

Digitale Codierungen

Zusammenfassung

Damien Flury

Juni/Juli 2024

Inhaltsverzeichnis

1 Das Stellenwertsystem	3
1.1 Begriffe	3
1.2 Polynomschreibweise	3
1.3 Dezimalsystem	3
1.4 Dualsystem	3
1.4.1 Umerchnung ins Dualsystem	3
1.4.2 Umrechnung einer Dezimalzahl ins Dualsystem	3
1.4.3 Beispiel	4
1.5 Oktalsystem	4
1.5.1 Beispiel	4
1.6 Kommazahlen	4
1.7 Subtrahieren durch Addieren	4
1.7.1 Additive Zahl berechnen	4
1.8 Dualzahlen	5
1.9 Multiplikation	5
1.9.1 Unsigned	5
1.9.2 Signed	5
1.10 Indexschreibweise	5
1.11 Subtrahieren	5
1.11.1 Betrag mit Vorzeichen	5
1.11.2 Einerkomplement ($b-1$)	5
1.11.3 Zweierkomplement (b -Komplement)	6
1.12 Allgemeine Berechnung des b -Komplements	6
1.13 Darstellung von negativen Zahlen	6
1.13.1 Vorzeichen und Betrag (Sign and Magnitude)	6
1.13.2 Einerkomplement ($b - 1$)	6
1.13.3 Zweierkomplement (b)	7
1.13.4 Excesscode	7
1.14 Präfixe	7
1.15 Fixkommazahlen	8
1.15.1 Absoluter Fehler	8
1.15.2 Relativer Fehler	8
1.16 Gleitkommazahlen	8
1.16.1 Standard IEEE 754	8
1.16.2 Addition	8
2 Interpretation eines Codewortes	9
2.1 Interpretation als Vektor	9
2.2 Interpretation als Polynom	9
3 Gruppe, Ring und Körper	9

3.1 Verknüpfung	10
3.1.1 Neutrales Element	10
3.1.2 Inverses Element	10
3.2 Gruppe	10
3.3 Ring	10
3.4 Körper	10
3.5 Zyklische Gruppe	11
3.5.1 Idee von Galois	11
4 Boolsche Algebra	12
4.1 Minterme und Maxterme	12
4.2 Addition	12
4.2.1 Halbaddierer	12
4.2.2 Volladdierer	13
4.3 Kanonische Disjunktive Normalform (KDNF)	13
5 Wahrscheinlichkeit	13
5.1 Begriffe	13
5.2 Laplace	13
5.3 Verbundwahrscheinlichkeit	13
5.4 Kombinatorik	13
5.4.1 Übersicht	13
5.4.2 Geordnete Proben (ohne Wiederholung)	14
5.4.3 Permutation	14
5.4.4 Ungeordnete Proben	14
5.4.5 Binomialverteilung	14
6 Informationstheorie	14
6.1 Entscheidungsgehalt	15
6.2 Informationsgehalt	15
6.2.1 Tatsächliche Codewortlänge L	15
6.3 Entropie	15
6.3.1 Mittlere Codewortlänge	15
6.3.2 Quelle mit Gedächtnis	15
6.3.3 Wann wird die Entropie einer Quelle/Senke maximal?	15
6.3.4 Shannon'sches Codierungstheorem	16
6.4 Redundanz einer Quelle	16
6.5 Präfixeigenschaft	17
6.6 Quelle mit Gedächtnis	17
7 Kompressionsmethoden	18
7.1 Lauflängenkomprimierung	18
7.2 Lempel-Ziv-Verfahren	18
7.2.1 Lempel-Ziv-Welch (LZW)	18
7.2.2 Lempel-Ziv für Binärzahlen	19
7.3 Kombination Lempel-Ziv und Huffman	19
8 Kryptologie	20
8.1 RSA	20
8.1.1 Eulerfunktion	20
8.1.2 Euklidischer Algorithmus	21
9 Kanalmodell	21
9.1.1 3x3-Matrix	21
9.1.2 Symmetrischer Kanal	22

9.1.3 Verbundentropie	22
9.1.4 Äquivokation (Verlust)	22
9.1.5 Irrelevanz (Rauschen)	22
9.1.6 Transinformation	22
9.1.7 Maximum-Likelihood-Verfahren	22
9.2 Fehlerkorrektur	23
9.2.1 Korrigierkugel	24
9.2.2 Paritätsbit-Code	25
9.2.3 Hammingcode	25
9.2.4 Ermitteln der Kontrollstellen durch rückgekoppeltes Schieberegister	26
9.2.5 Abramson- bzw. CRC-Code	27
9.3 Codierungsdichte	27
9.4 Fehlersyndrom	27
9.5 Zyklischer Hamming-Code	27
9.5.1 Ermitteln der Kontrollstellen durch Mehrfachaddition	28
9.6 Faltungscodes	29
9.6.1 Zustandsdiagramm	30

