\ldots a_{i_n} = a_{j_1}a_{j_2}\ldots a_{j_m}.$$

Damit erhalten wir $a_{i_1} = a_{j_1}$ und folglich

$$x_{i_1} = \phi(a_{i_1}) = \phi(a_{j_1}) = x_{j_1},$$

womit Bedingung (1.1) nachgewiesen ist.

Erfülle nun die Menge C die Bedingung (1.1). Es sei A eine zu C gleichmächtige Menge. Dann gibt es eine eineindeutige Funktion $\phi : A \to C$. Wir zeigen nun indirekt, dass auch die Erweiterung von ϕ auf A^* injektiv ist.

Angenommen, die Erweiterung ist nicht injektiv. Dann gibt es Elemente $a_{i_1}, a_{i_2}, \ldots, a_{i_n}$, $a_{j_1}, a_{j_2}, \ldots, a_{j_m}$ so, dass

$$a_{i_1}a_{i_2}\ldots a_{i_n} \neq a_{j_1}a_{j_2}\ldots a_{j_m} \tag{1.2}$$

und

$$\phi^*(a_{i_1}a_{i_2}\ldots a_{i_n}) = \phi^*(a_{j_1}a_{j_2}\ldots a_{j_m}) \tag{1.3}$$

gelten. Wir wählen nun diese Elemente mit diesen Eigenschaften so, dass m minimal ausfällt. Weiterhin setzen wir

$$x_{i_t} = \phi(a_{i_t}) \quad \text{und} \quad x_{j_s} = \phi(a_{j_s})$$

für $1 \leq t \leq n$ und $1 \leq s \leq m$. Dann gilt

$$x_{i_1}x_{i_2}\ldots x_{i_n} = \phi^*(a_{i_1}a_{i_2}\ldots a_{i_n}) = \phi^*(a_{j_1}a_{j_2}\ldots a_{j_m}) = x_{j_1}x_{j_2}\ldots x_{j_m}.$$

Wegen Bedingung (1.1) erhalten wir $x_{i_1} = x_{j_1}$. Damit gilt $\phi(a_{i_1}) = \phi(a_{j_1})$, woraus $a_{i_1} = a_{j_1}$ folgt. Somit ergeben sich

$$a_{i_2}a_{i_3}\ldots a_{i_n} \neq a_{j_2}a_{j_3}\ldots a_{j_m}$$

(die Gleichheit dieser Elemente würde auch die Gleichheit von $a_{i_1}a_{i_2}\ldots a_{i_n}$ und $a_{j_1}a_{j_2}\ldots a_{j_m}$ nach sich ziehen, womit ein Widerspruch zu (1.2) gegeben wäre) und

$$\phi^*(a_{i_2}a_{i_3}\ldots a_{i_n}) = \phi^*(a_{j_2}a_{j_3}\ldots a_{j_m})$$

(dies folgt aus (1.3) mittels Kürzen von $\phi(a_{i_1}) = \phi(a_{j_1})$). Die beiden letzten Relationen ergeben einen Widerspruch zur Minimalität von m. □

Beispiel 1.1 Die Mengen

$$\begin{aligned} C_1 &= \{a, bb, aab, bab\}, \\ C_2 &= \{aa, bb, aba, baa\}, \\ C_3 &= \{aaa, aba, bab, bbb\}, \\ C_4 &= \{a, ab, bb\} \end{aligned}$$

über $X = \{a, b\}$ sind Codes. Wir zeigen dies nur für C_1; die Beweise für C_2, C_3 und C_4 können analog geführt werden, jedoch kann mittels der weiter unten gegebenen Definitionen für C_2 und C_3 direkt ein Nachweis geführt werden. Wir zeigen, dass Bedingung 1.1 erfüllt und damit C_1 nach Satz 1.1 ein Code ist.

Seien $x_{i_1}, x_{i_2}, \ldots, x_{i_n}, x_{j_1}, x_{j_2}, \ldots, x_{j_m}$ Elemente aus C_1 mit

$$x_{i_1} x_{i_2} \ldots x_{i_n} = x_{j_1} x_{j_2} \ldots x_{j_m}. \tag{1.4}$$

Das in (1.4) gegebene Wort sei w. Wir diskutieren nun die möglichen Fälle für x_{i_1}.

Fall 1. $x_{i_1} \in \{bb, bab\}$. Dann muss offenbar $x_{i_1} = x_{j_1}$ gelten, womit (1.1) nachgewiesen ist.

Fall 2. $x_{i_1} = aab$. Dann folgt $x_{i_1} = x_{j_1} = aab$ oder $x_{j_1} = a$. Im ersten Fall ist (1.1) erfüllt. Im zweiten Fall muss dann auch $x_{j_2} = a$ gelten und und für x_{j_3} gibt es zwei Möglichkeiten.

Fall 2.1. $x_{j_3} = bab$, d.h.

$$w = y x_{i_2} \ldots x_{i_n} = yab x_{j_4} \ldots x_{j_m}.$$

Folglich ist $x_{i_2} = a$, und es ergibt sich

$$w = z x_{i_3} \ldots x_{i_n} = zb x_{j_4} \ldots x_{j_m},$$

womit wir im wesentlichen die Situation aus Fall 2.2 erhalten.

Fall 2.2. $x_{j_3} = bb$, d.h.

$$w = y x_{i_2} \ldots x_{i_n} = yb x_{j_4} \ldots x_{j_m}.$$

Folglich ist $x_{i_2} = bb$ oder $x_{i_2} = bab$, und es ergibt sich

$$w = z' b x_{i_3} \ldots x_{i_n} = z' x_{j_4} \ldots x_{j_m}$$

oder

$$w = z'' ab x_{i_3} \ldots x_{i_n} = z'' x_{j_4} \ldots x_{j_m},$$

womit wir erneut im wesentlichen die Situation aus Fall 2.2 oder Fall 2.1 erhalten.

Die Situation der beiden Unterfälle wird also stets erneuert, womit gezeigt ist, dass (1.4) im Widerspruch zur Annahme nicht gilt.

Fall 3. $x_{i_1} = aab$. Wir können wie in Fall 2 zeigen, dass (1.1) erfüllt sein muss.

Wir merken an, dass ein Code das Leerwort nicht enthalten kann, denn wegen

$$\lambda x = x \lambda$$

für jedes Wort x des Codes ist Bedingung (1.1) nicht erfüllt.

Wir geben nun eine leichte Verallgemeinerung von Satz 1.1.

Satz 1.2 *Eine Sprache C ist genau dann ein Code, wenn für beliebige*

$$x_{i_1}, x_{i_2}, \ldots, x_{i_n}, x_{j_1}, x_{j_2}, \ldots, x_{j_m} \in C,\ n \geq 1, m \geq 1,$$

mit

$$x_{i_1}x_{i_2}\ldots x_{i_n} = x_{j_1}x_{j_2}\ldots x_{j_m}$$

die Gleichheiten

$$n = m \quad \text{und} \quad x_{i_t} = x_{j_t} \text{ für } 1 \leq t \leq n$$

gelten.

Beweis. Sei zuerst C ein Code, und es gelte $x_{i_1}x_{i_2}\ldots x_{i_n} = x_{j_1}x_{j_2}\ldots x_{j_m}$ für gewisse Wörter $x_{i_1}, x_{i_2}, \ldots, x_{i_n}, x_{j_1}, x_{j_2}, \ldots, x_{j_m} \in C$, $n \geq 1$, $m \geq 1$. Nach Satz 1.1 haben wir $x_{i_1} = x_{j_1}$. Damit gilt

$$x_{i_2}x_{i_3}\ldots x_{i_n} = x_{j_2}x_{j_3}\ldots x_{j_m}.$$

Erneut erhalten wir aus Satz 1.1 $x_{i_2} = x_{j_2}$. So fortfahrend erhalten wir alle gewünschten Gleichheiten.

Gelten umgekehrt die Gleichheiten, so folgt, dass C ein Code ist wegen $x_{i_1} = x_{j_1}$ aus Satz 1.1. □

Ausgehend von Satz 1.1 definieren wir eine spezielle Klasse von Codes.

Definition 1.2 *Ein Code C heißt strenger Code, wenn für beliebige $x_{i_k} \in C$ und $x_{j_k} \in C$, $k \geq 1$, für die für alle $n \geq 1$*

$$x_{i_1}x_{i_2}\ldots x_{i_n} \text{ ein Präfix von } x_{j_1}x_{j_2}\ldots x_{j_n}$$

oder

$$x_{j_1}x_{j_2}\ldots x_{j_n} \text{ ein Präfix von } x_{i_1}x_{i_2}\ldots x_{i_n}$$

ist, die Gleichheit $x_{i_1} = x_{j_1}$ gilt.

Offensichtlich gilt für strenge Codes die Satz 1.2 entsprechende Charakterisierung.

Bemerkung *Ein Code C ist ein strenger Code, wenn für beliebige $x_{i_k} \in C$ und $x_{j_k} \in C$, $k \geq 1$, für die für alle $n \geq 1$*

$$x_{i_1}x_{i_2}\ldots x_{i_n} \text{ ein Präfix von } x_{j_1}x_{j_2}\ldots x_{j_n}$$

oder

$$x_{j_1}x_{j_2}\ldots x_{j_n} \text{ ein Präfix von } x_{i_1}x_{i_2}\ldots x_{i_n}$$

ist, die Gleichheiten $x_{i_k} = x_{j_k}$ für $k \geq 1$ gelten.

Wir bemerken, dass die Forderung, dass

$$x_{i_1}x_{i_2}\ldots x_{i_n} \text{ ein Präfix von } x_{j_1}x_{j_2}\ldots x_{j_n}$$

oder

$$x_{j_1}x_{j_2}\ldots x_{j_n} \text{ ein Präfix von } x_{i_1}x_{i_2}\ldots x_{i_n}$$

ist, kürzer dadurch ausgedrückt werden kann, dass die Gleichheit

$$x_{i_1}x_{i_2}\ldots x_{i_n}\ldots = x_{j_1}x_{j_2}\ldots x_{j_n}\ldots$$

der „unendlichen Wörter“ $x_{i_1}x_{i_2}\ldots$ und $x_{j_1}x_{j_2}\ldots$ gilt.

Offensichtlich ist der Code C_3 aus Beispiel 1.1 ein strenger Code, denn da alle Wörter aus C_3 die Länge 3 haben und verschieden voneinander sind, müssen bei gleichem Präfix die ersten Wörter übereinstimmen.

C_4 ist dagegen kein strenger Code, denn mit $x_1 = a$, $x_2 = ab$ und $x_3 = bb$ gilt die Gleichheit

$$x_1x_3x_3x_3\ldots = x_2x_3x_3x_3\ldots = abbbbbbb\ldots$$

Wir definieren nun Klassen von Codes.

Definition 1.3 *Eine nichtleere Sprache C heißt Präfixcode, wenn kein Wort aus C Präfix eines anderen Wortes aus C ist.*

Wir zeigen, dass ein Präfixcode C ein Code ist. Seien dazu $x_{i_1}, x_{i_2}, \ldots, x_{i_n}, x_{j_1}, x_{j_2}, \ldots, x_{j_m}$ beliebige Elemente aus C mit

$$x_{i_1}x_{i_2}\ldots x_{i_n} = x_{j_1}x_{j_2}\ldots x_{j_m}.$$

Ohne Beschränkung der Allgemeinheit sei $|x_{i_1}| \leq |x_{j_1}|$. Dann gilt $x_{i_1}v = x_{j_1}$ für ein Wort $v \in X^*$. Wegen der Eigenschaft von Präfixcodes müssen $v = \lambda$ und $x_{i_1} = x_{j_1}$ gelten, womit Bedingung (1.1) als gültig nachgewiesen ist. Nach Satz 1.1 ist deshalb gezeigt, dass C ein Code ist.

C_1 aus Beispiel 1.1 ist ein Code, aber kein Präfixcode. C_2 und C_3 sind Präfixcodes.

Definition 1.4 *Sei $n \geq 1$ eine natürliche Zahl. Eine Teilmenge C von X^n heißt Blockcode der Länge n über X.*

Offensichtlich sind Blockcodes Präfixcodes und folglich Codes.

C_2 ist ein Präfixcode, aber kein Blockcode. C_3 ist ein Blockcode.

Satz 1.3 *Für einen Code C und eine natürliche Zahl $k \geq 1$ ist auch C^k ein Code.*

Beweis. Wir betrachten beliebige Elemente $y_{i_1}, y_{i_2}, \ldots, y_{i_n}, y_{j_1}, y_{j_2}, \ldots, y_{j_m} \in C^k$ mit

$$y_{i_1}y_{i_2}\ldots y_{i_n} = y_{j_1}y_{j_2}\ldots y_{j_m}. \tag{1.5}$$

Da jedes y_{i_t}, $1 \leq t \leq n$, und jedes y_{j_s}, $1 \leq s \leq m$, ein Produkt von jeweils k Elementen aus C ist, kann (1.5) als eine Gleichheit zwischen Elementen aus C^* aufgefasst werden (die linke Seite ist ein Produkt aus $n \cdot k$ Faktoren aus C, die rechte Seite enthält $m \cdot k$ Faktoren). Nach Satz 1.2 gilt $nk = mk$, d.h. $n = m$ und die Faktoren aus C sind entsprechend ihrer Reihenfolge gleich. Insbesondere bedeutet dies die Gleichheit der Produkte aus den ersten k Faktoren der linken und rechten Seite. Das bedeutet, dass $y_{i_1} = y_{j_1}$ gilt. Nach Satz 1.1 ist damit bewiesen, dass C^k ein Code ist. □

1.2 Codierung und Decodierung durch Automaten

Wir wollen nun zeigen, dass für beliebige Codes die Codierung und für strenge Codes auch die Dekodierung durch Automaten realisiert werden kann. Dafür benötigen wir Automaten, die ein Eingabe-Ausgabe-Verhalten beschreiben können.

