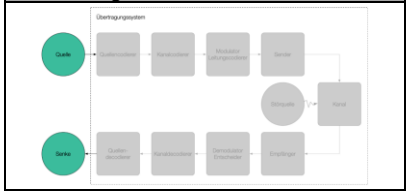
Informationstheorie

Einordnung

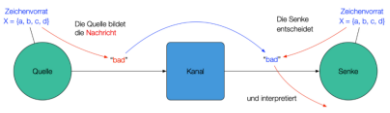


Was ist «Information»?

Eine Nachricht wird als «Information» bezeichnet, wenn sie «relevant» und «nicht-redundant» ist.

- Relevant: Der Empfänger kann die Nachricht verstehen (die «Sprache» stimmt).
- Nicht-Redundant: Der Empfänger kann die Nachricht nicht voraussagen.

⇨ Eine Nachricht wie z.B. «111111 ...» wäre redundant.



Nachricht (Darstellung & Bedeutung)	redundant	nicht-redundant
irrelevant	Zeichenvorrat bei Quelle und Senke verschieden	
relevant	vorhersagbar	Information

⇨ Ein Kanal kann z.B. die Sprache oder ein Netzwerk sein.

Grundbegriffe & Formeln

Entscheidungsgehalt H_0

Die Grösse des «Aufwands», also die Anzahl von Bits, welche für die Bildung eines Zeichens benötigt werden.

$$\boldsymbol{H_0 = \log_2(N)} \; [Bit]$$

⇨ N = Anzahl der Zeichen im «Zeichenvorrat»
⇨ Hinweis: Bei diesen Formeln haben wir «gebrochene» Bits.

Entscheidungsfluss H_0^*

Die Übertragungsrate eines Zeichens über einen Kanal in Bits pro Sekunde.

$$\boldsymbol{H_0^* = \frac{\log_2(N)}{\tau}} \; \left[\frac{Bit}{s}\right]$$

⇨ N = Anzahl der Zeichen, τ = Zeit für eine Übertragung

Informationsgehalt $I(x_k)$

Die Grösse der «Information», also die Anzahl von Bits eines Zeichens in Anbetracht von dessen Häufigkeit.

$$\boldsymbol{I(x_k) = \log_2\left(\frac{1}{p(x_k)}\right)} \; [Bit]$$

⇨ $p(x_k)$ = Wahrscheinlichkeit für das Auftreten des Zeichens x_k
⇨ Seltene Zeichen sind «Informationsreicher» als häufige.
⇨ Resultat: Häufige Zeichen brauchen weniger Bits als seltene.

Entropie $H(X)$

Bezeichnet den mittleren Informationsgehalt von allen Zeichen in der Quelle.

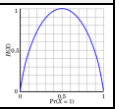
$$\boldsymbol{H(X) = \sum_{k=1}^{N} p(x_k) * \log_2\left(\frac{1}{p(x_k)}\right)} \; \left[\frac{Bit}{Zeichen}\right]$$

⇨ Wird maximal, wenn alle Zeichen gleichwahrscheinlich sind.
⇨ Wobei bei dieser Formel gilt: $I(x_k) = \log_2(1/p(x_k))$

Beweis für die max. Entropie

$X = \{x_1, x_2\}, p(x_1) = p, p(x_2) = 1 - p$

$\Rightarrow H(X) = p * \log_2\left(\frac{1}{p}\right) + (1-p) * \log_2\left(\frac{1}{1-p}\right)$



Nun können wir das Diagramm rechts aufzeichnen oder die Formel ableiten, gleich 0 setzten und das globale Max. bestimmen.

Codewortlänge $L(x_k)$ (vgl. Informationsgehalt)

Die tatsächliche Bitgrösse der obigen, oftmals gebrochenen Bits bezeichnet man als «Codewortlänge».

$$\boldsymbol{L(x_k) = \text{Aufgerundet}(I(x_k))} \; [Bit]$$

⇨ Wobei wir «mathematisch» auch $\lceil I(x_k) \rceil$ schreiben können.

Mittlere Codewortlänge L (vgl. Entropie)

Wir können nun auch die mittlere Codewortlänge von allen Zeichen berechnen.

$$\boldsymbol{L = \sum_{k=1}^{N} p(x_k) * \left\lceil \log_2\left(\frac{1}{p(x_k)}\right) \right\rceil} \; \left[\frac{Bit}{Zeichen}\right]$$

⇨ Günstig ist, wenn der Wert L möglichst klein ist.
⇨ Wobei bei dieser Formel gilt: $L(x_k) = \lceil \log_2(1/p(x_k)) \rceil$

Redundanz R

Bezeichnet die Anzahl «unnötigen» Bits, welche bei einer Codierung ohne Häufigkeit / mit ganzen Bits benötigt werden.

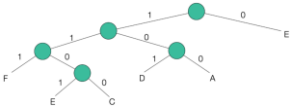
$$\boldsymbol{R_Q = H_0 - H(X)}$$
$$\boldsymbol{R_C = L - H(X)}$$

⇨ Wobei gilt Q = Quelle und C = Code

Präfixeigenschaft

Eine Gruppe von Codeworten besitzt die Präfixeigenschaft, wenn alle Codes ohne Trennzeichen identifizierbar sind.

⇨ Die Zeichen befinden sich in den «Blättern» des Baumes.
⇨ ASCII hat die Präfixeigenschaft, der Morsecode nicht.



⇨ Beispiel eines Codewortbaums mit Präfixeigenschaft

Shannon'sches Codierungstheorem

Das Codierungstheorem besagt, dass:

1. Für jede beliebige Binärcodierung mit Präfixeigenschaft ist die mittlere Codewortlänge nicht kleiner als die Entropie: $H(X) \leq L$
2. Für jede beliebige Quelle kann eine Binärcodierung gefunden werden, so dass gilt: $H(X) \leq L \leq H(X) + 1$

Quelle mit / ohne Gedächtnis

Quelle **ohne** Gedächtnis (QoG)

Bisher sind wir immer von einer QoG ausgegangen. Das Bedeutet, dass ein Zeichen «nicht» von dem zuvor gesendeten Zeichen abhängig ist.

⇨ Formell gilt daher, dass $p(x_i, y_k) = p(x_i) * p(y_k)$

Quelle **mit** Gedächtnis (QmG)

Allgemein können wir «nicht» von einer QoG ausgehen. Oft sind die gesendeten Zeichen voneinander abhängig.