1 Das Stellenwertsystem

1.1 Begriffe

- Nibble: Binärzahl mit 4 Bit
- Oktett: Binärzahl mit 8 Bit (Byte)

1.2 Polynomschreibweise

$d_n = \text{Ziffer} \in Z_n, R^n = \text{Wertigkeit}$

$$N_n = d_n R^n + d_{n-1} R^{n-1} + \dots + d_1 R^1 + d_0 R^0$$

$$\text{Beispiel: } 123 = 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0$$

Basis \Leftrightarrow Wertigkeit

Potenz \Leftrightarrow Position der Stelle

1.3 Dezimalsystem

$R_{10} = 10$ (Basis); $Z_{10} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

1.4 Dualsystem

$R_2 = 2$ (Basis); $Z_2 = \{0, 1\}$

1.4.1 Umerchnung ins Dualsystem

$$\begin{array}{r}
 \frac{25}{2} = 12 \text{ Rest } 1 \\
 \frac{12}{2} = 6 \text{ Rest } 0 \\
 \uparrow \frac{6}{2} = 3 \text{ Rest } 0 \\
 \frac{3}{2} = 1 \text{ Rest } 1 \\
 \frac{1}{2} = 0 \text{ Rest } 1
 \end{array} \tag{1}$$

$$= 11001$$

1.4.2 Umrechnung einer Dezimalzahl ins Dualsystem

$$\downarrow \begin{array}{rcl} 0.75 \cdot 2 = 1.5 & \text{Stelle } 1 \\ 0.5 \cdot 2 = 1 & \text{Stelle } 1 \\ 0 \cdot 2 = 0 & \text{Stelle } 0 \end{array} \quad (2)$$

$= 0.11$

1.4.3 Beispiel

$$N_2 = 110 \ N_2 = 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 4_d + 2_d = 6_d$$

1.5 Oktalsystem

$$R_8 = 8 \ (\text{Basis}); Z_8 = \{0, 1, 2, 3, 4, 5, 6, 7\}$$

1.5.1 Beispiel

$$N_8 = 110 \ N_8 = 1 \cdot 8^2 + 1 \cdot 8^1 + 0 \cdot 8^0 = 72_d$$

1.6 Kommazahlen

$$\mathbb{R}_{10} = 110.13 \ N_{10} = 1 \cdot 10^2 + 1 \cdot 10^1 + 0 \cdot 10^0 + 1 \cdot 10^{-1} + 3 \cdot 10^{-2}$$

$$\mathbb{R}_2 = 101.110 \ N_2 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} = 5.75_d$$

1.7 Subtrahieren durch Addieren

Annahme: Bei 1000 gibt es einen **Überlauf**.

$$753 + 247 = 0, \text{ daraus folgt } 753 \equiv -247$$

$$\text{Somit ist } 620 - 247 \equiv 620 + 753 = 1373 \equiv 373.$$

1.7.1 Additive Zahl berechnen

Gesucht: Additive Zahl von -247 (, also 753).

$$\begin{aligned} 999 - 247 &= 752 \ (\text{Neunerkomplement}) \\ 752 + 1 &= 753 \ (\text{Zehnerkomplement}) \end{aligned} \quad (3)$$

Beispiel:

$$\begin{aligned} 760 - 233 &=? \\ 999 - 233 &= 766 \\ 760 + 767 &= 1527 \quad \text{Überlauf bei 1000} \\ 527 &= 760 - 233 \end{aligned} \quad (4)$$

Beispiel (Zweiersystem):

$$\begin{aligned} 10110 - 01010 &=? \\ 11111 - 01010 &= 10101 \\ 10110 + 10110 &= 101100 \quad \text{Überlauf} \\ 10110 - 01010 &= 01100 \end{aligned} \quad (5)$$

Beispiel (Negatives Resultat):

$$\begin{aligned} 0101 - 0110 &=? \\ -0110 \Rightarrow 1001 + 1 &= 1010 \\ 0101 + 1010 &= 1111 \end{aligned} \quad (6)$$

-1 ist also 1111

1.8 Dualzahlen

-1:

$$1 = 0001_2.$$

Einerkomplement: 1110_2

Zweierkomplement: $1111_2 = -1$

1.9 Multiplikation

1.9.1 Unsigned

Die unsigned Multiplikation ist eine Summe von Links-Shifts.