Definition 1.5 *Ein Mealy-Automat ist ein 6-Tupel $\mathcal{A} = (X, Y, Z, f, g, z_0)$, wobei*
– X, Y, Z Alphabete (endliche nichtleere Mengen) sind,
– $f : Z \times X \to Z$ und $g : Z \times X \to Y^*$ Funktionen sind, und
– z_0 ein Element von Z ist.

X, Y und Z sind das Eingabe- bzw. Ausgabealphabet bzw. die Menge der Zustände. f und g werden Überführungs- bzw. Ausgabefunktion genannt. Befindet sich der Automat im Zustand z und liest das Symbol x, so geht er in den Zustand $g(z, x)$ und gibt $f(z, x)$ aus. z_0 heißt Anfangszustand.

Da wir bei Codierungen Folgen von Buchstaben, d.h. Wörter über dem Eingabealphabet, verarbeiten wollen, setzen wir die Funktionen f und g auf $Z \times X^*$ fort. Die entsprechenden Funktionen $f^* : Z \times X^* \to Z$ und $g^* : Z \times X^* \to Y^*$ definieren wir durch

$$f^*(z, \lambda) = z,\ g^*(z, \lambda) = \lambda,$$
$$f^*(z, wa) = f(f^*(z, w), a),\ g^*(z, wa) = g^*(z, w)g(f^*(z, w), a) \text{ für } w \in X^* \text{ und } a \in X.$$

$f^*(z, w)$ gibt den Zustand an, den der Automat vom Zustand z durch Abarbeitung des Eingabewortes $w \in X^*$ erreicht; $g^*(z, w)$ ist das dabei produzierte Ausgabewort.

Wir betrachten zwei Beispiele.

Beispiel 1.2 Der Automat $\mathcal{A} = (X, Y, Z, f, g, z_0)$ sei durch

$$X = \{A, B, C\},\ Y = \{a, b\} \text{ und } Z = \{z_0\},$$
$$f(z_0, A) = f(z_0, B) = f(z_0, C) = z_0,$$
$$g(z_0, A) = aa,\ g(z_0, B) = ab,\ g(z_0, C) = bb$$

gegeben. Dann erhalten wir $f(z_0, w) = z_0$ für jedes Wort $w \in X^*$ und zum Beispiel $g^*(z_0, CA) = bb\,aa$ und $g^*(z_0, ABBA) = aa\,ba\,ba\,aa$.

Beispiel 1.3 Wir wollen nun einen Mealy-Automaten angegeben, der (zumindest partiell) einen Automaten beschreibt, an dem Scheine zum Parken gezogen werden können. Die möglichen Eingaben für derartige Automaten sind 50-Cent-, 1-Euro- und 2-Euromünzen. Die möglichen Parkzeiten sind eine halbe Stunde, eine Stunde, anderthalb Stunden oder zwei Stunden, wobei für jeweils eine halbe Stunde 50 Cent zu zahlen sind. Längeres Parken ist nicht erlaubt. Der Parkschein wird aber nur ausgegeben, wenn durch Knopfdruck ein Parkschein angefordert wurde. Hieraus resultiert, dass wir als Eingabe- und Ausgabemenge

$$X = \{50, 1, 2, A\} \quad \text{und} \quad Y = \{30, 60, 90, 120\}$$

(A für Anfordern; Parkzeit in Minuten) verwenden können. Damit der Automat die richtige Parkdauer ausgibt, muss er sich den in den Automaten gezahlten Betrag merken. Folglich verwenden wir die Zustandsmenge

$$Z = \{0, 50, 100, 150, 200\}$$

(eingezahlter Betrag in Cent). Zwar können größere Beträge in den Automaten gesteckt werden, aber die Parkdauer wird dadurch nicht erhöht, stimmt also mit der von 200 Cent überein. Es ergeben sich die folgende Überführungs- und Ausgabefunktion, die wir durch eine Tabelle angeben, bei der die Zeilen den Eingabesymbolen und die Spalten den Zuständen entsprechen und im Schnittpunkt der Zeile zu x und der Spalte zu z das Paar $(f(z,x), g(z,x))$ angegeben ist.

	0	50	100	150	200
50	$(50,\lambda)$	$(100,\lambda)$	$(150,\lambda)$	$(200,\lambda)$	$(200,\lambda)$
1	$(100,\lambda)$	$(150,\lambda)$	$(200,\lambda)$	$(200,\lambda)$	$(200,\lambda)$
2	$(200,\lambda)$	$(200,\lambda)$	$(200,\lambda)$	$(200,\lambda)$	$(200,\lambda)$
A	$(0,\lambda)$	$(0,30)$	$(0,60)$	$(0,90)$	$(0,120)$

Als Anfangszustand verwenden wir 0, da zu Beginn kein Betrag gezahlt worden ist.

Für die sinnvollen Eingabefolgen, d.h. es werden nur Beträge 50 oder 100 oder 150 oder 200 Cent gezahlt und am Ende von jedem Nutzer die Anforderung A ausgelöst, ergibt sich unter Verwendung von Wörtern w_{x_i} (der i-te Nutzer hat den Betrag x_i durch seine Eingabefolge gezahlt) und $y_i = \frac{3}{5}x_i$

$$f^*(0, w_{x_1}Aw_{x_2}A\dots w_{x_k}A) = 0 \text{ und } g^*(0, w_{x_1}Aw_{x_2}A\dots w_{x_k}A) = y_1y_2\dots y_k.$$

Sei die Codierung $\phi : \{a_1, a_2, \dots, a_n\} \to \{c_1, c_2, \dots, c_n\} \subseteq X^*$ mit $\phi(a_i) = c_i$, $1 \le i \le n$, gegeben. Dann gilt für die homomorphe Erweiterung ϕ^* offenbar

$$\phi^*(a_{i_1}a_{i_2}\dots a_{i_m}) = c_{i_1}c_{i_2}\dots c_{i_m}.$$

Wenn wir diese Abbildung durch einen Automaten erzeugen wollen, so benötigen wir einen Mealy-Automaten, der auf die Eingabe a_i, $1 \le i \le n$, die Ausgabe c_i erzeugt. Eine Abhängigkeit von einem Zustand ist hier nicht erforderlich. Formal ergibt sich der Automat

$$\mathcal{A}_{cod} = (\{a_1, a_2, \dots, a_n\}, X, \{z\}, f, g, z)$$

mit

$$f(z, a_i) = z \text{ und } g(z, a_i) = c_i \quad \text{für} \quad 1 \le i \le n.$$

Offensichtlich ergibt sich dann

$$g^*(z, a_{i_1}a_{i_2}\dots a_{i_m}) = c_{i_1}c_{i_2}\dots c_{i_m} = \phi^*(a_{i_1}a_{i_2}\dots a_{i_m}).$$

Der Automat $\mathcal{A}_{cod}$ realisiert also gerade die durch ϕ gegebene Codierung.

Der in Beispiel 1.2 angegebene Automat ist der codierende Automat für die Codierung $\phi : \{A, B, C\} \to \{aa, ab, bb\}$.

Wir betrachten nun das umgekehrte Problem. Wir wollen eine Folge $c_{i_1}c_{i_2}\dots c_{i_m}$, die durch die Codierung ϕ entstanden ist, zurückübersetzen in die eingegebene Folge $a_{i_1}a_{i_2}\dots a_{i_m}$. Dieser Vorgang wird Decodierung genannt.

Wir wollen auch hier einen Automaten $\mathcal{A}_{dec} = (X, \{a_1, a_2, \dots, a_n\}, Z, f_1, g_1, z_0)$ konstruieren, für den

$$g_1^*(z_0, c_{i_1}c_{i_2}\dots c_{i_m}) = a_{i_1}a_{i_2}\dots a_{i_m} = (\phi^*)^{-1}(c_{i_1}c_{i_2}\dots c_{i_m})$$

gilt. Wir geben die Konstruktion zuerst für Präfixcodes und benutzen die Charakterisierung von Codes entsprechend Satz 1.1). Intuitiv bedeutet dies: Wir verarbeiten das Eingabewort ohne Ausgabe (d.h. mit Ausgabe des Leerwortes λ), bis wir ein Codewort c gelesen haben. Dann geben wir $\phi^{-1}(c)$ aus. Das verbleibende Eingabewort wird nun genauso verarbeitet. Das Testen, ob ein Codewort vorliegt, wird dadurch realisiert, dass wir uns den schon gelesenen Teil merken und mit den Codewörter vergleichen. Formal ergibt sich der endliche Mealy-Automat

$$\mathcal{A}_{dec} = (X, Y, Z, z_0, f, g)$$

mit der Menge X von Eingabesymbole, der Ausgabemenge

$$Y = A \cup \{\text{Fehler}\}$$

(neben den zu codierenden Elementen aus A, die als Ergebnis der Decodierung auftreten, enthält Y noch eine Fehlermeldung), der Zustandsmenge

$$Z = \{[w] : w \in \mathit{Präf}(C)\} \cup \{z_{Fehler}\}$$

(wobei $\mathit{Präf}(U) = \{w \mid wx = z \in U \text{ für gewisse } x, z\}$ die Menge der Präfixe von Wörtern aus U ist), dem Anfangszustand

$$z_0 = [\lambda]$$

(zu Beginn hat der Automat noch nichts gelesen), der Zustandsüberführungsfunktion $f : Z \times X \to Z$ und der Ausgabefunktion $g : Z \times X \to Y$, die durch

$$f(z,x) = \begin{cases} [wx] & z = [w] \text{ und } wx \text{ ist echter Präfix eines Wortes aus } C \\ [\lambda] & z = [w] \text{ und } wx \in C \\ z_{Fehler} & \text{sonst} \end{cases}$$

und

$$g(z,x) = \begin{cases} \lambda & z = [w] \text{ und } wx \text{ ist echter Präfix eines Wortes aus } C \\ a_i & z = [w] \text{ und } wx = c_i \in C \\ \lambda & z = z_{Fehler} \\ \text{Fehler} & \text{sonst} \end{cases}$$

gegeben sind. $\mathcal{A}$ liest die Eingabe und merkt sich das bereits gelesene Wort w als Zustand $[w]$ und gibt nichts (d.h. das Leerwort) aus, solange dies der Anfang eines Codewortes ist. Ist ein Codewort c_i vollständig gelesen worden, so vergisst $\mathcal{A}$ den bereits gelesenen Teil und gibt das Symbol a_i aus, das durch c_i codiert wird. Ist der gelesenen Teil kein Anfang eines Codewortes, so geht $\mathcal{A}$ in einen speziellen Fehlerzustand z_{Fehler} und gibt eine Fehlermeldung aus. Liegt bereits der Fehlerzustand vor, so bleibt $\mathcal{A}$ in diesem Zustand und gibt nichts mehr aus. Daher überführt der Automat $\mathcal{A}$ das Eingabewort $c_{i_1}c_{i_2}\ldots c_{i_m}$ mit $c_{i_j} \in C$, $1 \leq j \leq m$, in das Ausgabewort $a_{i_1}a_{i_2}\ldots a_{i_m}$, womit eine korrekte Decodierung vorgenommen wird. Ist dagegen das Eingabewort kein Produkt von Codewörtern, wird ein Wort der Form yFehler ausgegeben, womit ausgesagt wird, dass die Eingabe nicht decodierbar ist, da sie keine Codierung eines Wortes über A ist.

Offenbar liefert der oben konstruierte Automat $\mathcal{A}_{dec}$ nur für Präfixcodes ein richtiges Ergebnis. Dies ist wie folgt zu sehen: Enthält der Code zwei Wörter x und xy für ein

gewisses $y \in X^*$, so weiß der Automat nach Lesen von x nicht, ob er das Codewort x oder nur den Anfang von xy gelesen hat.

Wir bemerken, dass sich der Gedanke aber auch für strenge Codes nutzen lässt, indem man die Entscheidung, ob der Automat ein Codewort oder den echten Präfix eines Codewortes gelesen hat, erst etwas später trifft. Für die entsprechende Konstruktion des Automaten verweisen wir auf [9].

Diese Ausführung lassen sich zu folgendem Satz zusammenfassen.

Satz 1.4 *Es gibt einen Algorithmus, der für jede strenge Codierung $\phi : A \to C \subseteq X^+$ und jedes Wort $x \in X^+$ in linearer Zeit $(\phi^*)^{-1}(x)$ berechnet bzw. feststellt, dass $(\phi^*)^{-1}(x)$ nicht definiert ist.* □

1.3 Entscheidbarkeit der Eigenschaft, Code zu sein

Wir geben zuerst eine weitere Charakterisierung von Codes an.

Definition 1.6 *Eine Sprache L heißt produktunabhängig, falls kein Wort in L als Produkt von mindestens zwei Wörtern aus L dargestellt werden kann.*

Bei einer produktunabhängigen Sprache L gilt nach Definition für jedes Produkt $w = x_1 x_2 \dots x_n$ mit $n \geq 2$ und $x_i \in L$ für $1 \leq i \leq n$ die Relation $w \notin L$.

Offenbar ist jeder Code wegen Bedingung (1.1) und Satz 1.1 produktunabhängig. Andererseits ist die zu Anfang des Abschnitts betrachtete Menge $\{a, ab, ba\}$ offensichtlich produktunabhängig, aber kein Code.

Satz 1.5 *Sei C eine produktunabhängige Menge über einem Alphabet X. Dann ist C genau dann ein Code, wenn für jedes Wort $w \in X^*$ gilt, dass*

$$wC^* \cap C^* \neq \emptyset \text{ und } C^*w \cap C^* \neq \emptyset \quad \text{implizieren} \quad w \in C^*. \tag{1.6}$$

Beweis. Wir nehmen zuerst an, dass (1.6) für jedes $w \in X^*$ gilt, und zeigen, dass C ein Code ist.