⇨ Formell gilt daher, dass $p(x_i, y_k) = p(x_i) * p(y_k|x_i)$
⇨ Beispiel: Die deutsche Sprache («u» folgt auf «Q», etc.)

Berechnung

Die Formeln für den Informationsgehalt und die Entropie ändern sich wiefolgt:

$$\boldsymbol{H(X,Y) = \sum_{k=1}^{N}\sum_{i=1}^{N} p(x_k, y_i) * \log_2\left(\frac{1}{p(x_k, y_i)}\right)}$$

Wobei ausserdem gilt:

$$\boldsymbol{H(X,Y) = H(X) + H(Y|X)}$$
$$\boldsymbol{H(Y|X) = H(X,Y) - H(X)}$$

⇨ Erinnerung: $p(x_i, y_k) = p(x_i) * p(y_k|x_i)$

Markov-Diagramm

Wir können die Entropie auch über das «Markov-Diagramm» berechnen.

⇨ Das Diagramm mit Berechnung befindet sich im Anhang.

Interpretation

Wir können aufzeigen, dass:

$$H_0 \geq H(X) \geq H(Y|X)$$

Daraus folgt, dass die Entropie einer QoG stets grösser oder gleich der Entropie einer QmG ist:

$$H_{\text{oG}}(X) \geq H_{\text{mG}}(X)$$

⇨ Daraus gilt nun auch: $R_Q = H(X) - H_{\text{oG}}(X) \leq H(X) - H_{\text{mG}}(X)$
⇨ D.h. man soll stets Zeichenketten statt Zeichen codieren.

Datenkomprimierung

Einordnung

⇨ Quellencodierung beinhaltet Kompression & Verschlüsselung

Was bedeutet Datenkomprimierung?

Die Kompression hat das Ziel, den Aufwand der Datenspeicherung und Datenübertragung zu reduzieren.

⇨ D.h. Sie soll Redundanz und Irrelevanz entfernen.

Arten von Datenkomprimierung

Wir unterscheiden zwischen:

- Verlustfrei: Die Ausgangsdaten können im Nachhinein wieder rekonstruiert werden.
- Verlustbehaftet: Die Ausgangsdaten lassen sich im Nachhinein nicht rekonstruieren.

⇨ Im Modul wurden nur «verlustfreie» Methoden angeschaut.

Anforderungen

Eine optimale Kompression hat:

- Eine hohe Komprimierungsrate
- Eine hohe En- und Decode-Geschwindigkeit
- Geringe Ansprüche an die Hardware

Arten der Verfahren

Huffman-Codierung

Huffman ist ein rekursives Verfahren für die Bildung eines kommafreien Codes mit minimaler mittleren Codewortlänge.

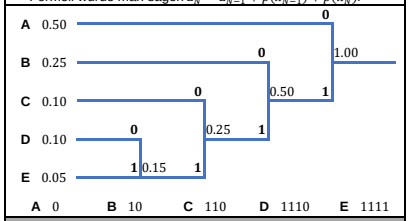
⇨ «kommafrei» heisst, dass der Code die Präfixeigenschaft hat.
⇨ «rekursiv» heisst, dass wir in den Blättern beginnen.

Funktionsweise

Huffman funktioniert wiefolgt:

1. Initialisierung: Ordne die Zeichen gemäss ihren Auftrittswahrscheinlichkeiten.
2. Rekursion: Fasse in jedem Schritt die zwei Zeichen mit den kleinsten Wahrscheinlichkeiten zusammen und bilde daraus einen Binärbaum mit 1 und 0.
3. Abschluss: Schreibe nun «von links nach rechts» die Codewörter im Binärbaum auf.

⇨ Formell würde man sagen $L_N = L_{N-1} + p(x_{N-1}) + p(x_N)$.



Lauflängenkomprimierung

Diese Kompressionsmethode versucht, aufeinanderfolgende Zeichen zu erkennen und zu verkürzen.

⇨ Auch «Run Length Encoding» oder «Run Length Coding»
⇨ Wird bei vielen Bildformaten genutzt (z.B. BPM, TIFF, etc.)

Funktionsweise Zeichen

Bei der Komprimierung von Textzeichen wird jede Zeichensequenz (> 1) durch das Zeichen und dessen Anzahl ersetzt.

- Quelltext: w = Agggbbehfffgggg
- Codiert: w_c = A3g2beh3f4g
- Zeichenanzahl: $|w| = 15$, $|w_c| = 11$

Kompression: $\frac{|w_c|}{|w|} = \frac{11}{15} = 0.73 = 73\%$

Funktionsweise Bitfolgen

Bei der Komprimierung von Bitfolgen gibt es nur zwei Arten von Sequenzen, nämlich $N * 1$ oder $N * 0$.

⇨ Auf eine 1-Sequenz folgt immer eine 0-Seq. und umgekehrt.
⇨ Die Ausnahme ist das Ende der Nachricht (EOF)

Bei Bitfolgen müssen sich Sender und Empfänger darauf einigen, ob mit 1 oder 0 begonnen wird. Danach wird nur noch die Länge der Sequenzen codiert.

- Quelltext: $w = 1111111000001000000011111$
- Definition: Start mit einer «1»
- Sequenzen: $S = 7\ 5\ 1\ 6\ 5$
- Codegrösse: $\max(S) = 7 = 111_b \Rightarrow 3$ Bits
- Codiert: w_c = 111 101 001 110 101
- Zeichenanzahl: $|w| = 24$, $|w_c| = 15$

Kompression: $\frac{|w_c|}{|w|} = \frac{15}{24} = 0.62 = 62\%$

⇨ Die Definition kann z.B. Konvention oder ein Startbit sein.

Lempel-Ziv

Lempel-Ziv ist ein tabellengesteuertes Verfahren, bei welchem wiederkehrende Muster / Phrasen in einem Text erkennt, in einer Tabelle gespeichert und für die Codierung wiederverwendet werden.

⇨ Voraussetzung: Text muss Regelmässigkeiten beinhalten.
⇨ Die Tabelle wird zur Laufzeit erfasst und codiert.
⇨ Problem: Effiziente Umsetzung des «Phrasenspeichers»?

Funktionsweise Normal

Lempel-Ziv funktioniert wiefolgt:

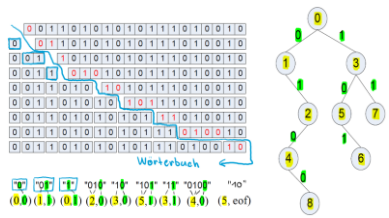
1. Suche im «Search-Buffer» (SB) die längste Zeichenfolge, die mit den nächsten n Zeichen übereinstimmt.
2. Kodiere den Eintrag in der Form (D, G, N):
   - D = Distanz zur Zeichenfolge im SB
   - G = Grösse der Zeichenfolge ($= n$)
   - N = Nächstes Zeichen (Position $n + 1$)
3. Schiebe nun die $n + 1$ Zeichen in den SB.
4. Wiederhole, bis alle Zeichen codiert sind.