$$a = 3, b = 5$$

$$\begin{aligned} 0011 \cdot 0101 &= 0101 + 1010 \\ &= 1111 = 15_d \end{aligned} \tag{7}$$

Oder durch Polynommultiplikation:

$$(2^1 + 2^0) \cdot (2^2 + 2^0) = 2 \cdot 4 + 2 \cdot 1 + 1 \cdot 4 + 1 \cdot 1 = 15 \tag{8}$$

1.9.2 Signed

Die signed Multiplikation funktioniert analog zur unsigned Multiplikation, aber wenn einer der Operanden negativ ist, muss das Zweierkomplement davon gebildet werden:

Beispiel (Signed):

$$1101 * 0111 ((-3) \cdot 7 = -21)$$

1101 ist negativ, das Zweierkomplement ist 0011.

$$0011 * 0111 = 0111 + 01110 = 010101. \text{ Das Zweierkomplement davon ist } 101011 (= -21).$$

1.10 Indexschreibweise

$$b = 1010$$

$$b_3 = 1, b_2 = 0, b_1 = 1, b_0 = 0$$

$$b_{3..1} = 101, b[3..1] = 101$$

1.11 Subtrahieren

1.11.1 Betrag mit Vorzeichen

$$\begin{array}{r} 5 - 1 \\ 5 = 0101 \\ -1 = 1001 \\ \hline 0101 - 0001 = 0100 = 4 \end{array} \tag{9}$$

1.11.2 Einerkomplement (b-1)

Von negativen Zahlen wird das Einerkomplement gebildet. Gibt es nach Addition einen Überlauf, muss noch +1 gerechnet werden.

$$\begin{aligned}
 & 5 - 1 \\
 & 5 = 0101 \\
 & 1 = 0001, -1 = 1110 \\
 & 0101 + 1110 = 10011 \\
 \Rightarrow & \text{Überlauf} \Rightarrow 0011 + 1 = 0100 = 4
 \end{aligned} \tag{10}$$

1.11.3 Zweierkomplement (b-Komplement)

$$\begin{aligned}
 & 5 - 1 \\
 & 5 = 0101 \\
 & 1 = 0001, -1 = 1111 \\
 & 0101 + 1111 = 10100 \Rightarrow \text{Überlauf} \Rightarrow 0100 = 4
 \end{aligned} \tag{11}$$

1.12 Allgemeine Berechnung des b-Komplements

$$C_{b,n}(N) = b^n - N \tag{12}$$

- n = Anzahl Stellen
- b = Basis
- N = Zahl, von welcher das Komplement gebildet werden soll

Beispiel:

$$\bullet C_{8,2}(6) = 8^2 - 6 = 58_{10} = 72_8$$

Im Dualsystem entspricht dies dem Zweierkomplement.

1.13 Darstellung von negativen Zahlen

1.13.1 Vorzeichen und Betrag (Sign and Magnitude)

- Vorzeichenbit (Sign): 0 für positive, 1 für negative Zahlen
- Betrag (Magnitude): absoluter Wert der Zahl

Beispiel:

- Positive Zahl (+5): 0101
- Negative Zahl (-5): 1101

→ Suboptimal: Wir haben zwei Nullen (+0 und -0)

- Einfach, intuitiv
- Zwei Darstellungen von Null
- Weniger effizient für mathematische Operationen (zwei Prozessschritte, Vorzeichen & Operation mit Beträgen)
- Anwendung: Vorallem im Bildungskontext

1.13.2 Einerkomplement (b - 1)

Es werden einfach alle Bits invertiert im Binärsystem

Beispiel: $+5 = 0101 \Rightarrow -5 = 1010$

Im allen Zahlensystem allgemein bildet man das Einerkomplement, indem man von der Basis 1 und die jeweilige Ziffer subtrahiert:

Beispiel: Sei $b = 5$, Zahl: 234_5 . Das Einerkomplement wird berechnet durch $b - 1 - z$, wobei z die Ziffer ist.

$$\begin{aligned}
 (5 - 1 - 2) &= 2 \\
 (5 - 1 - 3) &= 1 \\
 (5 - 1 - 4) &= 0
 \end{aligned} \tag{13}$$

Das Einerkomplement von 234_5 wäre also 210_5 .