Angenommen, C wäre kein Code. Dann gibt es wegen Satz 1.1 Elemente $x_{i_1}, x_{i_2}, \dots, x_{i_n}$, $x_{j_1}, x_{j_2}, \dots, x_{j_m} \in C$ mit

$$x_{i_1} x_{i_2} \dots x_{i_n} = x_{j_1} x_{j_2} \dots x_{j_m} \quad \text{und} \quad x_{i_1} \neq x_{j_1}.$$

Ohne Beschränkung der Allgemeinheit gelte $|x_{i_1}| \leq |x_{j_1}|$. Dann folgt die Existenz eines Wortes $y \in X^+$ mit

$$x_{j_1} = x_{i_1} y \quad \text{und} \quad x_{i_2} x_{i_3} \dots x_{i_n} = y x_{j_2}, x_{j_3} \dots x_{j_m}.$$

Damit gelten

$$x_{j_1} \in C^* \cap C^* y \quad \text{und} \quad x_{i_2} x_{i_3} \dots x_{i_n} \in C^* \cap y C^*.$$

Somit erhalten wir $y \in C^*$. Damit erhalten wir einen Widerspruch zur Produktunabhängigkeit von C, da x_{j_1} wegen $x_{j_1} = x_{i_1} y$ eine Produktdarstellung aus zwei Faktoren aus C besitzt.

Sei nun C ein Code. Wir zeigen, dass (1.6) für jedes $w \in X^*$ folgt.

Angenommen , dies wäre nicht der Fall. Dann muss es Elemente

$$x_{i_1}, x_{i_2}, \ldots, x_{i_n}, x_{j_1}, x_{j_2}, x_{j_m}, y_{k_1}, y_{k_2}, \ldots, y_{k_t}, y_{l_1}, y_{l_2}, \ldots, y_{l_s} \in C,\ n \geq 0, m \geq 0, t \geq 0, s \geq 0,$$

und

$$w \notin C^*$$

so geben, dass

$$x_{i_1}x_{i_2}\ldots x_{i_n}w = x_{j_1}x_{j_2}\ldots x_{j_m} \in C^*w \cap C^* \quad \text{und} \quad wy_{k_1}y_{k_2}\ldots y_{k_t} = y_{l_1}y_{l_2}\ldots y_{l_s} \in wC^* \cap C^*$$

gelten. Seien die Elemente so gewählt, dass m minimal ausfällt. Dann muss $x_{i_1} \neq x_{j_1}$ gelten. Damit erhalten wir

$$\begin{aligned} x_{j_1}x_{j_2}\ldots x_{j_m}y_{k_1}y_{k_2}\ldots y_{k_t} &= x_{i_1}x_{i_2}\ldots x_{i_n}wy_{k_1}y_{k_2}\ldots y_{k_t} \\ &= x_{i_1}x_{i_2}\ldots x_{i_n}y_{l_1}y_{l_2}\ldots y_{l_s}. \end{aligned}$$

Wegen $x_{i_1} \neq x_{j_1}$ ist somit die Bedingung (1.1) verletzt, woraus wegen Satz 1.1 ein Widerspruch dazu resultiert, dass C ein Code ist. □

Die Bedingung (1.6) aus Satz 1.5 kann noch verschärft werden:

$$C^*w \cap wC^* \cap C^* \neq \emptyset \quad \text{impliziert} \quad w \in C^* \tag{1.7}$$

gilt für jedes $w \in X^*$. Dies kann wie folgt eingesehen werden.

(1.6) $\Rightarrow$ (1.7) . Sei $C^*w \cap wC^* \cap C^* \neq \emptyset$. Dann sind auch $wC^* \cap C^*$ und $C^*w \cap C^*$ nichtleere Mengen. Damit gilt $w \in C^*$ wegen (1.6), womit (1.7) bewiesen ist.

(1.7) $\Rightarrow$ (1.6). Es gelte

$$wx_1 = x_2 \text{ und } x_3w = x_4 \quad \text{für gewisse } x_1, x_2, x_3, x_4 \in C^*.$$

Dann gilt auch

$$wx_1x_4 = x_2x_4 = x_2x_3w,$$

womit gezeigt ist, dass es ein Element in $wC^* \cap C^* \cap C^*w$ gibt. Somit erhalten wir $w \in C^*$ wegen (1.7). Daher ist (1.6) bewiesen.

Wir merken noch an, dass (1.6) in der Gruppentheorie ein Kriterium für Untergruppen darstellt, d.h. für eine Gruppe G und eine nichtleere Teilmenge $H \subseteq G$ gilt, dass H genau dann eine Untergruppe von G ist, wenn

$$fH \cap H \neq \emptyset \text{ und } Hf \cap H \neq \emptyset \quad \text{implizieren} \quad f \in H \tag{1.8}$$

für beliebige $f \in G$ gilt. Somit können Codes innerhalb der freien Halbgruppe X^* als Gegenstücke zu Untergruppen betrachtet werden.

Wir beweisen nun das Untergruppenkriterium.

Sei H eine Untergruppe von G. Ferner erfülle $f \in G$ die Beziehung $Hf \cap H \neq \emptyset$. Folglich gilt $h_1f = h_2$ für gewisse $h_1, h_2 \in H$. Somit ergibt sich $f = h_1^{-1}h_2 \in H$, und (1.8) ist bewiesen.

Gilt umgekehrt für eine nichtleere Menge $H \subseteq G$ und alle $f \in G$ die Aussage (1.8), so ergeben sich

- $e \in H$ für das neutrale Element e von G wegen $He \cap H = H \cap H = H \neq \emptyset$ und $eH \cap H \neq \emptyset$,
- $h^{-1} \in H$ für $h \in H$ wegen $h^{-1}h = e \in h^{-1}H \cap H \neq \emptyset$ und $Hh^{-1} \cap H \neq \emptyset$,
- $h_1h_2 \in H$ für $h_1, h_2 \in H$ wegen $h_1^{-1}h_1h_2 = h_2 \in Hh_1h_2 \cap H \neq \emptyset$ und $h_1h_2h_2^{-1} = h_1 \in h_1h_2H \cap H \neq \emptyset$.

Damit sind die Bedingungen des klassischen Kriteriums für Untergruppen erfüllt und H ist als Untergruppe nachgewiesen.

Die bisherigen Charakterisierungen von Codes sind leider nicht effektiv in dem Sinn, dass aus ihnen ein Algorithmus gewonnen werden kann, der entscheidet ob die gegebene Menge ein Code ist. Wir geben nun zwei Kriterien an, die effektiv sind.

Satz 1.6 *Die Menge $C = \{x, y\}$ bestehe aus zwei nichtleeren Wörtern x und y über X. Dann ist C genau dann ein Code, wenn $xy \neq yx$ gilt.*

Beweis. Für einen Code mit den Elementen x und y muss wegen Satz 1.1 die Ungleichheit $xy \neq yx$ gelten.

Um die umgekehrte Implikation zu beweisen, betrachten wir die Menge

$$A = \{\{x, y\} : xy \neq yx,\ \{x, y\} \text{ ist kein Code}\}.$$

Wir haben zu zeigen, dass A leer ist (und werden dies indirekt beweisen).

Dazu nehmen wir an, dass $A \neq \emptyset$ gilt. Wir wählen $\{r, s\} \in A$ so, dass $|rs|$ minimal ausfällt, d.h.

$$|rs| = \min\{|xy| : \{x, y\} \in A\}.$$

Da $\{r, s\}$ kein Code ist, muss es wegen Satz 1.1 ein Wort w geben, dass auf zwei verschiedene Arten als Produkt über $\{r, s\}$ dargestellt werden kann. Wir wählen w minimal unter allen Wörtern mit zwei Darstellungen als Produkt. Dann gilt

$$rxr = sx's \quad \text{für gewisse} \quad x, x' \in \{r, s\}^* \tag{1.9}$$

oder

$$rys = sy'r \quad \text{für gewisse} \quad y, y' \in \{r, s\}^*. \tag{1.10}$$

Da bei $|r| = |s|$ die Menge $\{r, s\}$ ein Blockcode wäre, haben r und s verschiedene Längen. Ohne Beschränkung der Allgemeinheit nehmen wir $|s| > |r|$ an. Sowohl aus (1.9) als auch (1.10) damit folgt

$$s = ur \quad \text{für ein gewisses} \quad u \in X^+.$$

Wegen $\lambda s = s\lambda$ folgt aus der Wahl von r und s, dass r nicht leer ist. Folglich gilt

$$|ur| = |s| < |rs|. \tag{1.11}$$

Falls $ur = ru$ gilt, so erhalten wir $sr = urr = rur = rs$ im Widerspruch zur Wahl von r und s. Folglich gilt

$$ur \neq ru. \tag{1.12}$$

Wegen (1.12), (1.11) und der Minimalität von $|rs|$, muss $\{r, u\}$ ein Code sein.

Andererseits erhalten wir durch Einsetzen von ur für s in (1.9) bzw. (1.10)

$$rxr = urx'ur \quad \text{bzw.} \quad ryur = ury'r$$

mit $x, x', y, y' \in \{r, u\}^*$, woraus wegen Satz 1.1 resultiert, dass $\{r, u\}$ kein Code sein kann. Damit haben wir den gewünschten Widerspruch. □

Die Bedingung aus Satz 1.6 ist sehr einfach zu testen, gilt aber nur für sehr spezielle Codes. Wir wollen nun ein Kriterium angeben, das für beliebige Codes gilt und für endliche Mengen effektiv ist.

Dazu definieren wir für eine nichtleere Sprache C über X induktiv die folgenden Mengen:

$$\begin{aligned} K_0(C) &= C, \\ K_{i+1}(C) &= \{w \in X^+ : yw = x \text{ oder } xw = y \text{ für gewisse } x \in C, y \in K_i(C)\}. \end{aligned}$$

Nach Definition ergibt sich damit $K_{i+1}(C)$ als die Menge aller Suffixe von Wörtern aus C bzw. $K_i(C)$, deren Präfix in $K_i(C)$ bzw. C liegen.

Zur Illustration der Konstruktion geben wir das folgende Beispiel.

Beispiel 1.4 Zuerst betrachten wir die Menge $C_0 = \{a, ab, ba\}$ (siehe Abschnitt 1.1). Es ergeben sich die folgenden Mengen:

$K_0(C_0) = C_0 = \{a, ab, ba\}$ nach Definition.

$K_1(C_0) = \{b\}$, da nur das Produkt aus $a \in C_0 = K_0(C_0)$ mit b ein Element aus $C_0 = K_0(C_0)$ ergibt.

$K_2(C_0) = \{a\}$, denn $b \in K_1(C_0)$ ist nicht als Produkt darstellbar und nur $ba \in C_0$ ergibt sich als Produkt von $b \in K_1(C_0)$ und a.

$K_3(C_0) = \{b\}$ ergibt sich analog zu $K_1(C_0)$.

Daher erhalten wir

$$K_i(C_0) = \begin{cases} \{a, ab, ba\} & \text{für } i = 0 \\ \{a\} & \text{für ungerades } i \geq 1 \\ \{b\} & \text{für gerades } i \geq 1. \end{cases}$$

Für den Code $C_1 = \{a, bb, aab, bab\}$ aus Beispiel 1.1 ergeben sich die Mengen

$K_1(C_1) = \{ab\}$, da aab das einzige Wort aus $C = K_0(C)$ mit einen Präfix aus $C_1 = K_0(C_1)$ ist, wobei der Suffix ab ist,

$K_2(C_1) = \{b\}$, denn $ab \in K_1(C_1)$ hat nur den Präfix $a \in C_1$, wobei b Suffix ist, und kein Element aus C_1 hat den Präfix ab, dem einzigen Wort aus $K_1(C_1)$,

$K_3(C_1) = \{b, ab\}$, da die einzigen zulässigen Zerlegungen von Elementen aus C_1 und $K_2(C_1)$ durch $b \cdot b$ und $b \cdot ab$ gegeben sind,

$K_4(C_1) = \{b, ab\}$ in Analogie zu $K_3(C_1)$. Somit erhalten wir

$$K_i(C) = \begin{cases} \{a, bb, aab, bab\} & \text{für } n = 0, \\ \{ab\} & \text{für } n = 1, \\ \{b\} & \text{für } n = 2, \\ \{b, ab\} & \text{für } n \geq 3. \end{cases}$$

Wir beweisen zuerst zwei Lemmata, die im Wesentlichen einen Schritt (von i nach $i+1$) auf den Übergang von i nach $j > i$ vollziehen. Dazu setzen wir noch für eine Menge U von Wörtern über X

$$Suff(C) = \{w \mid xw = z \in U \text{ für gewisse } x, z\}$$

setzen.

Lemma 1.7 *Für jeden Code C, jedes $n \geq 0$ und jedes $w \in K_n(C)$ gilt $w \in Suff(C)$.*

Beweis. Wir beweisen die Aussage durch vollständige Induktion über n.

$n = 0$. Nach Definition gilt $w \in C$. Wegen $C \subset Suff(C)$ ist der Induktionsanfang gezeigt.

$n \to n+1$. Sei $w \in K_{n+1}(C)$. Gelten $yw = x$, $y \in K_n(C)$ und $x \in C$, so folgt sofort $w \in Suff(x) \subseteq Suff(C)$. Gelten dagegen $xw = y$, $y \in K_n(C)$ und $x \in C$, so folgt mittels Induktionsvoraussetzung $w \in Suff(y) \subseteq Suff(Suff(C)) = Suff(C)$. □

Lemma 1.8 *Ein Wort v_n liegt genau dann in $K_n(C)$, $n \geq 1$, wenn es für jedes $i < n$ Wörter $v_i \in K_i(C)$ und Codewörter $x_{i_1}, x_{i_2}, \ldots, x_{i_k}, x_{j_1}, x_{j_2}, \ldots, x_{j_l} \in C$ mit $k + l = n - i$ derart gibt, dass entweder*

$$v_i x_{i_1} x_{i_2} \ldots x_{i_k} v_n = x_{j_1} x_{j_2} \ldots x_{j_l} \quad \textit{mit} \quad |v_n| < |x_{j_l}|$$

oder

$$v_i x_{i_1} x_{i_2} \ldots x_{i_k} = x_{j_1} x_{j_2} \ldots x_{j_l} v_n \quad \textit{mit} \quad |v_n| < |x_{i_k}| \textit{ für } k \neq 0$$

gilt.