SB	Look-ahead buffer	coding
	sir_sid_eastman	(0,0,"s")
s	ir_sid_eastman	(0,0,"i")
si	r_sid_eastman	(0,0,"r")
sir	_sid_eastman	(0,0,"_")
4321← sir_	sid_eastman	(4,2,"d")
4321← sir_sid	_eastman	(4,1, "e)

Funktionsweise Bitfolgen

Der binäre LZ funktioniert wiefolgt:

1. Beginne mit einem Binärbaum, der einen Root-Knoten mit dem Index 0 besitzt.
2. Suche im Binärbaum die längste Zeichenfolge (bzw. «Knotenfolge»), die mit den nächsten n Zeichen übereinstimmt.
3. Kodiere den Eintrag in der Form (I, N):
   - I = Index vom aktuellen Knoten im Baum
   - N = Nächstes Zeichen (Position $n + 1$)
4. Erstelle beim aktuellen Knoten ein neuen Kindknoten mit dem Index $I_{max} + 1$ und dem Zeichen an der Position $n + 1$ als Kantenbeschriftung.
5. Verschiebe das «Fenster» hinter die $n + 1$ Zeichen auf das nächste Zeichen.
6. Wiederhole, bis alle Zeichen codiert sind.



Funktionsweise Lempel-Ziv-Welch

LZ-Welch wird für die Komprimierung von Zahlenfolgen (Ziffern 0-9) verwendet. Das Verfahren funktioniert wiefolgt:

1. Beginne mit einem «Wörterbuch», welches die Ziffern 0-9 (Index I = Ziffer) beinhaltet.
2. Suche im «Wörterbuch» die längste Zeichenfolge, die mit den nächsten n Zeichen übereinstimmt.
3. Speichere den Index dieses Eintrags.
4. Bilde einen neuen Eintrag im «Wörterbuch» mit dem Index $I_{max}+1$ bestehend aus den n aktuellen und dem $n+1$-tem Zeichen.
5. Verschiebe das «Fenster» hinter die n Zeichen auf das $n+1$-te Zeichen.
6. Wiederhole, bis alle Zeichen codiert sind.

Index	Eintrag
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9

Buffer	Erkannte Zeichenfolge (Index)	Neuer Eintrag
123123123123	1 (1)	-> 10: 12
123123123123	2 (2)	-> 11: 23
123123123123	3 (3)	-> 12: 31
123**12**3123123	12 (10)	-> 13: 123
12312**312**3123	31 (12)	-> 14: 312
123123**123**123	23 (11)	-> 15: 231
123123123**123**	123 (13)	

- Quelltext: $w = 123123123123$
- Codiert: $w_c = 1\ 2\ 3\ 10\ 12\ 11\ 13$
- Codegrösse: $\max(w_c) = 13 = 1101_b \Rightarrow 4$ Bit
- Zeichenanzahl: $|w| = 12$, $|w_c| = 7$

Kompression: $\frac{4*|w_c|}{4*|w|} = \frac{28}{48} = 0.58 = 58\%$

⇨ Hinweis: $4 * |w|$ da immer gilt: $\max(w) = 9 = 1001_b \Rightarrow 4$ Bits.

Kombinationen

Wir können bei Bedarf die Komprimierungsverfahren auch kombinieren.

⇨ z.B. Die Daten mit LZ und dann das Wörterbuch mit Huffman.

Verschlüsselung

Was bedeutet Verschlüsselung?

Die Verschlüsselung hat das Ziel, eine Nachricht so zu verändern, dass ungewollte Personen sie nicht lesen können.

⇨ Kryptographie: Krypto (verborgen, geheim), Grafie (Schrift)

Arten von Verschlüsselung

Wir unterscheiden zwischen:

- Symmetrisch: Das Ver- und Endschlüsseln verwenden den gleichen Schlüssel.
- Asymmetrisch: Das Ver- und Endschlüsseln verwenden verschiedene Schlüssel.

Symmetrische Verfahren

Bei symmetrischen Verfahren erstellen wir genau einen Schlüssel für das Ver- und Endschlüsseln der Daten. Es gilt:

- Wollen 100 Personen paarweise geheime Botschaften austauschen, so braucht es für jedes Paar einen eigenen Schlüssel.
- Die Anzahl der erforderlichen Schlüssel können wir wiefolgt berechnen:

$$N_{\text{Schlüssel}} = \binom{100}{2} = \frac{100*99}{2} = 4950$$

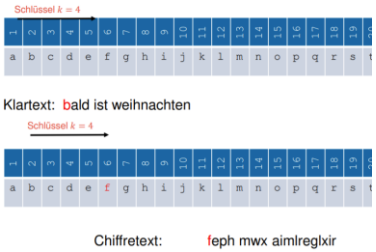
⇨ Siehe «Kombinationen ohne Wiederholung» (ExEv)

- Die Anzahl der Schlüssel, die eine Person somit abspeichern muss, lautet dann:

$$N_{\text{Personen}} - 1 = 100 - 1 = 99$$

Caesar Chiffre Substitutionsverfahren

Bei diesem Verfahren wird ein Text einfach um k Zeichen im Alphabet verschoben. Der Schlüssel ist somit k.



Nachteile

Das Verfahren hat einige Nachteile:

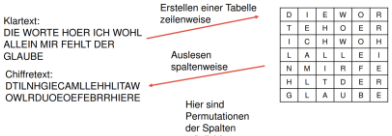
- Die statistischen Eigenschaften des Textes (z.B. häufige Buchstaben) bleiben erhalten.
- Die Anzahl der möglichen Schlüssel ist enorm klein (= Grösse vom Alphabet).

⇨ Daraus folgt: Kennen wir die Sprache und ist die Probe gross genug, so können wir den Schlüssel schnell ermitteln.

Transpositionsverfahren

Bei diesem Verfahren werden die Zeichenfolgen des Klartextes nach bestimmten Regeln «verwürfelt».

⇨ D.h. es findet keine Ersetzung (Substitution) der Zeichen statt.



⇨ Eine weitere Variante ist das «Vigenère-Chiffre».