- Suboptimal: Immernoch zwei Nullen (0 und -0)
- Berechnung effizienter, aber bei einem Überlauf bei Addition muss ein *End-Around-Carry* angewendet werden, d.h. wir entfernen und addieren den Überlauf zur Zahl hinzu

1.13.3 Zweierkomplement (b)

Um das Zweierkomplement zu erstellen, addieren wir 1 zum Einerkomplement

- Nur noch eine 0
- Effizient für Rechnen

1.13.4 Excesscode

Der Exzess- oder Überschusscode verschiebt den Zahlenstrahl auf negative Zahlen. So ist z.B. bei Exzess-8 die $0000 = -8$. Die tatsächliche Zahl kann errechnet werden, indem man eine gegebene binäre Zahl minus den Exzess rechnet:

$$\begin{array}{c}
 0101 \text{ in Exzess-8} \\
 0101_b = 5 \rightarrow 5 - 8 = -3
 \end{array} \tag{14}$$

Somit entspricht 0101 in Exzess-8: -3.

Man schreibt $C_{\text{ex},-q,n}(x)$, wobei:

- $-q$: Bias
- n : Länge der binären Schreibweise
- x : Zahl (normalerweise im Dezimalsystem)

Der Excesscode, der den Zahlenbereich in zwei gleich grosse Hälften aufteilt, hat einen besonderen Stellenwert. Dabei gehört die 0 zu den negativen Zahlen. Beispiel: Bei vierstelligen Codes würde der Excess-7-Code $C_{\text{Ex},-7,4}(x)$ die Zahlen von -7 bis 8 darstellen. Der Bias ergibt sich dann durch $2^{n-1} - 1$.

Um eine Zahl a zu kodieren, wählt man die kleinste Zahl b im Wertebereich und bildet die Differenz $d = |a - b|$. Beispiel:

$$\begin{array}{l}
 C_{\text{ex},-4,3}(-1) = ? \\
 d = |-1 - (-4)| = 3 \Rightarrow 011
 \end{array} \tag{15}$$

1.14 Präfixe

2^{10}	$1.024 \cdot 10^3$	K Kilo	Ki Kibi
2^{20}	$1.049 \cdot 10^6$	M Mega	Mi Mebi
2^{30}	$1.074 \cdot 10^9$	G Giga	Gi Gibi
2^{40}	$1.100 \cdot 10^{12}$	T Tera	Ti Tebi
2^{50}	$1.126 \cdot 10^{15}$	P Peta	Pi Pebi
2^{60}	$1.153 \cdot 10^{18}$	E Exa	Ei Exbi

Beispiel: Wie viele Datensätze zu je 128 Byte passen in einen Speicherbereich der Grösse 4 KB?

$$128B = 2^7B, 4KB = 2^2 \cdot 2^{10}B$$

$$\frac{2^2 \cdot 2^{10}B}{2^7B} = 2^5 = 32 \quad (16)$$

1.15 Fixkommazahlen

$C_{FK,k,n}(x)$, wobei

- k = Anzahl Nachkommastellen
- n = Länge der binären Schreibweise

Beispiel: $C_{FK,4,16}(453.1234) = 0001'1100'0101'0001$

1.15.1 Absoluter Fehler

$$E_{abs} = |x_{korrekt} - x_{gerundet}|$$

1.15.2 Relativer Fehler

$$E_{rel} = \frac{|x_{korrekt} - x_{gerundet}|}{x_{korrekt}}$$

1.16 Gleitkommazahlen

Bei Gleitkommazahlen wird zusätzlich zum Bitmuster z auch die Stelle k mitgeführt, an der das Komma steht.

- z = Signifikand, Mantisse
- k = Exponent
- $C_{GK,k,n}(z) = z \cdot 2^k$

Beispiel: 6.25.

$$6 = 0110, 0.25 = 0.01$$

Fixkommarepräsentation: 0110.01

Gleitkommazahlrepräsentation: 1.1001 · 2²

Für die Mantisse wird die Excess-Darstellung verwendet, d.h. bei 8-Bit-Mantisse der $C_{Ex,-127,8}$. Der Exponent 2 wird so zu 10000001.

Als 32-Bit-Gleitkommazahl: 0'10000001'10010000000000000000000000000000

0	10000001	10010000000000000000000000000000
Vorzeichen	Exponent	Mantisse

1.16.1 Standard IEEE 754

- Single: 24 Bit Präzision, 8 Bit Exponent
- Double: 53 Bit Präzision, 11 Bit Exponent
- Quadruple: 113 Bit Präzision, 15 Bit Exponent

1.16.2 Addition

1. Wenn Vorzeichen unterschiedlich: Subtraktion
2. Hidden Bits ergänzen
3. Wenn Exponenten unterschiedlich: Signifikand der kleineren Zahl um entsprechend viele Bits nach rechts verschieben
4. Addition durchführen
5. Falls Carry = 1: Ergebnis normalisieren:
 - Exponent um 1 erhöhen
 - Signifikand um 1 nach rechts schieben