Beweis. Für $n - i = 1$ also $i = n - 1$ entsprechen die Bedingungen gerade der Definition von Wörtern in $K_n(C)$.

Sei $n - i = 2$. Für ein Wort $v_2 \in K_n(C)$ gibt es nach Definition Wörter $v_1 \in K_{n-1}(C)$ und $x' \in C$ derart, dass entweder

$$v_1 v_2 = x' \tag{1.13}$$

oder

$$x' v_2 = v_1 \tag{1.14}$$

gilt. Weiterhin gibt es wegen $v_1 \in K_{n-1}(C)$ Wörter $v_0 \in K_{n-2}(C)$ und $x \in C$ derart, dass entweder

$$v_0 v_1 = x \tag{1.15}$$

oder

$$x v_1 = v_0 \tag{1.16}$$

gültig ist. Wir betrachten nun die vier Kombinationsmöglichkeiten:

Fall 1: 1.13 und 1.15. Wir erhalten einerseits $v_0 v_1 v_2 = v_0 x'$ und andererseits $v_0 v_1 v_2 = x v_2$, woraus mit $v_0 x' = x v_2$ eine Gleichheit der gewünschten Art resultiert.

Fall 2: 1.13 und 1.16. Wir erhalten einerseits $x v_1 v_2 = v_0 v_2$ und andereseits $x v_1 v_2 = x x'$. Hieraus ergibt sich die Gleichheit $v_0 v_2 = x x'$ der gewünschten Art.

Fall 3: 1.14 und 1.15. Wir erhalten $v_0 x' v_2 = v_0 v_1 = x$. Die letzte Gleichheit ist von der geforderten Art.

Fall 4: 1.14 und 1.16. Wir erhalten $v_0 = xv_1 = xx'v_2$, woraus die Gleichheit $v_0 = xx'v_2$ der Art resultiert.

Damit haben wir gezeigt, dass für $v_2 \in K_n(C)$ und $n - i = 2$ Wörter der gewünschten Art existieren.

Für die umgekehrte Richtung sei zuerst $v_0x'v_2 = x$ für $v_0 \in K_{n-2}(C)$ und $x, x' \in C$ gültig. Dann setzen wir $v_1 = x'v_2$. Wegen $v_0v_1 = x$ ist dann $v_1 \in K_{n-1}(C)$ und wegen $v_1 = x'v_2$ folglich $v_2 \in K_n(C)$. Dies entspricht dem obigen Fall 3. In analoger Weise behandeln wir die anderen drei möglichen Fälle $v_0v_2 = xx'$, $v_0x' = xv_2$ und $v_0 = xx'v_2$.

Damit ist die Aussage für $n - i = 2$ beweisen.

Mittels vollständiger Induktion lässt sich die Aussage nun für alle n und i beweisen. □

Wir verwenden nun die Mengen $K_n(C)$ für eine weitere Charakterisierung von (strengen) Codes.

Satz 1.9 *Eine nichtleere Sprache C über X ist genau dann ein Code, wenn $K_i(C) \cap C = \emptyset$ für $i \geq 1$ gilt.*

Beweis. Zuerst zeigen wir (indirekt), dass aus der Gültigkeit von $K_i(C) \cap C = \emptyset$ für alle $i \geq 1$ folgt, dass C ein Code ist.

Nehmen wir dazu an, dass C kein Code wäre. Dann gibt es nach Satz 1.1 Elemente $x_{i_1}, x_{i_2}, \ldots, x_{i_n}, x_{j_1}, x_{j_2}, \ldots, x_{j_m} \in C$ mit

$$x_{i_1}x_{i_2} \ldots x_{i_n} = x_{j_1}x_{j_2} \ldots x_{j_m}$$

und $x_{i_1} \neq x_{j_1}$. Wenn wir $n + m$ minimal wählen, so erhalten wir, dass

$$x_{i_1}x_{i_2} \ldots x_{i_t} \neq x_{j_1}x_{j_2} \ldots x_{j_s}$$

und sogar noch schärfer

$$|x_{i_1}x_{i_2} \ldots x_{i_t}| \neq |x_{j_1}x_{j_2} \ldots x_{j_t}|$$

für beliebige $1 \leq t \leq n$ und $1 \leq s \leq m$ gelten.

Für $1 \leq t \leq n$ sei t' die minimale natürliche Zahl mit

$$|x_{i_1}x_{i_2} \ldots x_{i_t}| < |x_{j_1}x_{j_2} \ldots x_{j_{t'}}|.$$

Offenbar gilt dann

$$x_{j_1}x_{j_2} \ldots x_{j_{t'}} = x_{i_1}x_{i_2} \ldots x_{i_t}v_t$$

für ein gewisses Wort v_t. In völlig analoger Weise definieren wir für $1 \leq s \leq m$ die minimale Zahl s' und z_s mit

$$x_{i_1}x_{i_2} \ldots x_{i_{s'}} = x_{j_1}x_{j_2} \ldots x_{j_s}z_s.$$

Wir zeigen nun mittels Induktion über die Länge von $x_{i_1}x_{i_2} \ldots x_{i_t}$ und $x_{j_1}x_{j_2} \ldots x_{j_s}$, dass die Wörter v_t bzw. z_s in einer der Mengen $K_i(C)$ mit $i \geq 1$ liegen.

Ohne Beschränkung der Allgemeinheit nehmen wir $|x_{i_1}| < |x_{j_1}|$ an. Dann gilt $x_{i_1}v_1 = x_{j_1}$. Wegen $x_{i_1}, x_{j_1} \in C = K_0(C)$, ergibt sich, dass v_1 in $K_1(C)$ liegt, womit der Induktionsanfang bewiesen ist.

Wir betrachten nun v_t. Wir unterscheiden zwei Fälle:

Fall 1. $|x_{i_1}x_{i_2}\dots x_{i_{t-1}}| < |x_{j_1}x_{j_2}\dots x_{j_{t'-1}}|$. Aufgrund der Minimalität von t' gilt

$$|x_{j_1}x_{j_2}\dots x_{j_{t'-1}}| < |x_{i_1}x_{i_2}\dots x_{i_t}|\,.$$

Damit ist $x_{i_1}x_{i_2}\dots x_{i_t}$ das Wort minimaler Länge, das länger als $x_{j_1}x_{j_2}\dots x_{j_{t'-1}}$ ist. Folglich erfüllt $z_{t'-1}$ folgende Gleichungen:

$$x_{j_1}x_{j_2}\dots x_{j_{t'-1}}z_{t'-1} = x_{i_1}x_{i_2}\dots x_{i_t} \tag{1.17}$$

und

$$z_{t'-1}v_t = x_{j_{t'}}. \tag{1.18}$$

Nach (1.17 und Lemma 1.8 gilt $z_{t'-1} \in K_i(C)$ für ein gewisses $i \geq 1$. Damit ergibt sich aus (1.18) dann $v_t \in K_{i+1}(C)$, womit die Induktionsbehauptung gezeigt ist.

Fall 2. $|x_{i_1}x_{i_2}\dots x_{i_{t-1}}| > |x_{j_1}x_{j_2}\dots x_{j_{t'-1}}|$. Dann gilt $(t-1)' = t'$ und folglich

$$x_{i_{t-1}}v_t = v_{t-1}.$$

Da nach Induktionsbehauptung $v_{t-1} \in K_i(C)$ für ein gewisses $i \geq 1$ ist und $x_{i_{t-1}} \in C$ gilt, erhalten wir $v_t \in K_{i+1}(C)$, womit auch in diesem Fall die Induktionsbehauptung bewiesen ist.

Analog beweist man, dass jedes z_s in einer der Mengen aus $K(C)$ liegt.

Ohne Beschränkung der Allgemeinheit sei $|x_{i_n}| < |x_{j_m}|$. Dann folgt $(n-1)' = m$ und $v_{n-1} = x_{j_n}$. Wegen $v_{n-1} \in K_j(C)$ für ein gewisses $j \geq 1$ und $x_{j_n} \in C$ erhalten wir, dass es ein $j \geq 1$ gibt, für das $K_j(C) \cap C$ nicht leer ist. Dies widerspricht unserer Voraussetzung.

Sei nun C ein Code. Wir zeigen zuerst, dass

$$C^*w \cap C^* \neq \emptyset \quad \text{für} \quad w \in K_i(C),\ i \geq 0 \tag{1.19}$$

gilt.

Für $i = 0$ folgt dies direkt aus der Definition von $K_0(C) = C$.

Sei nun $w \in K_i(C)$. Dann gibt es $x \in C$ und $y \in K_{i-1}(C)$ derart, dass

$$xw = y \quad \text{oder} \quad yw = x$$

gilt. Ferner ist nach Induktionsannahme

$$x_1y = x_2$$

für gewisse $x_1, x_2 \in C^*$ gültig. Damit erhalten wir

$$x_1xw = x_1y = x_2 \quad \text{oder} \quad x_1x = x_1yw = x_2w$$

und damit in beiden Fällen (1.19).

Wir nehmen nun an, dass $K_i(C) \cap C$ für ein $i \geq 1$ nicht leer ist. Sei $x \in K_i(C) \cap C$. Dann gibt es nach Definition von $K_i(C)$ Elemente $z \in C$ und $y \in K_{i-1}(C)$ mit

$$yx = z \quad \text{oder} \quad zx = y. \tag{1.20}$$

Gilt $yx = z$, so folgt $yC^* \cap C^* \neq \emptyset$. Außerdem haben wir $C^*y \cap C^* \neq \emptyset$ wegen (1.19). Damit folgt aus Satz 1.5 (beachte, dass jeder Code produktunabhängig ist), dass y in C^* liegt, d.h. es gilt $y = y_1y_2 \dots y_l$ für gewisse $y_k \in C$, $1 \leq k \leq l$. Somit erhalten wir

$$yx = y_1y_2 \dots y_lx = z$$

mit $y_1 \neq z$. Dies ist wegen Satz 1.1 ein Widerspruch zur Voraussetzung, dass C ein Code ist.

Folglich muss von den Gleichheiten in (1.20) die zweite gelten, d.h. $zx = y$. Wir bemerken zuerst, dass dann $i \geq 2$ gilt, denn für $i = 1$ würde $y \in K_0(C) = C$ als Produkt von Elementen $z \in C$ und $x \in C$ darstellbar sein, womit sich erneut ein Widerspruch zu Satz 1.1 ergeben würde. Daher gibt es nach Definition von $K_{i-1}(C)$ Elemente $y_1 \in K_{i-2}(C)$ und $z_1 \in C$ so, dass

$$y_1y = z_1 \quad \text{oder} \quad z_1y = y_1$$

gilt. Die erste dieser Möglichkeiten führt zu $y_1zx = z_1$ und dann analog zur ersten der beiden Gleichheiten in (1.20) zu einem Widerspruch. Aus der zweiten möglichen Gleichheit erhalten wir

$$z_1zx = z_1y = y_1.$$

Ist $i = 2$, so ergibt sich wegen $y_1 \in K_{i-2}(C) = K_0(C) = C$ aus der vorstehenden Gleichheit ein Widerspruch zu Satz 1.1. Damit muss $i \geq 3$ gelten.

Analog fahren wir fort und erhalten, dass $i \geq n_0$ für jede Schranke n_0 gilt. Das bedeutet aber gerade, dass für alle $i \geq 1$ die Menge $K_i(C) \cap C$ leer ist, was zu zeigen war. □

Satz 1.10 *Eine nichtleere endliche Sprache C über X ist genau dann ein strenger Code, wenn $K_n(C) = \emptyset$ für $n \geq card(C)(\max\{|c| : c \in C\} - 1) + 1$ gilt.*

Beweis. Hinlänglichkeit: Wir setzen

$$m = card(C)(\max\{|c| : c \in C\} - 1) + 1\,.$$

Wir nehmen an, dass $K_n(C) \neq \emptyset$ für ein $n \geq m$ gilt. Dann ist auch $K_m(C) \neq \emptyset$. Sei nun $v_m \in K_m(C)$. Dann gibt es ein $v_{m-1} \in K_{m-1}(C)$ derart, dass $xv_m = v_{m-1}$ oder $v_{m-1}v_m = x$ für ein $x \in C$ gilt. Zu v_{m-1} gibt es wieder ein $v_{m-2} \in K_{m-2}(C)$ usw. Sei $v_0, v_1, \dots, v_{m-1}, v_m$ die so erzeugte Folge von Elementen. Nach Lemma 1.7 liegt jedes Element in $Suff(C)$. Da jedes Wort x aus C höchstens $|x|$ verschiedene nichtleere Suffixe hat, gibt es insgesamt höchstens m nichtleere Wörter in $Suff(C)$. Daher müssen zwei Wörter der Folge $v_0, v_1, \dots v_m$ gleich sein. Sei $v_{h_1} = v_{h_2}$.