DES: Data Encrcryption Standard

Dieses Verfahren wurde 1977 als offizieller Standard der US-Regierung bestätigt und wurde seither oft eingesetzt.

⇨ Das Verfahren wird heute aufgrund der kleinen Schlüssellänge (56 Bits) für viele Anwendung als unsicher betrachtet.
⇨ Auch wurde DES wegen der Beteiligung der NSA oft kritisiert.

Asymmetrische Verfahren

Bei asymmetrischen Verfahren wird ein Schlüssel für das Verschlüsseln und einer für das Entschlüsseln erstellt.

- Public Key: Schlüssel fürs Verschlüsseln
- Private Key: Schlüssel fürs Entschlüsseln

Daraus folgt nun:

- Bei 100 Personen braucht es 100 Schlüssel.
- Wenn alle Personen miteinander kommunizieren wollen, so muss jede Person 101 Schlüssel abspeichern:

$$99_{\text{Fremde}} + 2_{\text{Eigene}} = 100_{\text{Public}} + 1_{\text{Private}} = 101$$

⇨ Bei einer «öffentlichen Schlüsselbank» nur ein Private Key.

RSA-Verschlüsselung

RSA ist die bekannteste Methode für die asymmetrische Verschlüsselung. Für ihre Anwendung werden einige mathematische Konzepte benötigt.

1. Inverse Zahl

Sind a und b zwei teilerfremde Zahlen, dann existiert eine Zahl a^{-1}, für die gilt:

$$a * a^{-1} \bmod b = 1$$

Das a^{-1} nennen wird das Inverse von a.

⇨ Erinnerung: Teilerfremd bedeutet, dass $\text{ggT}(a,b) = 1$

2. Eulerfunktion

Die Eulerfunktion φ gibt die Anzahl zu einer Zahl n teilerfremden Zahlen an.

⇨ Also die Zahlen $x < n$, für die gilt $\text{ggT}(n,x) = 1$.

Berechnung

Wenn n eine Primzahl ist, gilt:

- Standardregel: $\varphi(n) = n - 1$
- Potenzregel: $\varphi(n^k) = n^{k-1} * (n-1)$
- Produkteregel: $\varphi(n_a * n_b) = \varphi(n_a) * \varphi(n_b)$

Wenn n keine Primzahl ist, gilt:

- Primfaktorzerlegung: $\varphi(n) = \varphi(n_0 * \ldots * n_k)$

⇨ Doppelte Primzahlen immer in Potenzschreibweise bringen.
⇨ z.B.: $\varphi(12) = \varphi(2*2*3) = \varphi(2^2*3) = \varphi(2^2) * \varphi(3) = 4$

Satz von Euler

Wenn a und b zwei teilerfremde Zahlen sind, so gilt ausserdem:

$$a^{\varphi(b)} \bmod b = 1$$

⇨ Diese Aussage gilt, solange $a < b$ ist.

3. Euklidischer Algorithmus

Berechnung GGT

Der EEA ist ein Verfahren zur Bestimmung des grössten gemeinsamen Teilers zweier Zahlen a und b.

1. Beginne mit der Formel «$a = q * b + r$», wobei für die Werte gilt: «$a/b = q$ Rest r»
2. Berechne nun schrittweise diese Formel, wobei bei jedem Schritt für «$a \to b$» und für «$b \to r$» eingesetzt wird.
3. Wiederhole, bis «$r = 0$» ist.

1. Berechnung | **2. Umformung (Inverse)**

$48 = 9*5+3$ ⇒ $3 = 48 - 9*5$
$5 = 1*3+2$ ⇒ $2 = 5 - 1*3$
$3 = 1*2+1$ ⇒ $1 = 3 - 1*2$
$2 = 2*1+0$

⇨ In diesem Fall gilt also: $\text{ggT}(48,5) = 1$

Berechnung Inverse

Sind a und b teilerfremd, so kann über den EEA auch das Inverse b^{-1} bestimmt werden, so dass gilt: $b * b^{-1} \bmod a = 1$.

1. Starte mit der letzten Formel aus «2.»
2. Ersetze nun das r aus dem aktuellen Schritt mit der Formel aus dem vorherigen Schritt.
3. Nun werden nur die Klammerterme ausmultipliziert, die Faktoren bleiben stehen.
4. Wiederhole für alle Schritte.

3. Inverse

$$\begin{aligned} 1 &= 3 - 1*2 \\ &= 1*3 - 1*(5 - 1*3) \\ &= 1*3 - 1*5 + 1*3 = 2*3 - 1*5 \\ 1 &= 2*3 - 1*5 \\ &= 2*(48 - 9*5) - 1*5 \\ &= 2*48 - 18*5 - 1*5 = 2*48 - 19*5 \\ 1 &= 2*48 - 19*5 \end{aligned}$$

Anschliessend gilt:

- Das Inverse von $b = 5$ ist $b^{-1} = -19$
- Das positive Inverse ist $b^{-1} = -19 + 48 = 29$
- Es gilt: $5 * -19 \bmod 48 = 5 * 29 \bmod 48 = 1$

⇨ Für das positive Inverse gilt also: $b^{-1\prime} = b^{-1} + a$

Funktionsweise RSA

Wir können nun die RSA-Verschlüsselung anwenden. Dazu definieren wir:

- Privater Schlüssel: d
- Öffentlicher Schlüssel: n und e

1. Bestimme n

Wähle zwei Primzahlen p, q und berechne deren Produkt:

$$p = 11, q = 19 \to n = p * q = 209$$

2. Bestimme $\varphi(n)$

Berechne $\varphi(n)$ mit der Eulerfunktion:

$$\varphi(209) = \varphi(11) * \varphi(19) = 180$$

3. Bestimme d oder e

Berechne basierend auf einem gegebenen Schlüssel d oder e das positive Inverse über den EEA, wobei gilt:

$$a_{EEA} = \varphi(n) \qquad b_{EEA} = d \;/\; e$$

4. Ver- und Endschlüsseln

Verschlüssle einen Wert m mit:

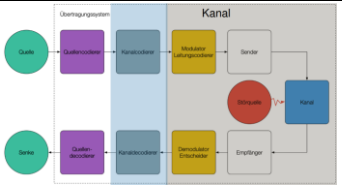
$$c = m^e \bmod n$$

Entschlüssle einen Wert c mit:

$$m = c^d \bmod n$$

Kanalmodell

Einordnung

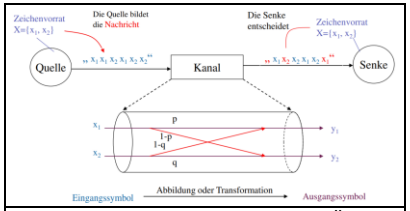


⇨ Das Kanalmodell wird für die Kanalkodierung benötigt.