- $x = 1.5, y = 0.75$
- $x = 0 | 0111\ 1111 | 100\ 0000\ 0000\ 0000\ 0000\ 0000$
- $y = 0 | 0111\ 1110 | 100\ 0000\ 0000\ 0000\ 0000\ 0000$
- $x' = 0 | 0111\ 1111 | (1) 100\ 0000\ 0000\ 0000\ 0000\ 0000$ mit hidden Bit
- $y' = 0 | 0111\ 1111 | (0) 110\ 0000\ 0000\ 0000\ 0000\ 0000$ m.h.B., $a + 1, m \cdot 2^{-1}$
- $z' = 0 | 0111\ 1111 | (10) 010\ 0000\ 0000\ 0000\ 0000\ 0000 = x' + y'$
- $z'' = 0 | 1000\ 0000 | (1) 001\ 0000\ 0000\ 0000\ 0000\ 0000 a + 1, m \cdot 2^{-1}$
- $z = 0 | 1000\ 0000 | 001\ 0000\ 0000\ 0000\ 0000\ 0000$ ohne hidden Bit
- $z = 2.25$

Abbildung 1: Floating-Point-Addition

2 Interpretation eines Codewortes

Ein Codewort 1001 kann auf verschiedene Arten interpretiert werden:

- Als Tupel $(1, 0, 0, 1)$
- Als Zahl $1001_2 = 9_{10}$. Es gelten die üblichen Operationen der Ganzzahlrechnung.
- Als Vektor $\begin{pmatrix} 1 & 0 & 0 & 1 \end{pmatrix}^T$. Es gelten die üblichen Operationen der Vektorrechnung.
- Als Polynom: $g(u) = u^3 + 1$. Es gelten die üblichen Operationen der Polynomrechnung.

Die obigen Darstellungsformen sind äquivalent und beschreiben alle dasselbe Codewort. Alle Berechnungen erfolgen in \mathbb{Z}_2 .

2.1 Interpretation als Vektor

Vektorraum \mathbb{Z}_2^3 :

$$\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \equiv \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \text{ mod } 2 \quad (17)$$

Vektoren werden in der Codierung zur Fehlererkennung und -behebung verwendet.

2.2 Interpretation als Polynom

Codewort 100101 als Polynom:

$$1u^5 + 0u^4 + 0u^3 + 1u^2 + 0u^1 + 1u^0 = u^5 + u^2 + u^0 \quad (18)$$

Multiplikation zweier Polynome in \mathbb{Z}_2 :

$$\begin{aligned} & (u^5 + u^2 + u^0)(u^2 + u^0) \text{ mod } 2 \\ &= (u^7 + u^5 + u^4 + 2u^2 + u^0) \text{ mod } 2 \\ &= u^7 + u^5 + u^4 + 1 \end{aligned} \quad (19)$$

Das resultierende Codewort ist 1011 0001.

3 Gruppe, Ring und Körper

Verknüpfung → Gruppe → Ring → Körper

3.1 Verknüpfung

Menge m und Operation w : (M, w)

Bsp: $(\mathbb{N}, +)$

3.1.1 Neutrales Element

Beispiel:

$$\begin{aligned} (\mathbb{N}, +) &\rightarrow 0 \\ (\mathbb{N}, \cdot) &\rightarrow 1 \end{aligned} \tag{20}$$

3.1.2 Inverses Element

$a \odot a' = n$

Beispiel:

$$\begin{aligned} (\mathbb{Z}, +) : 7 + (-7) &= 0 \\ (\mathbb{R}, \cdot) : 4 \cdot \frac{1}{4} &= 1 \end{aligned} \tag{21}$$

3.2 Gruppe

Verknüpfung (G, w) mit

- Bezügl. w abgeschlossen
- w assoziativ
- Neutrales Element existiert
- Inverses Element existiert

Zusatz: Wenn Kommutativ \rightarrow abelsche Gruppe (Bsp. $(\mathbb{Z}, +)$)

3.3 Ring

Zwei Operationen $R(M, w_1, w_2)$, wobei (M, w_1) abelsche Gruppe.