Mit Blick auf den Beweis von Lemma 1.8 erkennen wir, dass v_i gerade dass nach Lemma 1.8 existierende Element aus $K_i(C)$ ist.Folglich erhalten wir aus Lemma 1.8, dass entweder

$$v_{h_1}x_{i_1}x_{i_2} \dots x_{i_k}v_{h_2} = x_{j_1}x_{j_2} \dots x_{j_l} \text{ mit } k \geq 1$$

oder

$$v_{h_1}x_{i_1}x_{i_2} \dots x_{i_k} = x_{j_1}x_{j_2} \dots x_{j_l}v_{h_2} \text{ mit } k \geq 1,\ l \geq 1$$

und damit unter Verwendung von $v = v_{h_1} = v_{h_2}$

$$vx_{i_1}x_{i_2} \dots x_{i_k}v = x_{j_1}x_{j_2} \dots x_{j_l} \text{ mit } k \geq 1 \tag{1.21}$$

oder

$$vx_{i_1}x_{i_2}\dots x_{i_k} = x_{j_1}x_{j_2}\dots x_{j_l}v \text{ mit } k \geq 1,\ l \geq 1\,. \tag{1.22}$$

Im ersten Fall erhalten wir durch Multiplikation mit $x_{i_1}x_{i_2}\dots x_{i_k}v$

$$vx_{i_1}x_{i_2}\dots x_{i_k}vx_{i_1}x_{i_2}\dots x_{i_k}v = x_{j_1}x_{j_2}\dots x_{j_l}x_{i_1}x_{i_2}\dots x_{i_k}v$$

und

$$vx_{i_1}x_{i_2}\dots x_{i_k}vx_{i_1}x_{i_2}\dots x_{i_k}v = vx_{i_1}x_{i_2}\dots x_{i_k}x_{j_1}x_{j_2}\dots x_{j_l}\,,$$

woraus

$$vx_{i_1}x_{i_2}\dots x_{i_k}x_{j_1}x_{j_2}\dots x_{j_l} = x_{j_1}x_{j_2}\dots x_{j_l}x_{i_1}x_{i_2}\dots x_{i_k}v\,,$$

d.h. eine Gleichheit der Art aus (1.22). Daher brauchen wir im Folgenden nur noch (1.22) diskutieren.

Durch wiederholte Anwendung von 1.22 erhalten wir

$$\begin{aligned} v(x_{i_1}x_{i_2}\dots x_{i_k})^r &= vx_{i_1}x_{i_2}\dots x_{i_k}(x_{i_1}x_{i_2}\dots x_{i_k})^{r-1} \\ &= x_{j_1}x_{j_2}\dots x_{j_l}v(x_{i_1}x_{i_2}\dots x_{i_k})^{r-1} \\ &= (x_{j_1}x_{j_2}\dots x_{j_l})^2v(x_{i_1}x_{i_2}\dots x_{i_k})^{r-2} \\ &\vdots \quad \vdots \\ &= (x_{j_1}x_{j_2}\dots x_{j_l})^r v\,. \end{aligned} \tag{1.23}$$

Außerdem gilt nach Lemma 1.8 unter Beachtung von $v = v_{h_1} \in K_{h_1}(C)$

$$v_0y_{f_1}y_{f_2}\dots y_{f_s}v = y_{g_1}y_{g_2}\dots y_{g_t} \text{ oder } v_0y_{f_1}y_{f_2}\dots y_{f_s} = y_{g_1}y_{g_2}\dots y_{g_t}v$$

für gewisse Codewörter $y_{f_1}, y_{f_2}, \dots, y_{f_s}, y_{g_1}, y_{g_2}, \dots, y_{g_t}$. Unter Berücksichtigung von $v_0 \in K_0(C) = C$ ergibt sich

$$y_{p_1}y_{p_2}\dots y_{p_u}v = y_{q_1}y_{q_2}\dots y_{q_w} \tag{1.24}$$

für gewisse Codewörter $y_{p_1}, y_{p_2}, \dots y_{p_u}, y_{q_1}, y_{q_2}, \dots, y_{q_w}$. Hierbei können wir o.B.d.A. annehmen, dass $y_{p_1} \neq y_{q-1}$ gilt. Aus den Gleichheiten (1.23) und (1.24) erhalten wir

$$y_{p_1}y_{p_2}\dots y_{p_u}v(x_{i_1}x_{i_2}\dots x_{i_k})^r = y_{p_1}y_{p_2}\dots y_{p_u}(x_{j_1}x_{j_2}\dots x_{j_l})^r v$$

und

$$y_{p_1}y_{p_2}\dots y_{p_u}v(x_{i_1}x_{i_2}\dots x_{i_k})^r = y_{q_1}y_{q_2}\dots y_{q_w}(x_{i_1}x_{i_2}\dots x_{i_k})^r$$

und damit

$$y_{p_1}y_{p_2}\dots y_{p_u}(x_{j_1}x_{j_2}\dots x_{j_l})^r v = y_{q_1}y_{q_2}\dots y_{q_w}(x_{i_1}x_{i_2}\dots x_{i_k})^r\,.$$

Da diese Gleichung für beliebiges $r \geq 1$ gilt und $y_{p_1} \neq y_{q-1}$ ist, erhalten wir einen Widerspruch dazu, dass C ein strenger Code ist.

Notwendigkeit: Wir nehmen an, dass C kein strenger Code ist. Dann gibt es Codewörter x_{i_k} und x_{j_k}, $k \geq 1$, so dass

$$x_{i_1}x_{i_2}\dots = x_{j_1}x_{j_2}\dots \quad \text{mit} \quad i_1 \neq j_1$$

gilt. Folglich besteht für beliebiges k eine Gleichheit

$$x_{i_1}x_{i_2}\dots x_{i_k}v_k = x_{j_1}x_{j_2}\dots x_{j_{l(k)}} \quad \text{mit} \quad |v_k| < |x_{j_{l(k)}}\,.$$

Aus Lemma 1.8 (mit $v_0 = x_{i_1}$ oder $v_0 = x_{j_1}$) folgt nun $v_k \in K_{k+l(k)-1}(C)$. Da k beliebig groß werden kann, ist $K_m(C) \neq \emptyset$ und damit ein Widerspruch zur Voraussetzung hergeleitet. □

Sei C eine endliche Sprache. Wir setzen

$$k = \max\{|v| : v \in C\}.$$

Dann folgt aus Lemma 1.7, dass jede Menge $K_i(C)$, $i \geq 0$, eine Teilmenge aller Wörter der Länge $l \leq k$ ist. Folglich ist die Menge

$$K(C) = \{K_i(C) : i \geq 0\}$$

endlich. Daher gibt es Zahlen i und j mit $i < j$ und $K_i(C) = K_j(C)$. Es ist leicht zu sehen, dass dann $K_{j+k}(C) = K_{i+k}(C)$ für $k \geq 0$ gilt. Wählen wir j minimal mit dieser Eigenschaft, so gilt

$$K(C) = \{K_0(C), K_1(C), \dots, K_{j-1}(C)\}.$$

Hieraus folgt, dass es für endliche Sprachen C über endlichen Alphabeten einen Algorithmus gibt, der die Menge $K(C)$ bestimmt. Wir ermitteln einfach der Reihe nach die Mengen $K_0(C), K_1(C), K_2(C), ...$ und brechen ab, wenn wir eine Menge $K_j(C)$ erhalten, die bereits unter den Mengen $K_i(C)$ mit $i < j$ vorkommt.

Da die Menge aller Wörter der Länge l über einem m-elementigen Alphabet X aus m^l Wörtern besteht, woraus sich ergibt, dass das minimale j mit obiger Eigenschaft

$$j \leq 2^{m+m^2+\dots m^k} = 2^{\frac{m^{k+1}-1}{m-1}-1}$$

erfüllt.

Aus Satz 1.9 und Satz 1.10 ergibt sich sofort der folgende Satz.

Satz 1.11 *Es gibt einen Algorithmus, der für jede endliche Sprache C über dem endlichen Alphabet X entscheidet, ob C ein (strenger) Code ist.*

Beweis. Unter den gegebenen Voraussetzungen kann — wie oben gezeigt — $K(C)$ algorithmisch bestimmt werden. Wir haben nun nur noch zu testen, ob für die endlich vielen Mengen $K_i(C)$ mit $i \geq 1$ auch $K_i(C) \cap C$ leer ist. Fällt dieser Test positiv aus, so ist C entsprechend Satz 1.9 ein Code; andernfalls ist C kein Code.

Um nachzuweisen, ob C ein strenger Code ist, berechnen wir die Menge $K_m(C)$ für $m = card(C)(\max\{|c| : c \in C\})+1$ (was wegen der Periodizität der Mengen $K_i(C)$, $i \geq 1$, nach Obigem möglich ist) und testen, ob $K_m(C)$ die leere Menge ist. □

Beispiel 1.5 Wenden wir den Algorithmus auf

$$C_0 = \{a, ab, ba\} \quad \text{und} \quad C_1 = \{a, bb, aab, bab\}$$

aus Beispiel 1.4 an, so erhalten wir wegen der in Beispiel 1.4 jeweils bestimmten Mengen $K_i(C_0)$ und $K_i(C_1)$, dass C_0 kein Code ist, während C_1 ein Code ist. Außerdem ist C_1 kein strenger Code.

1.4 Codeindikator und Konstruktion von Codes

Entsprechend den bisher angegebenen Resultaten können wir für eine gegebene Menge feststellen, ob sie ein Code ist. Es bleibt aber noch die Aufgabe, einen Algorithmus zu finden, der einen Code konstruiert. Ein solches Verfahren wird sich aus der nachfolgenden hinreichenden Charakterisierung von Codes durch den Codeindikator ergeben.

Definition 1.7 *Sei X ein Alphabet der Kardinalität $n \geq 2$. Der Codeindikator $ci(w)$ eines Wortes $w \in X^*$ ist durch*

$$ci(w) = n^{-|w|}$$

definiert. Für eine Sprache L mit $X = \min(L)$[2] setzen wir

$$ci(L) = \sum_{w \in L} ci(w).$$

Beispiel 1.6 Für die zu Beginn eingeführte Menge C_0, die Mengen C_1 und C_3 aus Beispiel 1.1 und die Menge $C_5 = \{a, ba, bb, aab\}$ ergibt sich folgende Tabelle.

$$\begin{array}{ll} C_0 = \{a, ab, ba\} & ci(C_0) = \frac{1}{2} + \frac{1}{4} + \frac{1}{4} = 1 \\ C_1 = \{a, bb, aab, bab\} & ci(C_1) = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8} = 1 \\ C_3 = \{aaa, aba, bab, bbb\} & ci(C_3) = \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} = \frac{1}{2} \\ C_5 = \{a, ba, bb, aab\} & ci(C_4) = \frac{1}{2} + \frac{1}{4} + \frac{1}{4} + \frac{1}{8} = \frac{9}{8} \end{array}$$

Satz 1.12 *Seien L_1 und L_2 zwei Sprachen über dem Alphabet X, das aus n Buchstaben besteht. Dann gilt*

$$ci(L_1 \cdot L_2) \leq ci(L_1) \cdot ci(L_2),$$

und die Gleichheit tritt genau dann ein, wenn für je vier Wörter $w_1, w_2 \in L_1$ und $w_3, w_4 \in L_2$ aus $w_1 w_3 = w_2 w_4$ folgt, dass $w_1 = w_2$ gilt.

Beweis. Die behauptete Ungleichung ergibt sich aus

$$\begin{aligned} ci(L_1 \cdot L_2) &= ci(\{v_1 v_2 : v_1 \in L_1,\ v_2 \in L_2\}) \\ &\leq \sum_{v_1 \in L_1, v_2 \in L_2} n^{-|v_1 v_2|} \\ &= \sum_{v_1 \in L_1, v_2 \in L_2} n^{-(|v_1| + |v_2|)} \\ &= \sum_{v_1 \in L_1, v_2 \in L_2} n^{-|v_1|} \cdot n^{-|v_2|} \\ &= \sum_{v_1 \in L_1} n^{-|v_1|} \cdot \sum_{v_2 \in L_2} n^{-|v_2|} \\ &= ci(L_1) \cdot ci(L_2). \end{aligned}$$

Dabei gilt die Gleichheit genau dann, wenn kein Element w aus $L_1 L_2$ zwei verschiedene Darstellungen als Produkt besitzt, da w dann bei der Berechnung des Codeindikators von $L_1 L_2$ nur einmal betrachtet wird, während bei der Berechnung von $ci(L_1)ci(L_2)$ beide Darstellungen berücksichtigt werden. □

Wir geben nun den Satz an, der die Bezeichnung Codeindikator rechtfertigt.

[2] Mit $\min(L)$ bezeichnen wir das bezüglich der Inklusion kleinste Alphabet X mit $L \subseteq X^*$.

Satz 1.13 *Für jeden Code C gilt $ci(C) \leq 1$.*

Beweis. Wir zeigen zuerst mittels Induktion, dass

$$ci(C^i) = (ci(C))^i$$

gültig ist.

Für $i = 1$ ist dies offensichtlich.

Sei $w \in C^i$. Dann gibt es wegen Satz 1.2 genau eine Darstellung von w als Produkt von i Elementen aus C. Insbesondere ist damit w in genau einer Weise als Produkt von einem Element aus C und einem Element aus C^{i-1} darstellbar. Somit erhalten wir aus Satz 1.12 und der Induktionsvoraussetzung

$$ci(C^i) = ci(C \cdot C^{i-1}) = ci(C) \cdot ci(C^{i-1}) = ci(C) \cdot (ci(C))^{i-1} = (ci(C))^i.$$

Seien nun

$$k = \min\{|w| : w \in C\} \quad \text{und} \quad K = \max\{|w| : w \in C\},$$

so gilt für jedes Wort $v \in C^j$, $j \geq 1$,

$$|v| \geq jk \quad \text{und} \quad |v| \leq jK.$$

Hieraus folgt

$$\begin{aligned} ci(C^j) &= \sum_{i=jk}^{jK} \sum_{v \in C^j, |v|=i} n^{-|v|} \\ &\leq \sum_{i=jk}^{jK} \sum_{|v|=i} n^{-|v|} \\ &= \sum_{i=jk}^{jK} 1 = jK - jk + 1 = (K-k)j + 1. \end{aligned}$$

Gilt nun $ci(C) > 1$, so wächst $(ci(C))^j$ exponentiell in j, und somit gibt es eine natürliche Zahl j mit

$$ci(C^j) = (ci(C))^j > (K-k)j + 1,$$

womit wir einen Widerspruch zur Ungleichung davor erhalten. Daher muss $ci(C) \leq 1$ erfüllt sein. □

Die Umkehrung von Satz 1.13 gilt nicht, denn die Menge C_0 ist kein Code, wie zu Beginn dieses Kapitels nachgewiesen wurde, erfüllt aber die Bedingung $ci(C_0) \leq 1$ (siehe Beispiel 1.6).