Was ist ein Kanalmodell?

Das Kanalmodell ist eine abstrakte Abbildung eines Kanals. Es beschreibt u.a. die Schwierigkeiten bei der Datenübertragung in Bezug auf den Kanal.

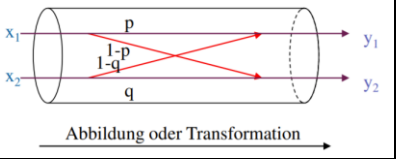
⇨ z.B. die Fehlerwahrscheinlichkeit bei der Datenübertragung.



Die Abbildung zeigt, dass bei der Übertragung von Daten aufgrund von «Rauschen» Fehler auftreten können.

⇨ Das «Rauschen» kann z.B. eine schlechte Verbindung sein.
⇨ Wir können dieses Phänomen in einer Kanalmatrix abbilden.

Kanalmatrix



Die Kanalmatrix beschreibt die Wahrscheinlichkeiten, dass ein Zeichen x_i auf ein korrektes oder inkorrektes Zeichen y_i abgebildet wird.

$$P(Y|X) = \begin{array}{c} \\ x_1 \\ \vdots \\ x_m \end{array} \begin{array}{c} \begin{array}{ccc} y_1 & \cdots & y_n \end{array} \\ \begin{bmatrix} p(y_1|x_1) & \cdots & p(y_n|x_1) \\ \vdots & \ddots & \vdots \\ p(y_1|x_m) & \cdots & p(y_n|x_m) \end{bmatrix} \end{array}$$

⇨ Lese z.B. $p(y_1|x_2)$ als: Die Wahrscheinlichkeit, dass ein y_1 ankommt, unter der Voraussetzung das ein x_2 gesendet wurde.

Eigenschaften

Bei einer Kanalmatrix gilt:

- Ist die Wahrscheinlichkeit für eine inkorrekte Zuweisung 0, so ist der Kanal «nicht» gestört.

$$P(Y|X) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

- Sind alle Zuweisungen gleichwahrscheinlich, so ist der Kanal «vollständig» gestört.

$$P(Y|X) = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$$

- Ist $n = m$, so ist der Kanal symmetrisch.

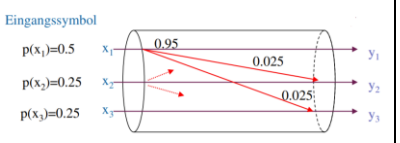
Ausgangswahrscheinlichkeit

Wir können nun die Wahrscheinlichkeit für das Auftreten eines Zeichens y_i anhand der Kanalmatrix berechnen.

$$p(y_i) = \sum_{k=1}^{m} p(x_k) * p(y_i|x_k)$$

⇨ Die Summe aus den inkorrekten und korrekten Zuweisungen.

Berechnungsbeispiel



Kanalmatrix:

$$p(Y|X) = \begin{bmatrix} 0.95 & 0.025 & 0.025 \\ 0.025 & 0.95 & 0.025 \\ 0.025 & 0.025 & 0.95 \end{bmatrix}$$

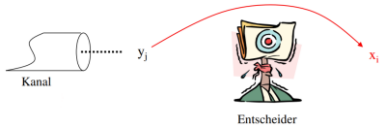
Ausgangswahrscheinlichkeiten:

$$\begin{pmatrix} p(y_1) \\ p(y_2) \\ p(y_3) \end{pmatrix} = \begin{bmatrix} p(x_1)\cdot p(y_1|x_1) + p(x_2)\cdot p(y_1|x_2) + p(x_3)\cdot p(y_1|x_3) \\ p(x_1)\cdot p(y_2|x_1) + p(x_2)\cdot p(y_2|x_2) + p(x_3)\cdot p(y_2|x_3) \\ p(x_1)\cdot p(y_3|x_1) + p(x_2)\cdot p(y_3|x_2) + p(x_3)\cdot p(y_3|x_3) \end{bmatrix}$$

$$\begin{bmatrix} 0.4875 \\ 0.25625 \\ 0.25625 \end{bmatrix} = \begin{bmatrix} 0.5\cdot 0.95 + 0.25\cdot 0.025 + 0.25\cdot 0.025 \\ 0.5\cdot 0.025 + 0.25\cdot 0.95 + 0.25\cdot 0.025 \\ 0.5\cdot 0.025 + 0.25\cdot 0.025 + 0.25\cdot 0.95 \end{bmatrix}$$

Maximum-Likelihood-Verfahren

Ist ein Kanal gestört, so müssen wir anhand des erhaltenden Zeichens y_i entscheiden, welches Zeichen x_i tatsächlich gesendet wurde.



Funktionsweise

Beim «Maximum-Likelihood» nehmen wir dabei einfach den Wert x_i, welcher in der Kanalmatrix für ein gegebenes y_i am wahrscheinlichsten ist.

⇨ Bestimme also in jeder «Spalte» den grössten Wert.

$$p(Y|X) = \begin{bmatrix} 0.6 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.6 \\ 0.3 & 0.2 & 0.5 \end{bmatrix} \quad \begin{matrix} y_1 \to x_1 \\ y_2 \to x_3 \\ y_3 \to x_2 \end{matrix}$$

y_1 y_2 y_3

⇨ Lese: Wenn ich ein y_3 erhalte, interpretiere ich es als x_2.

Transinformation

Wir können feststellen, dass bei der Datenübertragung über einen gestörten Kanal «Informationen» verloren gehen. Das bedeutet, der mittlere Informationsgehalt (die Entropie) nimmt ab.

$$H(X) \neq H(Y)$$

⇨ $H(X)$: Eingangsentropie, $H(Y)$: Ausgangsentropie

Verbundentropie

Beschreibt die Kombination der Entropien am Ein- und Ausgang des Kanals.

$$\boldsymbol{H(X,Y)} = -\sum_{k=1}^{N}\sum_{i=1}^{N} p(x_k, y_i) * \log_2\big(p(x_k, y_i)\big)$$

Äquivokation Verlust

Beschreibt die Ungewissheit über ein «gesendetes» Zeichen bei bekannten Empfangszeichen.

$$\boldsymbol{H(X|Y)} = -\sum_{k=1}^{N}\sum_{i=1}^{N} p(y_i) * p(x_k|y_i) * \log_2\big(p(x_k|y_i)\big)$$

⇨ Ist der Kanal fehlerfrei, so ist die Äquivokation gleich 0.
⇨ Wird auch «Rückschlussentropie» genannt.