3.4 Körper

Ein Ring ist ein Körper, wenn:

- a) Jedes Element des Rings (ausser der 0) ein multiplikatives Inverses hat
- b) Die Multiplikation kommutativ ist: $ab = ba$ (Abel'sche Gruppe)
- c) Das Distributivgesetz gilt: $a(1 - b) = a - ab$
- d) Wenn er eine 1 hat (multiplikatives neutrales Element)

Wir definieren Mengen wie:

- $\mathbb{Z}_2 = \{0, 1\}$
- $\mathbb{Z}_5 = \{0, 1, 2, 3, 4\}$
- $\mathbb{Z}_M = \{0, 1, 2, \dots, M - 1\}$

Wir bewegen uns haupsächlich in $\mathbb{Z}_2 = \{0, 1\}$ und definieren folgende Operationen:

- Addition $+$,
 - Subtraktion $-$,
 - Multiplikation \cdot ,
 - Ganzzahldivision mit Rest (Restklassen)
 - Um die Abgeschlossenheit sicherzustellen, verwenden wir Operationen zusammen mit Modulo M.
- Beispiel:

$$\begin{aligned}\mathbb{Z}_5 &= \{1, 2, 3, 4, 5\} \\ 3 \cdot 4 &= 12, 12 \notin \mathbb{Z}_5 \\ 12 \bmod 5 &= 2, 2 \in \mathbb{Z}_5\end{aligned}\tag{22}$$

Modulo von negativen Zahlen: $-1 \equiv 1 \bmod 2, -4 \equiv 1 \bmod 5$

Codewörter können als Elemente eines endlichen Ganzzahlkörpers betrachtet werden. In der Informatik bewegen wir uns in \mathbb{Z}_2 . Die Anzahl der darstellbaren Codewörter wird durch die Codewortlänge bestimmt.

- Byte: 8 Bit
- Word: 16, 32, 64 Bit
- TCP-Paket: 1024 Bit

Im endlichen Ganzzahlkörper gibt es immer eine grösste und eine kleinste Zahl. Die Darstellung dieser Zahl kann durch Speicher oder Definition der Wortgrösse begrenzt werden → Keine Unendlichkeit.

3.5 Zyklische Gruppe

Definition 3.5.1: In der Gruppentheorie ist eine zyklische Gruppe eine Gruppe, die von einem einzelnen Element erzeugt wird. Sie besteht nur aus Potenzen des Erzeugers.

Veranschaulichung:

$$\begin{aligned}M &= \{0^\circ, 90^\circ, 180^\circ, 270^\circ\} \\ \text{Verknüpfung} &= \text{Drehung } \bmod 360^\circ = a \\ 0^\circ \cdot a^4 &\equiv 0^\circ \bmod 360^\circ\end{aligned}\tag{23}$$

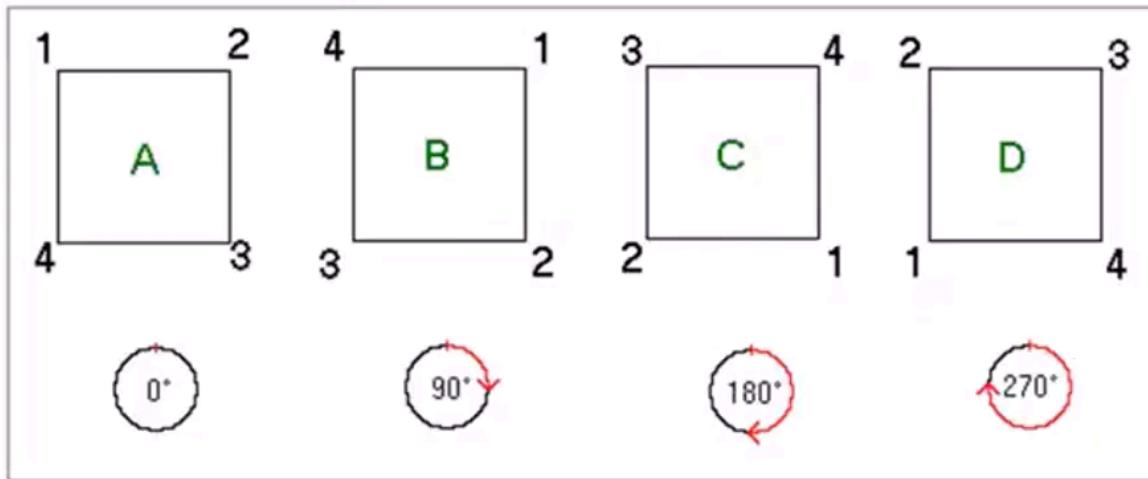


Abbildung 2: Zyklische Gruppe

Polynom $f(x) = x^3 + x + 1$, hat nach Fundamentalsatz der Algebra 3 Nullstellen. Die Frage ist nun, ob das Polynom in \mathbb{Z}_2 eine Lösung hat.