Aus Satz 1.13 folgt aber sofort, dass C_5 aus Beispiel 1.6 kein Code ist.

Satz 1.14 *Seien $n \geq 2$ und $l_1, l_2, \ldots, l_m$, $m \geq 1$, natürliche positive Zahlen, die der Bedingung*

$$\sum_{i=1}^{m} n^{-l_i} \leq 1$$

genügen. Dann gibt es einen Code (Präfixcode)

$$C = \{c_0, c_1, \ldots, c_{m-1}\}$$

über dem n-elementigen Alphabet X mit

$$|c_{i-1}| = l_i \quad \textit{für} \quad 1 \leq i \leq m.$$

Beweis. Ohne Beschränkung der Allgemeinheit nehmen wir an, dass die gegebenen Zahlen geordnet sind, d.h.

$$l_1 \leq l_2 \leq \ldots \leq l_m.$$

Wir definieren zuerst die Zahlen q_i, $0 \leq i \leq m-1$, wie folgt:

$$\begin{aligned} q_0 &= 0, \\ q_i &= q_{i-1} + n^{-l_i} = \sum_{j=1}^{i} n^{-l_j} \quad \text{für} \quad 1 \leq i \leq m-1. \end{aligned}$$

Für $0 \leq i \leq m-1$ gelten dann folgende Aussagen:

- $0 \leq q_i = \sum_{j=1}^{i} n^{-l_j} < \sum_{j=1}^{m} n^{l_j} \leq 1$.
- Die n-äre Darstellung von q_i besitzt offensichtlich höchstens l_i Stellen hinter dem Komma (da n^{-l_i} der minimale Wert der Summanden bei der Bildung von q_i ist).

Sei nun $0, a_i^{(1)} a_i^{(2)} \ldots a_i^{(l_{i+1})}$ die n-äre Darstellung von q_i (sollte $l_{i+1} > l_i$ sein, so ergänzen wir am Ende einige Nullen). Dann setzen wir

$$c_i = a_i^{(1)} a_i^{(2)} \ldots a_i^{(l_{i+1})}$$

Offenbar gilt damit $|c_i| = l_{i+1}$.

Wir zeigen, dass

$$C = \{c_0, c_1, \ldots, c_{m-1}\}$$

ein Präfixcode ist. Dazu nehmen wir an, dass c_r Präfix von c_s für gewisse r und s gilt, und führen dies zu einem Widerspruch. Sei

$$c_s = c_r b_r^{(l_{r+1}+1)} b_r^{(l_r+2)} \ldots b_r^{(l_{s+1})}$$

für ein gewisses k. Dann folgt

$$q_s = q_r + \sum_{j=1}^{l_{s+1}-l_{r+1}} b_r^{(l_{r+1}+j)} n^{-(l_{r+1}+j)} < q_r + n^{-l_{r+1}}.$$

Andererseits gilt

$$q_s \geq q_{r+1} = q_r + n^{-l_{r+1}}.$$

Die beiden letzten Ungleichungen widersprechen sich. □

Wir illustrieren die im Beweis von Satz 1.14 gegebene Methode durch ein Beispiel.

Beispiel 1.7 Wir setzen $n = 2$ und $X = \{0, 1\}$. Ferner sei

$$l_1 = l_2 = l_3 = 3, \quad l_4 = l_5 = l_6 = l_7 = 4.$$

Dann ist die Bedingung

$$\sum_{i=1}^{7} 2^{-l_i} = \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{16} + \frac{1}{16} + \frac{1}{16} + \frac{1}{16} = \frac{5}{8} \leq 1$$

erfüllt. Entsprechend dem Beweis von Satz 1.14 ergeben sich die Werte der folgenden Tabelle:

i	l_i	q_{i-1}	Dualdarst. von q_{i-1}	c_{i-1}
1	3	$q_0 = 0$	0,000	000
2	3	$q_1 = q_0 + \frac{1}{8} = 0 + \frac{1}{8} = \frac{1}{8}$	0,001	001
3	3	$q_2 = q_1 + \frac{1}{8} = \frac{1}{8} + \frac{1}{8} = \frac{1}{4}$	0,010	010
4	4	$q_3 = q_2 + \frac{1}{8} = \frac{1}{4} + \frac{1}{8} = \frac{3}{8}$	0,0110	0110
5	4	$q_4 = q_3 + \frac{1}{16} = \frac{3}{8} + \frac{1}{16} = \frac{7}{16}$	0,0111	0111
6	4	$q_5 = q_4 + \frac{1}{16} = \frac{7}{16} + \frac{1}{16} = \frac{1}{2}$	0,1000	1000
7	4	$q_6 = q_5 + \frac{1}{16} = \frac{1}{2} + \frac{1}{16} = \frac{9}{16}$	0,1001	1001

Daher ergibt sich für die vorgegebenen Längen der Präfixcode

$$C = \{000, 001, 010, 0110, 0111, 1000, 1001\}.$$

Definition 1.8 *Ein Code C heißt maximal, wenn für jedes Wort $w \notin C$ die Menge $C \cup \{w\}$ kein Code ist.*

Zu einem maximalen Code lässt sich nach Definition 1.8 kein Wort hinzufügen, ohne dass die Eigenschaft, Code zu sein, verloren geht.

Satz 1.15 *Ein Code C mit $ci(C) = 1$ ist ein maximaler Code.*

Beweis. Sei $ci(C) = 1$ für einen Code C. Wenn C nicht maximal ist, gibt es ein Wort $w \notin C$ derart, dass auch $C' = C \cup \{w\}$ ein Code ist. Wegen

$$ci(C') = ci(C) + ci(w) > 1$$

ergibt sich ein Widerspruch zu Satz 1.13. □

Der folgende Satz, den wir ohne Beweis angeben, verschärft Satz 1.15 für endliche Codes.

Satz 1.16 *Ein endlicher Code C ist genau dann maximal, wenn $ci(C) = 1$ gilt.* □

Kapitel 2

Optimale Codes

Im vorhergehenden Abschnitt haben wir eine Methode angegeben, mittels derer entschieden werden kann, ob eine vorgebene Menge von Wörtern ein Code ist, bzw. mittels derer ein Code konstruiert werden kann. Dabei wurde aber nicht darauf geachtet, ob der erzeugte Code noch weitere Eigenschaft hat, die aus praktischen oder theoretischen Gründen für eine Codierung wichtig sein können. In diesem und dem folgenden Abschnitt werden wir daher klären, ob Codes mit gewissen Eigenschaften existieren und wie diese dann erzeugt werden können.

Als erstes sind wir daran interessiert, Codes zu konstruieren, bei denen die zu übermittelnde Nachricht nach der Codierung im Durchschnitt möglichst kurz ist.

Sei eine Quelle gegeben, die in zufälliger Weise nacheinander Buchstaben eines Alphabets $A = \{a_1, a_2, \ldots, a_m\}$ erzeugt, wobei das Erzeugen eines einzelnen Buchstaben unabhängig von den bereits erzeugten Buchstaben erfolgt und der Buchstabe a_i, $1 \leq i \leq m$, mit der Wahrscheinlichkeit p_i erzeugt wird. Dann müssen $p_i \geq 0$ für $1 \leq i \leq m$ und $\sum_{i=1}^{m} p_i = 1$ gelten. Der dadurch erzeugte Text $T = a_{i_1} a_{i_2} \ldots a_{i_n}$ der Länge n werde entsprechend der Codierung $\phi : A = \{a_1, a_2, \ldots, a_n\} \to C \subseteq X^+$ vermöge $\phi(a_i) = c_i$, $1 \leq i \leq n$, codiert. Für die Länge des Textes nach der Codierung ergibt sich

$$|\phi^*(T)| = |c_{i_1}||c_{i_2}| \ldots |c_{i_n}| = \sum_{i=1}^{m} \#_{a_i}(T) \cdot |c_i|.$$

Wenn der Text T hinreichend lang ist, können wir annehmen, dass jeder Buchstabe a_i, $1 \leq i \leq m$, entsprechend seiner Wahrscheinlichkeit p_i in T vorkommt, d.h. es gilt $\#_{a_i}(T) = p_i \cdot |T| = p_i \cdot n$. Folglich ergibt sich

$$|\phi^*(T)| = \sum_{i=1}^{m} p_i \cdot n \cdot |c_i| = n \cdot \sum_{i=1}^{m} p_i \cdot |c_i|.$$

Da n durch $|T|$ festgelegt ist, können wir die Länge von $\phi^*(T)$ nur dadurch optimieren, dass wir ϕ so wählen, dass $\sum_{i=1}^{m} p_i |c_i|$ möglichst klein wird.

Die folgenden Definitionen formalisieren die vorstehenden Überlegungen.

Definition 2.1 *i) Für einen Code $C = \{c_1, c_2, \ldots, c_m\}$ und eine Wahrscheinlichkeitsverteilung $P = \{p_1, p_2, \ldots, p_m\}$, $p_i \geq 0$ für $1 \leq i \leq m$, $\sum_{i=1}^{m} p_i = 1$, definieren wir die Kosten von C unter P durch*

$$\mathcal{L}(C, P) = \sum_{i=1}^{m} p_i |c_i|.$$

ii) Für eine Wahrscheinlichkeitsverteilung $P = \{p_1, p_2, \ldots, p_m\}$, $p_i \geq 0$ *für* $1 \leq i \leq m$, $\sum_{i=1}^{m} p_i = 1$, *und ein Alphabet* X *setzen wir*

$$\mathcal{L}_X(P) = \inf \mathcal{L}(C, P),$$

wobei das Infimum über alle m*-elementigen Codes über* X *zu nehmen ist. Ein Code* C' *über* X *heißt optimal für* P*, wenn*

$$\mathcal{L}(C', P) = \mathcal{L}_X(P)$$

gilt.

Nach der Definition von $\mathcal{L}_X(P)$ haben wir das Infimum über alle m-elementigen Codes über X zu nehmen. Es reicht aber das Infimum über alle m-elementigen Präfixcodes zu nehmen. Dies folgt daraus, dass die Größe $\mathcal{L}_X(P)$ nur von den Längen der Codewörter abhängt (da die Wahrscheinlichkeiten als fest gegeben angesehen werden können) und zu vorgegebenen Längen der Codewörter stets ein Präfixcode konstruiert werden kann (siehe Beweis von Satz 1.14). Aus gleichem Grund ist auch die Existenz eines für die Verteilung P optimalen Präfixcodes gesichert, falls es einen für P optimalen Code gibt.

Im Folgenden nehmen wir stets an, dass alle Wahrscheinlichkeiten p_i, $1 \leq i \leq m$, der Verteilung P positiv sind. [1]

Wir zeigen zuerst, dass unter dieser Voraussetzung für jede Verteilung und jedes Alphabet ein optimaler Code existiert.

Satz 2.1 *Für jede Verteilung* P*, deren Wahrscheinlichkeiten alle positiv sind, und jedes Alphabet* X *existiert ein (Präfix)-Code über* X*, der optimal für* P *ist.*

Beweis. Es habe P genau $m \geq 2$ und X genau $n \geq 2$ Elemente. Dann setzen wir

$$l_i = \lceil m + \log_n(m) \rceil \quad \text{für} \quad 1 \leq i \leq m.$$

Offensichtlich gilt dann

$$\sum_{i=1}^{m} n^{-l_i} \leq \sum_{i=1}^{m} n^{-m-\log_n(m)} = m \cdot n^{-m-\log_n(m)} = m \cdot \frac{n^{-m}}{n^{\log_n(m)}} = m \cdot \frac{n^{-m}}{m} = n^{-m} < 1.$$

Entsprechend Satz 1.14 können wir nun einen Präfixcode $C = \{c_1, c_2, \ldots, c_m\}$ mit $|c_i| = l_i$ für $1 \leq i \leq m$ erzeugen.

Sei p die minimale der Wahrscheinlichkeiten von P. Dann setzen wir

$$k = \frac{\mathcal{L}(C, P)}{p}.$$

Besitzt ein Code C' ein Wort einer Länge $l > k$, so gilt für seine Kosten

$$\mathcal{L}(C', P) \geq p \cdot l > p \cdot k = p \cdot \frac{\mathcal{L}(C, P)}{p} = \mathcal{L}(C, P).$$

[1] Die meisten der folgenden Betrachtungen gelten auch für den Fall, dass einige der Wahrscheinlichkeiten 0 sind, jedoch verkomplizieren sich sich dann die Beweise. Andererseits kann auf Buchstaben, die mit der Wahrscheinlichkeit 0 vorkommen, in der Praxis meist verzichtet werden.

Folglich kann C' nicht optimal für P sein, da seine Kosten die von C übersteigen. Daher muss ein optimaler Code für P nach den obigen Ausführungen ein (Präfix)-Code sein, der aus m Wörtern mit einer Länge $\leq k$ besteht. Offensichtlich gibt es nur eine endliche Anzahl von Mengen, die aus m Wörtern der Länge $\leq k$ bestehen. Unter diesen bestimmen wir alle (Präfix)-Codes und erhalten einen optimalen Code durch Minimierung der Kosten über dieser endlichen Menge. □

Diese Methode ist aber sehr aufwendig, da wir maximal $\binom{n^k}{m}$ Mengen betrachten müssen. Daher sind auch Verfahren von Interesse, mittels deren kostengünstige, aber unter Umständen nicht optimale Codes gewonnen werden können. Deshalb geben wir jetzt eine Abschätzung für die Größe $\mathcal{L}_X(P)$ an.