Irrelevanz Rauschen

Beschreibt die Ungewissheit über ein «empfangenes» Zeichen bei bekannten Sendezeichen.

$$\boldsymbol{H(Y|X)} = -\sum_{k=1}^{N}\sum_{i=1}^{N} p(x_k) * p(y_i|x_k) * \log_2\big(p(y_i|x_k)\big)$$

⇨ Wir können diese Werte aus der Kanalmatrix ablesen!
⇨ Wird auch «Streuentropie» genannt.

Transinformation

Beschreibt den maximalen, fehlerfreien Informationsfluss über einen Kanal.

$$\boldsymbol{T} = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

⇨ Bei einem ungestörten Kanal ist $T = 1$.
⇨ Bei einem vollständig gestörten Kanal ist $T = 0$.

Eigenschaften

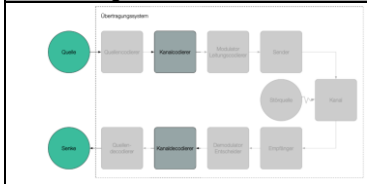
Bei einer Transinformation gilt:

- Verändert sich die Entropie der Quelle, verändert sich auch die Transinformation.
- Nimmt die Fehlerwahrscheinlichkeit zu, so verringert sich die Transinformation.

⇨ D.h. Die Transinformation wird durch die Quelle bestimmt.

Blockcodes

Einordnung



⇨ Kanalkodierung beinhaltet Blockcodes & Faltungscodes

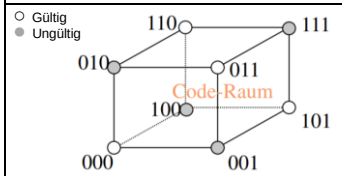
Was bedeutet Kanalkodierung?

Die Kanalkodierung hat das Ziel, bewusst Redundanz in eine Nachricht zu bringen, um den Fehlern bei der Datenübertragung entgegenzuwirken.

⇨ Wir teilen den Coderaum in gültige & ungültige Codeworte auf.

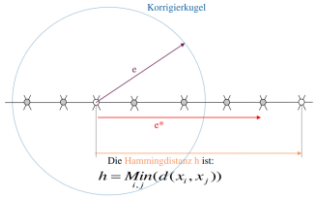
n-Dimensionale Coderaum

Der «Coderaum» beschreibt die Menge aller gültigen und ungültigen Codeworte. Wir können den «Coderaum» auch in einem Diagramm visualisieren.



⇨ Beispiel eines 3-dimensionalen Coderaums

Definitionen



⇨ Beispiel eines n-dimensionalen Coderaums

Hammingdistanz h

Beschreibt den minimalen Abstand zwischen zwei gültigen Codeworten im gesamten Coderaum.

$$h = \min_{i,j}\Big(d\big(x_i, x_j\big)\Big)$$

Anzahl erkennbare Fehler e^*

Beschreibt die «erkennbaren» Fehler bei einem ungültigen Codewort.

$$e^* = h - 1$$

Anzahl korrigierbare Fehler e

Beschreibt die «korrigierbaren» Fehler, sodass ein ungültiges Codewort noch dem korrekten, gültigen Codewort zugeordnet werden kann.

$$e = \frac{h-2}{2} \quad \boldsymbol{h = \text{Gerade}} \qquad e = \frac{h-1}{2} \quad \boldsymbol{h = \text{Ungerade}}$$

⇨ Treten mehr Fehler auf als korrigierbar sind, so wird entweder falsch korrigiert oder der Fehler wird nicht erkannt.

Dichtgepackt oder nicht?

Ein Coderaum ist «dichtgepackt», wenn sich alle Codeworte (gültig & ungültig) in einer Korrigierkugel befinden.

$$2^m \cdot \sum_{w=0}^{e}\binom{n}{w} \leq 2^n$$

Anzahl der CW bzw. Korrigierkugeln | Anzahl der CW pro Korrigierkugel = k | Anzahl aller CW

⇨ n: Codestellen, m: Nachrichtenstellen, k: Kontrollstellen

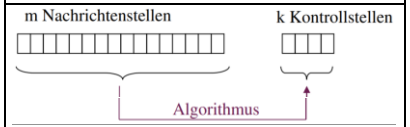
Ein Code ist «dichtgepackt», wenn gilt:

$$2^m * \sum_{w=0}^{e}\binom{n}{w} = 2^n$$

⇨ Auch grafisch durch Aufzeichnen der Korrigierkugeln lösbar.

Blockcodes

Bei Blockcodes werden die Codeworte in Nachrichten- und Kontrollstellen unterteil. Anschliessend wird ein Algorithmus definiert, der die Nachrichtenstellen auf die Kontrollstellen abbildet.



Wir definieren nun:

- n die Anzahl der Codestellen
- m die Anzahl der Nachrichtenstellen
- k die Anzahl der Kontrollstellen

Wobei ausserdem gilt:

- 2^n die Anzahl aller Codeworte
- 2^m die Anzahl gültige Codeworte
- Umrechnungen: $n = 2^k - 1 = m + k$

⇨ Bei Abramson-Codes gilt $2^{k-1} - 1$ (s. weiter unten)
⇨ Annahme: Bei Blockcodes gilt $h = 3$ (Abramson $h = 4$)

Gültigkeit eines Codewortes

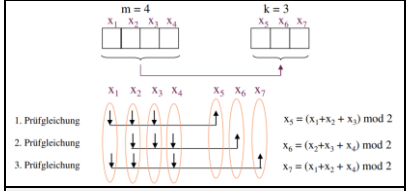
Der Algorithmus vom Blockcode erlaubt es uns, herauszufinden, ob ein Codewort gültig ist oder nicht:

- Gültige Codeworte: Erfüllen den Algorithmus und werden korrekt abgebildet
- Ungültige Codeworte: Erfüllen den Algorithmus nicht und liefern ein Fehlermuster.

Hamming-Code

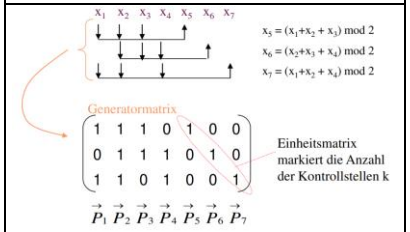
Beim Hamming-Code werden Gleichungen basierend auf den einzelnen Stellen des Codewortes definiert.

⇨ Ein Codewort ist gültig, wenn es alle diese Gleichungen erfüllt.



Generatormatrix

Die Hamming-Gleichungen lassen sich auch in einer Generatormatrix abbilden.



Formell können wir nun definieren:

$$\sum_i x_i * \vec{P}_i \equiv \vec{0} \bmod 2$$

Das bedeutet z.B. für die 1. Gleichung:

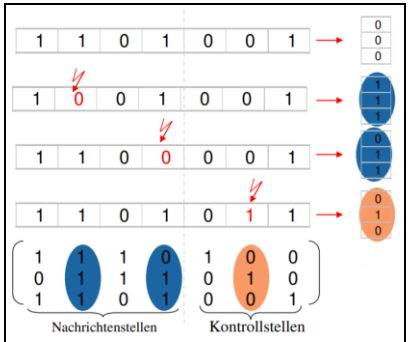
- Wenn: $x_5 = (x_1 + x_2 + x_3) \bmod 2$
- Dann: $0 = (x_1 + x_2 + x_3) \bmod 2 - x_5$
  $0 = (x_1 + x_2 + x_3 + x_5) \bmod 2$

⇨ D.h. durch die Aufsummierung der jeweiligen Spalten, erhalten wir bei einem gültigen Codewort einen Nullvektor.

Fehlersyndrom

Bei einem fehlerhaften Codewort liefert uns die obige Formel keinen Nullvektor, sondern genau die Spalte der Generatormatrix, in der ein Fehler aufgetreten ist.

⇨ Funktioniert nicht, wenn mehr als ein Fehler aufgetreten ist.



⇨ Formell können wir auch sagen: $\vec{Z} = \sum_i x_i * \vec{P}_i \bmod 2$

Zyklische Codes

Generatorpolynome

Die Generatormatrix lässt sich auch als Generatorpolynom beschreiben. Wir können diese in der Polynom- und Binärschreibweise notieren.

- Polynom: $G(u) = u^3 + u + 1$
- Binär: $G(u) = (g_3\, g_2\, g_1\, g_0) = (1\,0\,1\,1)$

⇨ Der höchste Grad bestimmt die Anzahl der Kontrollstellen.

Ermitteln der Kontrollstellen Polynom

Die Berechnung der Kontrollstellen einer gebenden Nachricht funktioniert über die Polynomdivision:

1. Beginne mit der Nachricht
2. Schreibe nun unter die erste 1 das Generatorpolynom aus der Aufgabe hin.
3. Berechne jede Stelle mit $(\ldots + \ldots) \bmod 2$
4. Wiederhole mit dem aktuellen Resultat, bis alle Kontrollstellen berechnet wurden.

	u^6	u^5	u^4	u^3	u^2	u^1	u^0
	1	0	0	0	▪	▪	▪
%	1	0	1	1			
	0	0	1	1	0	0	
			1	0	1	1	
			0	1	1	1	0
				1	0	1	1
				0	1	0	1

⇨ Nachricht: 1000, Generator: $G(u) = (1\,0\,1\,1)$

Codebedingung

Wir können nun über die Polynomdivision auch bestimmen, ob ein Codewort gültig ist oder nicht.

	u^6	u^5	u^4	u^3	u^2	u^1	u^0
	1	0	0	0	1	0	1
%	1	0	1	1			
	0	0	1	1	1	0	
			1	0	1	1	
			0	1	0	1	1
				1	0	1	1
				0	0	0	0

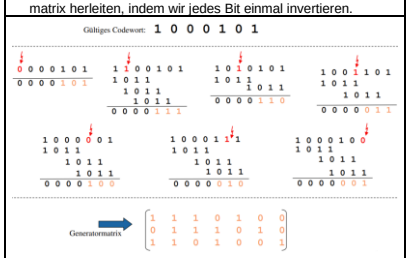
⇨ Codewort gültig!

⇨ Codewort: 1000101, Generator: $G(u) = (1\,0\,1\,1)$

Herleitung der Generatormatrix

Bei einem fehlerhaften Codewort liefert uns auch die Polynomdivision genau die Spalte, in der ein Fehler aufgetreten ist.

⇨ Wir können also mit einem gültigen Codewort die Generatormatrix herleiten, indem wir jedes Bit einmal invertieren.



⇨ Hinweis: Diese Darstellung der Polynomdivision wird als «Mehrfachaddition» bezeichnet, funktioniert aber identisch.

⇨ Extrahinweis: Ist die Anzahl der Kontrollstellen bekannt, so kann man sich deren Berechnung sparen (Einheitsmatrix).

Spezielle Codes

Zyklische Hamming-Code

Hammingdistanz h=3

Diese werden gebildet durch sogenannte primitive Polynome p(x) = g(x):

$p(x) = 1+x+x^3$
$p(x) = 1+x+x^4$
$p(x) = 1+x^2+x^5$
$p(x) = 1+x+x^6$
$p(x) = 1+x^3+x^7$
$p(x) = 1+x^2+x^3+x^4+x^5+x^6+x^7$
$p(x) = 1+x^2+x^3+x^4+x^5+x^8$
$p(x) = 1+x^4+x^9$
$p(x) = 1+x^3+x^{10}$
$p(x) = 1+x^2+x^{11}$
$p(x) = 1+x+x^4+x^6+x^{12}$
$p(x) = 1+x+x^3+x^4+x^{13}$
$p(x) = 1+x^2+x^6+x^{10}+x^{14}$
$p(x) = 1+x+x^{15}$
$p(x) = 1+x^5+x^{23}$
$p(x) = 1+x+x^2+x^4+x^5+x^7+x^8+x^{10}+x^{11}+x^{12}+x^{16}+x^{22}+x^{23}+x2^6+x^{32}$

Zyklische Abramson-Codes CRC-Codes

Hammingdistanz h=4

Diese werden gebildet durch die Multiplikation eines primitven Polynoms mit dem Term (1+x)

Abramson-Code: $g(x)= p(x)\,(1+x)$

Bsp.:

$g(x) = (1+x+x^3)\,(1+x)$
$g(x) = 1+x^2+x^3+x^4$

Faltungscodes

Bedeutung

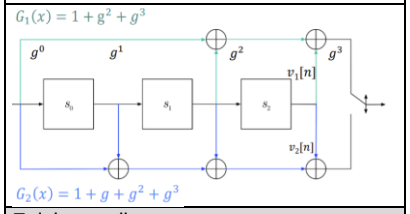
Faltungscodes erlauben die fortlaufende Codierung eines kontinuierlichen Datenstroms, wobei keine Blockbildung oder Synchronisation benötigt wird.