Satz 2.2 *Für jede Verteilung $P = \{p_1, p_2, \ldots, p_m\}$ und jedes Alphabet X mit $card(X) = n$ gilt*

$$\sum_{i=1}^{m} p_i \log_n \left(\frac{1}{p_i}\right) \leq \mathcal{L}_X(P) \leq 1 + \sum_{i=1}^{m} p_i \log_n \left(\frac{1}{p_i}\right),$$

wobei die Gleichheit

$$\mathcal{L}_X(P) = \sum_{i=1}^{m} p_i \log_n \left(\frac{1}{p_i}\right)$$

genau dann gilt, wenn $\log_n(p_i)$ für $1 \leq i \leq m$ ganze Zahlen sind.

Beweis. Es sei $C = \{c_1, c_2, \ldots, c_m\}$ ein beliebiger Code über X. Mit l_i bezeichnen wir die Länge des Wortes c_i, $1 \leq i \leq m$. Unter Beachtung der üblichen Regeln für das Rechnen mit Logarithmen, den Beziehungen $\log_n(x) = \log_n(e)\ln(x)$ (wobei ln den Logarithmus zur Basis e bezeichnet), $\ln(x) \leq x - 1$ und Satz 1.13 erhalten wir

$$\begin{aligned}
\sum_{i=1}^{m} p_i \log_n \left(\frac{1}{p_i}\right) - \sum_{i=1}^{m} p_i l_i &= \sum_{i=1}^{m} p_i (\log_n \left(\frac{1}{p_i}\right) - l_i) = \sum_{i=1}^{m} p_i (\log_n \left(\frac{1}{p_i}\right) - \log_n(n^{l_i})) \\
&= \sum_{i=1}^{m} p_i \log_n \left(\frac{n^{-l_i}}{p_i}\right) = \sum_{i=1}^{m} p_i \log_n(e) \ln \left(\frac{n^{-l_i}}{p_i}\right) \\
&= \log_n(e) \sum_{i=1}^{m} p_i \ln \left(\frac{n^{-l_i}}{p_i}\right) \leq \log_n(e) \sum_{i=1}^{m} p_i \left(\frac{n^{-l_i}}{p_i} - 1\right) \\
&= \log_n(e)(\sum_{i=1}^{m} n^{-l_i} - \sum_{i=1}^{m} p_i) = \log_n(e)(\sum_{i=1}^{m} n^{-l_i} - 1) \leq 0
\end{aligned}$$

und damit

$$\mathcal{L}(C, P) = \sum_{i=1}^{m} p_i l_i \geq \sum_{i=1}^{m} p_i \log_n \left(\frac{1}{p_i}\right).$$

Da diese Abschätzung für alle Codes C gilt, gilt sie auch für das Infimum der Kosten, womit

$$\mathcal{L}_X(P) \geq \sum_{i=1}^{m} p_i \log_n \left(\frac{1}{p_i}\right)$$

bewiesen ist.

Da $\ln(x) = x - 1$ nur für $x = 1$ gilt, folgt aus der vorstehenden Abschätzung sofort, dass die Gleichheit nur gilt, wenn $n^{-l_i} = p_i$ für $1 \leq i \leq m$ erfüllt ist.

Zum Beweis der oberen Abschätzung im Satz konstruieren wir zuerst wie folgt einen Code, wobei wir ohne Beschränkung der Allgemeinheit annehmen, dass die Wahrscheinlichkeiten in absteigender Folge geordnet sind, dass also

$$p_1 \geq p_2 \geq \ldots \geq p_m$$

gilt. Wir setzen

- $l_i = \lceil \log_n \left(\frac{1}{p_i}\right) \rceil$,
- $q_0 = 0$ und $q_i = \sum_{j=1}^{i} p_j$ für $1 \leq i \leq m-1$

und bestimmen die Codewörter $c_0, c_1, \ldots, c_{m-1}$ dann entsprechend der im Beweis von Satz 1.14 angegebenen Methode aus den n-ären Darstellungen der Zahlen $q_0, q_1, \ldots, q_{m-1}$. Wie im Beweis von Satz 1.14 kann gezeigt werden, dass die so konstruierten Wörter einen Präfixcode bilden.

Nach Konstruktion gilt für $C = \{c_0, c_1, \ldots, c_{m-1}\}$ die Abschätzung

$$\begin{aligned} \mathcal{L}(C,P) &= \sum_{i=1}^{m} p_i |c_{i-1}| = \sum_{i=1}^{m} p_i l_i \\ &\leq \sum_{i=1}^{m} p_i (\log_n \left(\frac{1}{p_i} + 1\right) = \sum_{i=1}^{m} p_i \log_n \left(\frac{1}{p_i}\right) + \sum_{i=1}^{m} p_i = \sum_{i=1}^{m} p_i \log_n \left(\frac{1}{p_i}\right) + 1. \end{aligned}$$

Da nach Definition für jeden Code über X auch $\mathcal{L}_X(P) \leq \mathcal{L}(C,P)$ gültig ist, folgt

$$\mathcal{L}_X(P) \leq 1 + \sum_{i=1}^{m} p_i \log_n \left(\frac{1}{p_i}\right) .$$

□

Beispiel 2.1. Wir betrachten die Verteilung P mit den Wahrscheinlichkeiten

$$p_1 = p_2 = 0.20, \quad p_3 = 0.19, \quad p_4 = 0.12, \quad p_5 = 0.11, \quad p_6 = p_7 = 0.09$$

und das Alphabet $X = \{0, 1\}$. Dann ergibt sich bei einer Rechnung mit fünf Stellen hinter dem Komma

$$\sum_{i=1}^{m} p_i \log \left(\frac{1}{p_i}\right) = 2.72666$$

und somit nach Satz 2.2

$$2.72666 \leq \mathcal{L}_X(P) \leq 3.72666 .$$

Wir konstruieren nun nach der im Beweis der oberen Abschätzung angegebenen Methode einen Code C_S, dessen Kosten höchstens 3.72666 sind. Die notwendigen Angaben können der folgenden Tabelle entnommen werden:

i	p_i	$\frac{1}{p_i}$	l_i	q_{i-1}	Dualdarst. q_{i-1}	c_{i-1}
1	0.20	5	3	0	0.0000000	000
2	0.20	5	3	0.20	0.0011001...	001
3	0.19	5.26...	3	0.40	0.0110011...	011
4	0.12	8.33...	4	0.59	0.1001011...	1001
5	0.11	9.09...	4	0.71	0.1011010...	1011
6	0.09	11.1...	4	0.82	0.1101001...	1101
7	0.09	11.1...	4	0.91	0.1110100...	1110

Die Kosten des so erhaltenen Präfixcodes

$$C_S = \{\ 000,\ 001,\ 011,\ 1001,\ 1011,\ 1101,\ 1110\ \}$$

betragen

$$\mathcal{L}(C_S, P) = \sum_{i=1}^{m} p_i l_i = (0.20 + 0.20 + 0.19) \cdot 3 + (0.12 + 0.11 + 0.09 + 0.09) \cdot 4 = 3.41\ .$$

Jedoch ist einfach zu sehen, dass sich die Codewörter aus C_S verkürzen lassen, ohne dass die Eigenschaft Präfixcode zu sein, verlorengeht. So sind die Anfänge 01, 100, 101, 110, 111 der letzten fünf Codewörter nicht Präfix der anderen Codewörter. Daher können wir den Code C_S zum Code

$$C'_S = \{\ 000,\ 001,\ 01,\ 100,\ 101,\ 110,\ 111\ \}$$

verkürzen. Für diesen Code ergeben sich die Kosten

$$\mathcal{L}(C'_S, P) = (0.20 + 0.20) \cdot 3 + 0.19 \cdot 2 + (0.12 + 0.11 + 0.09 + 0.09) \cdot 3 = 2.81\ ,$$

die deutlich unter denen des Codes C_S liegen. Es lässt sich noch eine weitere Verbesserung erreichen, wenn wir das kürzeste Codewort 01 der höchsten Wahrscheinlichkeit zuordnen. Durch diese Umordnung erhalten wir den Code

$$C''_S = \{\ 01,\ 000,\ 001,\ 100,\ 101,\ 110,\ 111\ \}$$

mit den Kosten

$$\mathcal{L}(C''_S, P) = 2.80\,.$$

Die Methode aus dem Beweis von Satz 2.2 zur Konstruktion eines kostengünstigen Codes geht auf C.E. Shannon zurück. Sie erfordert einigen Rechenaufwand. Ein erheblich einfacheres Verfahren wurde von R.M. Fano für Codes über $\{0, 1\}$ vorgeschlagen. Hierbei wird wie folgt vorgegangen:

Wir ordnen die Wahrscheinlichkeiten so, dass

$$p_1 \geq p_2 \geq \ldots \geq p_m$$

gilt. Wir beginnen mit

$$E_\lambda = \{p_1, p_2, \ldots, p_m\}$$

und konstruieren aus einer Menge

$$E_x = \{p_s, p_{s+1}, \ldots, p_t\}$$

mit $x \in \{0,1\}^*$ und $t - s \geq 1$ zwei Mengen E_{x0} und E_{x1} entsprechend der folgenden Vorschrift bis alle Mengen einelementig sind:

1. Für k, $s \leq k \leq t$ setzen wir

$$s_{k0} = \sum_{j=s}^{k} p_j \quad \text{und} \quad s_{k1} = \sum_{j=k+1}^{t} p_j$$

2. Wir bestimmen r so, dass

$$|s_{r1} - s_{r0}| = \min\{|s_{k1} - s_{k0}| \mid s \leq k \leq t\}$$

gilt. 3. Wir setzen

$$E_{x0} = \{p_s, p_{s+1}, \ldots, p_r\} \quad \text{und} \quad E_{x1} = \{p_{r+1}, p_{r+2}, \ldots, p_t\}.$$

Dadurch haben wir E_x in zwei Teile aufeinanderfolgender Wahrscheinlichkeiten geteilt, bei denen sich die Summen der Wahrscheinlichkeiten eines jeden Teiles möglichst wenig unterscheiden.

Gilt $E_x = \{p_j\}$, so nehmen wir x als j-tes Element des Codes, d.h. den Buchstaben, der mit der Wahrscheinlichkeit p_j auftritt, codieren wir durch x.

Wir haben zu zeigen, dass diese Konstruktion einen Code liefert. Dies folgt sofort aus der Tatsache, dass für zwei einelementige Mengen E_x und E_y weder x ein Präfix von y noch y ein Präfix von x ist, denn es gibt eine Menge E_z mit $x = z0z_1$ und $y = z1z_2$ oder $y = z0z_1$ und $x = z1z_2$. Daraus resultiert dann auch, dass die mittels der Methode von FANO gewonnenen Codes Präfixcodes sind.

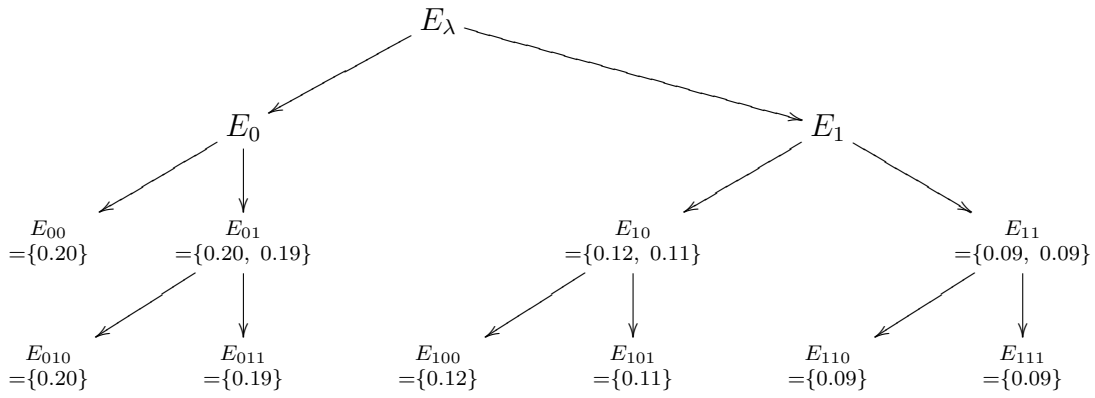
Beispiel 2.1. (Fortsetzung) Wir illustrieren die Methode von FANO anhand der Verteilung

$$P = \{0.20,\ 0.20,\ 0.19,\ 0.12,\ 0.11,\ 0.09,\ 0.09\}\ .$$

Es ergibt sich mit

$$E_\lambda = \{0.20,\ 0.20,\ 0.19,\ 0.12,\ 0.11,\ 0.09,\ 0.09\},$$
$$E_0 = \{\ 0.20,\ 0.20,\ 0.19\}, \quad E_1 = \{0.12,\ 0.11,\ 0.09,\ 0.09\},$$

der folgende Baum von Mengen.



und damit der Code

$$C_F = \{00,\ 010,\ 011,\ 100,\ 101,\ 110,\ 111\}$$

mit den Kosten

$$\mathcal{L}(C_F, P) = 0.20 \cdot 2 + (0.20 + 0.19 + 0.12 + 0.11 + 0.09 + 0.09) \cdot 3 = 2.80.$$

Daher ist der nach dem Verfahren von FANO konstruierte Code C_F in unserem Beispiel noch kostengünstiger als der mittels der Methode von SHANNON mit anschließender Kürzung gewonnene Code C'_S.

Die nach den Verfahren von SHANNON und FANO konstruierten Codes müssen nicht optimal sein (Beispiel 2.1 zeigt dies bereits für die Methode von SHANNON). Wir wollen nun eine Methode angeben, die auf D.A. HUFFMAN zurückgeht und mittels derer optimale Codes über $\{0,1\}$ erzeugt werden. Sie basiert auf dem folgenden Satz.