⇨ Gute Faltungscode werden mit Rechnersimulation gefunden.

Encoderschaltung

Bei Faltungscodes werden mehrere Generatorpolynome in eine Encoderschaltung abgebildet, wobei gilt:

- Jedes Generatorpolynom bildet eine «Linie»
- Der höchste Grad bestimmt die «Kastenzahl»
- Jeder Grad eines Polynoms bildet ein «⊕»



Zeichencodierung

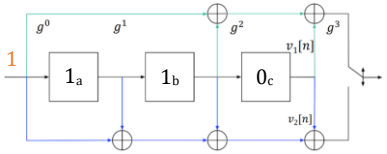
Wir nun ein zu codierendes Zeichen u von rechts in die Schaltung geschoben, so gilt für die Codierung:

- Jedes Generatorpolynom (also jede «Linie») erstellt ein Zeichen des Codes.
- Bei jedem «⊕» wird der jeweilige Kasteninhalt zum Zeichen u hinzuaddiert.
- Anschliessend werden die Kasteninhalte nach rechts geschoben.

⇨ Die «Kasten» werden mit 0 vorbelegt.

Diagrammbeispiel

Zeichen $u = 1$, Codierung: $v = 01$

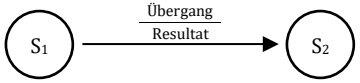


$v_1 = (1 + 1_b + 0_c) \bmod 2 = 0$
$v_2 = (1 + 1_a + 1_b + 0_c) \bmod 2 = 1$

⇨ Der anschliessende Kastenzustand wäre: $1_a\,1_b\,1_c$

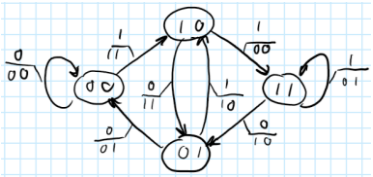
Zustandsdiagramm

Das Zustandsdiagramm beschreibt alle mögliche Kastenzustände einer Encoderschaltung, inklusive deren Übergänge und Codierungsresultate.



⇨ S_1: Anfangszustand des Kastens, S_2: Endzustand des Kastens

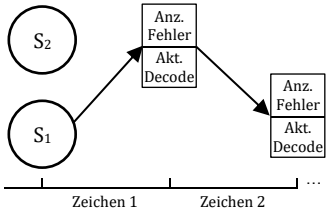
Diagrammbeispiel



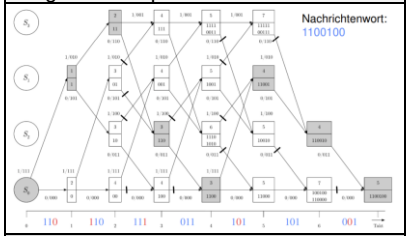
⇨ Herleitung: Zeichne zuerst eine Tabelle mit allen Kastenzuständen und berechne dann den Code und den Folgezustand.

Netzdiagramm

Das Netzdiagramm bezeichnet ein aufgespanntes Zustandsdiagramm bei einer Folge von Eingabezeichen. Wir können damit Zeichenketten decodieren.



Diagrammbeispiel



Wann ist ein Faltungscode «gut»?

Ein Faltungscode ist gut, wenn der Unterschied der Ausgabe bei einem Zustandsübergang immer maximal ist.

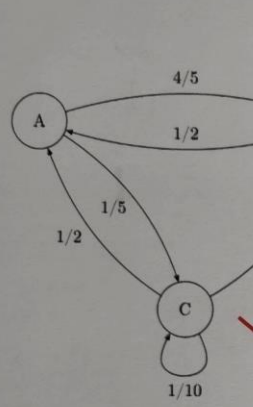
$S_1 = 00$: $u = 0 \longrightarrow v = 00$
$u = 1 \longrightarrow v = 11$

⇨ Maximaler Unterschied der Ausgaben (2 Zeichen)

⇨ Siehe «Zustandsdiagramm»-Bespiel für einen guten Code.

Anhang / Nachtrag

Beispiel: Diskrete Quelle mit Gedächtnis



ablesen

$\neq H(Y|X)$

$p(y\|x)$		zu $y=$ A	B	C	
Von $x=$	A	0 (1)	4/5 (2)	1/5 (3)	= 1
	B	1/2	1/2	0	= 1
	C	1/2	2/5	1/10	= 1

berechnen

berechnen

$H(X)$ *

$xi=$	$p(x)$
A	9/27 (a)
B	16/27
C	2/27

berechnen

$H(X,Y)$

$p(x,y)$		$y=$ A	B	C
$x=$	A	$0_{1\cdot a}$	$4/15_{2\cdot a}$	$1/15_{3\cdot a}$
	B	8/27	8/27	0
	C	1/27	4/135	1/135

$$H(X,Y) = \sum_{i=1}^{N}\sum_{k=1}^{N} p(x_i, y_k) \cdot \log_2\left(\frac{1}{p(x_i, y_k)}\right)$$

* $P(A) = 0 \cdot P(A) + 1/2 \cdot P(B) + 1/2 \cdot P(C)$
$P(B) = 4/5 \cdot P(A) + 1/2 \cdot P(B) + 2/5 \cdot P(C)$
$P(C) = 1/5 \cdot P(A) + 0 \cdot P(B) + 1/10 \cdot P(C)$
$1 = P(A) + P(B) + P(C)$

Spaltenweise ↓

Solve()

OST

---

maximale fehlerfreie Übertragungsrate
= T · Übertragungsrate (z.B in MBit/s)

Tailbits: Die Anzahl Bits, um ein Faltungscodierer wieder in den Anfangszustand zu bringen.

⇒ Anzahl Kasten = Anzahl Tailbits

⇒ Tailbits sind immer 0 z.B 0001 : 1 Startbit, 3 Tailbits

Reduzibel oder Irreduzibel?
Generatorpolynom durch $(x+1)$ teilen:

$x^7 + x^6 + x^2 + 1 \quad : x+1$
$-(x^7 + x^6) \quad \cdot x^6$
$= x^2 + 1 \quad : x+1$
$-(x^2 + x) \quad \cdot x$
$= -x + 1 \quad : x+1$
$-(-x-1) \quad \cdot -1$
$= 2$

⇒ Rest heisst reduzibel!

Reduzibel: CRC- / Abramson-Code
Irreduzibel: Hamming-Code