Satz 2.3 *Sei $C = \{c_1, c_2, \ldots, c_m\} \subseteq \{0,1\}^+$ ein optimaler Präfixcode für die Verteilung $P = \{p_1, p_2, \ldots, p_m\}$. Ferner gelte*

$$p_j = q_0 + q_1$$

und

$$p_1 \geq p_2 \geq \ldots \geq p_{j-1} \geq p_j \geq p_{j+1} \geq \ldots \geq p_m \geq q_0 \geq q_1.$$

Dann ist

$$C' = \{c_1, c_2, \ldots, c_{j-1}, c_{j+1}, \ldots, c_m, c_j 0, c_j 1\}$$

ein optimaler Präfixcode für die Verteilung

$$P' = \{p_1, p_2, \ldots, p_{j-1}, p_{j+1}, \ldots, p_m, q_0, q_1\}.$$

Beweis. Da C ein Präfixcode ist, bilden auch die Elemente von C' einen Präfixcode. Außerdem gilt

$$\begin{aligned}
\mathcal{L}(C', P') &= \sum_{i=1}^{j-1} p_i|c_i| + \sum_{i=j+1}^{m} p_i|c_i| + q_0|c_j 0| + q_1|c_j 1| \\
&= \sum_{i=1}^{j-1} p_i|c_i| + \sum_{i=j+1}^{m} p_i|c_i| + q_0(|c_j|+1) + q_1(|c_j|+1) \\
&= \sum_{i=1}^{j-1} p_i|c_i| + \sum_{i=j+1}^{m} p_i|c_i| + (q_0+q_1)(|c_j|+1) \\
&= \sum_{i=1}^{j-1} p_i|c_i| + \sum_{i=j+1}^{m} p_i|c_i| + p_j|c_j| + p_j \\
&= \sum_{i=1}^{m} p_i|c_i| + p_j \\
&= \mathcal{L}(C, P) + p_j.
\end{aligned}$$

Wir haben zu zeigen, dass C' optimal für P' ist.

Angenommen,

$$D' = \{d_1, d_2, \ldots, d_{j-1}, d_{j+1}, \ldots, d_{m-1}, d_m, d_{m+1}, d_{m+2}\}$$

ist optimal für die Verteilung P'. Ohne Beschränkung der Allgemeinheit können wir annehmen, dass D' ein Präfixcode ist. Sei l die maximale Länge eines Wortes aus D'. Wenn es genau ein Wort w der Länge l in D' gibt, so ist auch

$$D'' = (D \setminus \{w\}) \cup \{v\},$$

wobei v aus w durch Streichen des letzten Buchstaben entsteht, ein Präfixcode. Offenbar gilt $\mathcal{L}(D'', P') < \mathcal{L}(D', P)$, da die Wörter von D'' höchstens die gleiche Länge und in einem Fall eine kleinere Länge haben. Dies widerspricht aber der Annahme, dass D' optimal für P' ist.

Daher enthält D' mindestens zwei Wörter w_1 und w_2 der Länge l. Falls keine zwei Codewörter der Länge l einen gemeinsamen Präfix der Länge $l-1$ haben, so definieren wir v_1 und v_2 als die Präfixe der Länge $l-1$ von w_1 bzw. w_2. Dann ist $D''' = (D' \setminus \{w_1, w_2\}) \cup \{v_1, v_2\}$ erneut ein Präfixcode mit geringeren Kosten als D'.

Folglich gibt es ein Wort w so, dass $w0$ und $w1$ die Wörter der Länge l in D' sind. Wir können annehmen, dass $w0$ und $w1$ den minimalen Wahrscheinlichkeiten q_0 und q_1 zugeordnet sind, da sonst durch Umordnung der Codewörter ein Code erreicht werden kann, dessen Kosten nicht größer sind. Also gelten $d_{m+1} = w0$ und $d_{m+2} = w1$.

Wir betrachten nun den Code

$$D = \{d_1, d_2, \ldots, d_{j-1}, w, d_{j+1}, d_{j+2}, \ldots d_{m-1}, d_m\}.$$

Analog zur Rechnung zu Beginn des Beweises erhalten wir

$$\mathcal{L}(D', P') = \mathcal{L}(D, P) + p_j$$

und damit wegen der vorausgesetzten Optimalität von C für P

$$\mathcal{L}(C', P') = \mathcal{L}(C, P) + p_j \leq \mathcal{L}(D, P) + p_j = \mathcal{L}(D', P').$$

Da D' optimal für P' ist, muss daher auch C' optimal für P' sein. □

Aus Satz 2.3 folgt die folgende Methode zur Erzeugung eines optimalen Codes über $\{0, 1\}$ für die Verteilung

$$P = \{p_1, p_2, \ldots, p_m\}.$$

1. Wir setzen $P_1 = P$.

2. Für $2 \leq i \leq m-1$ konstruieren wir die Verteilung P_i wie folgt: Ist

   $$P_{i-1} = \{r_1, r_2, \ldots, r_{m-i}, r_{m-i+1}\}$$

   mit

   $$r_1 \geq r_2 \geq \ldots \geq r_{m-i} \geq r_{m-i+1},$$

   so setzen wir

   $$P_i = \{r_1, r_2, \ldots, r_{m-i-1}, r_{m-i} + r_{m-i+1}\}.$$

3. Für $P_{m-1} = \{t_1, t_2\}$, $t_1 \geq t_2$, setzen wir $C_{m-1} = \{0, 1\}$. (Dies ist der optimale Code für eine zweielementige Verteilung.)

4. Für $1 \leq j \leq m-2$ konstruieren wir den optimalen Code C_i für P_i aus dem optimalen Code C_{i+1} für P_{i+1} entsprechend dem Verfahren aus Satz 2.3.

5. Wir setzen $C = C_1$, welches der optimale Code für $P = P_1$ ist.

Beispiel 2.1. (Fortsetzung) Wir illustrieren die Methode anhand unseres Beispiels mit der Verteilung

$$P = \{0.20,\ 0.20,\ 0.19,\ 0.12,\ 0.11,\ 0.09,\ 0.09\}\ .$$

Die nachstehenden Tabellen geben die Konstruktion der Verteilungen P_i, $1 \leq i \leq m-1$, wobei wir stets die Wahrscheinlichkeiten von oben nach unten der Größe nach ordnen (da dies bei Satz 2.3 gefordert ist) und der Codes C_i, $1 \leq i \leq m-1$.

P_1	P_2	P_3	P_4	P_5	P_6
0.20	0.20	$\rightarrow$ 0.23	$\rightarrow$ 0.37	$\rightarrow$ 0.40	$\rightarrow$ 0.60
0.20	0.20	0.20	0.23	0.37 —	0.40
0.19	0.19	0.20	0.20 —	0.23 —	
0.12	$\rightarrow$ 0.18	0.19 —	0.20 —		
0.11	0.12 —	0.18 —			
0.09 —	0.11 —				
0.09 —					

C_6	C_5	C_4	C_3	C_2	C_1
0 —	1 —	00 —	01 —	10	10
1	$\rightarrow$ 00	01	10	11	11
	$\rightarrow$ 01	$\rightarrow$ 10	11	000	000
		$\rightarrow$ 11	$\rightarrow$ 000	001 —	010
			$\rightarrow$ 001	$\rightarrow$ 010	011
				$\rightarrow$ 011	$\rightarrow$ 0010
					$\rightarrow$ 0010

Für den so gewonnen Code

$$C_H = \{\ 10,\ 11,\ 000,\ 010,\ 011,\ 0010,\ 0011\ \}$$

ergeben sich die Kosten

$$\mathcal{L}(C_H, P) = (0.20 + 0.20) \cdot 2 + (0.19 + 0.12 + 0.11) \cdot 3 + (0.09 + 0.09) \cdot 4 = 2.78.$$

Da C_H optimal ist, gilt $\mathcal{L}_{\{0,1\}}(P) = 2.78$, womit auch gezeigt ist, dass die oben nach den Methoden von Shannon bzw. Fano gewonnenen Codes C'_S und C''_S bzw. C_F relativ kostengünstig für P sind.

Bei allen bisher betrachteten Konstruktionen von kostengünstigen Codes geht man davon aus, oft auftretenden Buchstaben relativ kurze Codewörter und seltener auftretenden Buchstaben längere Wörter zuzuordnen. Bei gewissen Aufgaben sind aber Blockcodes besonders günstig. Sind dabei k Symbole über $\{0, 1\}$ zu codieren, so wird mit Codewörter der Länge $\lceil log_2(k) \rceil$ eine kostenoptimale Variante erreicht. Für Texte der deutschen Sprache (ohne Berücksichtigung der Großschreibung) sind die 26 Buchstaben (ohne ä,ö,ü und ß), ein Leerzeichen _ (zur Trennung der Wörter) und die Satzzeichen Punkt, Ausrufezeichen, Fragezeichen, Semikolon, Doppelpunkt, Gedankenstrich, Apostroph und zwei Arten Anführungsstriche zu codieren. Mit $\mathcal{S}$ bezeichnen wir die Menge der Satzzeichen. Damit sind mehr als 32 und weniger als 64 Symbole zu codieren. Daher sind Codewörter der Länge 6 erforderlich.

Es ist aber offensichtlich, dass die Satzzeichen relativ selten im Vergleich zu den Buchstaben und dem Leerzeichen auftreten. In der deutschen Sprachen steht im Mittel nach 30 Buchstaben bzw. Leerzeichen ein Satzzeichen. Deshalb kann man mit dem folgenden Trick eine Codierung verwenden, bei der nur Codewörter der Länge 5 benutzt werden. Wir betrachten Codierungen

$$\varphi : \{a, b, c, \ldots, x, y, z, \_\} \to \{0,1\}^5 \text{ und } \psi : \mathcal{S} \to \{0,1\}^5.$$

Da sowohl das eigentliche Alphabet als auch die Menge der Satzzeichen keine 32 Symbole enthalten, nutzen wir bei den Codierungen φ und ψ nicht das Wort 11111 und verwenden dieses als Markierung, die den Wechsel von φ zu ψ und umgekehrt bezeichnet. Mit

$$\varphi(a) = 00001, \varphi(i) = 01001, \varphi(m) = 01101, \psi(.) = 01001$$

wird durch

01101 00001 01101 00001 01101 01001 00001 11111 01001 11111 01001 00001

die Folge

$$m\ a\ m\ a\ m\ i\ a\ .\ i\ a$$

codiert. Geht man davon aus dass durchschnittlich auf 30 Buchstaben und Leerzeichen ein Satzzeichen folgt, so benötigt man zur Codierung von diesen 30 Buchstaben und dem Satzzeichen 165 Elemente aus $\{0,1\}$, da zusätzlich noch zweimal die Markierung 11111 auftritt. Dies entspricht nur durchschnittlich 5,32 Ziffern 0 bzw. 1 je codiertem Symbol. Daher ist diese Art der Codierung (mit Wechsel der Codes) kostengünstiger als die mit einer Folge aus 6 Ziffern bei Verwendung einer direkten Codierung der Gesamtmenge $\{a, b, \ldots, y, z\} \cup \mathcal{S}$.

Literaturverzeichnis

[1] J. Berstel / D. Perrin, *Theorie of Codes*. Academic Press, 1985.

[2] A. Beutelspacher, Kryptologie. Vieweg, 1991.

[3] J. Dassow, A note on DT0L Systems. *Bull. EATCS* **22** (1984) 11–14.

[4] J. Duske / H. Jürgensen, *Kodierungstheorie*. BI-Taschenbuch 25, Mannheim, 1977.

[5] M.R. Garey / D.S. Johnson, *Computers and Intractability / A Guide to NP-Completeness.* Freeman & Company, 1978.

[6] T. Grams, *Codierungsverfahren*. BI-Hochschultaschenbuch 625, Mannheim, 1986.

[7] J.E. Hopcroft / J.D. Ullman, Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie. Addison-Wesley, 1990.

[8] J. Kari, Observations concerning a public-key cryptosystem based on iterated morphisms. *Theor. Comp. Sci.* **66**(1989) 45–53.

[9] W.I. Löwenstein, Kodierungstheorie. In: *Diskrete Mathematik und mathematische Fragen der Kybernetik*, Herausg.: S.W.Jablonski / O.B.Lupanov, Akademie-Verlag, 1980.

[10] B. Martin, *Codage, cryptologie et applications*. Presses Polytechniques et Universitaires Romandes, 2004.

[11] R. Merkle / M. Hellman, Hiding informations and signatures in trapdoor knapsacks. *IEEE Transactions on Information Theory* **IT-24** (1978) 525–530.

[12] W.W. Peterson / E.J. Weldon, *Error-Correcting Codes.* MIT Press, Cambridge, 1972.

[13] R.L. Rivest / A. Shamir / L. Adleman, A method for obtaining digital signatures and public-key cryptosystems. *Comm. ACM* **21** (1978) 120–126.

[14] G. Rozenberg / A. Salomaa, *Mathematical Theory of L Systems*. Academic Press, 1980.

[15] A. Salomaa, *Jewels of Formal Language Theory*. Computer Science Press, 1981.

[16] A. Salomaa, *Public-Key Cryptography*. Springer-Verlag, 1996.

[17] A. Salomaa / E. Welzl, On a public-key cryptosystems based on iterated morphisms and substitutions. Manuskript, 1983.

[18] H.J. Shyr, *Free Monoids and Languages*. Hon Min Book Co., Taichung, Taiwan, 1991.

[19] P. Sweeney, *Codierung zur Fehlererkennung und Fehlerkorrektur*. Hanser-Verlag, 1992.

[20] D. Wätjen, Kryptographie. Grundlagen, Algorithmen, Protokolle. Spektrum-Verlag, 2003.

[21] W. Willems, *Codierungstheorie*. Walter de Gruyter, Berlin, 1999.