3.5.1 Idee von Galois

Polynom $f(x) = x^3 + x + 1$ hat keine Nullstelle in \mathbb{Z}_2 . Also nehmen wir als (imaginäres) *erzeugendes Element* a :

$$f(a) = a^3 + a + 1 = 0\tag{24}$$

Eine zyklische Gruppe (gemäss Évariste Galois (1811 - 1832)):

- Wird von einem einzigen Element erzeugt
- Besteht nur aus Potenzen des Erzeugers
- Das erzeugende Element a wird als Lösung eingesetzt: $f(a) = a^3 + a + 1$

Beispiel:

$$\begin{aligned}
 a &= a \\
 a^2 &= a^2 \\
 a^3 &= a + 1 && \text{Umstellung } f(a) \text{ in } \mathbb{Z}_2 \\
 a^4 &= a(a + 1) = a^2 + a \\
 a^5 &= a(a^2 + a) = a^3 + a^2 = a^2 + a + 1 && (25) \\
 a^6 &= a(a^2 + a + 1) = a^2 + 2a + 1 = a^2 + 1 \\
 a^7 &= a(a^2 + 1) = 1 \\
 a^8 &= a && \text{Zyklus beginnt von vorne}
 \end{aligned}$$

Neue Menge ist gemäss Gleichung 25:

$$\mathbb{Z}'_2 = \{0, 1, a, a^2, a + 1, a^2 + a, a^2 + a + 1, a^2 + 1\} \quad (26)$$

Diese Menge können wir codieren:

$$\{000, 001, 010, 100, 011, 110, 111, 101\} \quad (27)$$

4 Boolesche Algebra

Es gibt genau 16 binäre Funktionen (2^4).

x	y	AND				XOR	OR	NOR	EQUIV	$y \Rightarrow x$	$x \Rightarrow y$	NAND			
		0	xy	$x\bar{y}$	$\bar{x}y$	y	$x\bar{y} \vee \bar{x}y$	$x \vee y$	$\overline{x \vee y}$	$xy \vee \bar{y}$	\bar{y}	$x \vee \bar{y}$	$\bar{x} \vee y$	\overline{xy}	1
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	1	1	1	0	0	0	1	1	1	1	1
1	0	0	0	1	1	0	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

NAND wird auch bezeichnet als: $x \mid y = \overline{x \wedge y} = \overline{xy}$.

$$x \oplus y = (\neg x \wedge y) \vee (x \wedge \neg y) = (x \mid \neg y) \mid (\neg x \mid y)$$

4.1 Minterme und Maxterme

Minterme haben als Ergebnis immer 1.

Maxterme haben als Ergebnis immer 0.

4.2 Addition

XOR bildet die Addition zweier Bits, AND den Übertrag.

4.2.1 Halbaddierer

- Kann 2 einstellige Binärzahlen addieren
- Hat 2 Eingänge (x, y) und 2 Ausgänge

4.2.2 Volladdierer

- Hat 3 Eingänge (x, y, C)

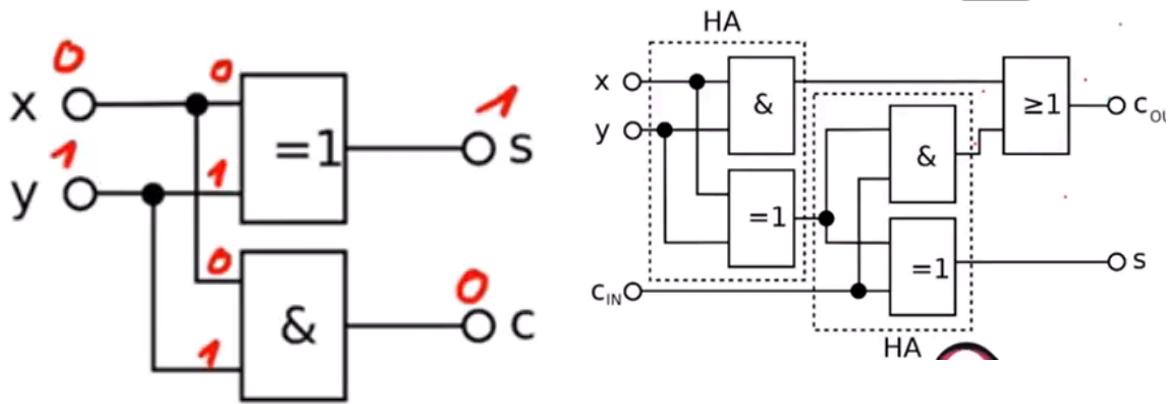


Abbildung 3: Half adder und full adder

4.3 Kanonische Disjunktive Normalform (KDNF)

Bei der KDNF kommen in jedem Minterm alle Variablen vor:

$$\begin{aligned}
 & (A \wedge B) \vee (A \wedge C) \\
 \Leftrightarrow & (A \wedge B \wedge (C \vee \neg C)) \vee (A \wedge C \wedge (B \vee \neg B)) \\
 \Leftrightarrow & (A \wedge B \wedge C) \vee (A \wedge B \wedge \neg C) \vee (A \wedge B \wedge C) \vee (A \wedge \neg B \wedge C) \\
 \Leftrightarrow & (A \wedge B \wedge C) \vee (A \wedge B \wedge \neg C) \vee (A \wedge \neg B \wedge C)
 \end{aligned} \tag{28}$$

5 Wahrscheinlichkeit

5.1 Begriffe

- Ereignismenge Ω : Menge aller möglichen Ausgänge (Ergebnisse) eines Zufallsvorgangs
- Ergebnis ω : Ein einzelnes Element $\omega \in \Omega$

Beispiel (Werfen eines Würfels): $\Omega = \{1, 2, 3, 4, 5, 6\}$

Beispiel (Zweifaches Werfen einer Münze): $\Omega = \{ZZ, ZK, KZ, KK\}$

Beispiel (Werfen einer Münze so lange, bis Kopf erscheint): $\Omega = \{K, ZK, ZZK, ZZZK, \dots\}$

- Anzahl aller Ergebnisse $|\Omega|$
- Wahrscheinlichkeit des Komplementärergebnisses: $P(\overline{A}) = 1 - P(A)$
- Wahrscheinlichkeit des unmöglichen Ereignisses: $P(\emptyset) = 0$
- Wertebereich der Wahrscheinlichkeit: $0 \leq P(A) \leq 1$

5.2 Laplace

$$P(A) = \frac{\text{Anzahl der günstigen Ergebnisse } A}{\text{Anzahl aller Ergebnisse } \Omega} = \frac{|A|}{|\Omega|}$$

5.3 Verbundwahrscheinlichkeit

$$p(a, b) = p(a) \cdot p(b|a) \tag{29}$$

5.4 Kombinatorik

5.4.1 Übersicht

	Beachtung der Reihenfolge
--	---------------------------

n-Optionen auswählen		MIT	OHNE
Zurücklegen	MIT	$A = n^k$	$A = \binom{n+k-1}{k}$
	OHNE	$A = n(n-1)\dots(n-k+1)$ $A = \frac{n!}{(n-k)!}$	$A = \frac{n!}{k!(n-k)!} = \binom{n}{k}$

5.4.2 Geordnete Proben (ohne Wiederholung)

Die Anzahl der k -Tupel aus einer n -Menge:

$$\begin{aligned}
& n \cdot (n-1) \cdot (n-2) \cdot \dots \cdot (n-k+1) \\
&= \frac{n!}{(n-k)!} \\
&= \prod_n^{n-k+1} n
\end{aligned} \tag{30}$$

5.4.3 Permutation

Die Zahl der Permutationen einer n -Menge ist $n!$. Dies ist ein Spezialfall von Gleichung 30 mit $n = k$.

5.4.4 Ungeordnete Proben

Die Anzahl der k -elementigen Teilmengen aus einer n -elementigen Menge ist (Binomialkoeffizient):

$$\binom{n}{k} = \frac{\prod_n^{n-k+1} n}{k!} = \frac{n!}{k!(n-k)!} \tag{31}$$

5.4.5 Binomialverteilung

$$P_x = \binom{n}{x} \cdot p^x \cdot q^{n-x} \tag{32}$$

Beispiel (Bitstream):



Wie gross ist die Wahrscheinlichkeit, dass **genau** 1 Bit gestört ist?

Die Wahrscheinlichkeit für eine Störung ist $p = 0.1$. Somit ist die Wahrscheinlichkeit für ein korrektes Bit $q = 0.9$.

$$P_1 = \binom{5}{1} \cdot p^1 \cdot q^4 \tag{33}$$

Wahrscheinlichkeit für zwei Fehler:

$$P_2 = \binom{5}{2} \cdot p^2 \cdot q^3 \tag{34}$$

6 Informationstheorie

Nachricht	redundant	nicht-redundant
irrelevant	Zeichenvorrat bei Quelle und Senke verschieden	
relevant	vorhersagbar	Information

6.1 Entscheidungsgehalt

$$H_0 = \log_2(N) \quad [\text{bit}], N = \text{Anzahl Elemente} \quad (35)$$

6.2 Informationsgehalt

Der Informationsgehalt eines Zeichens ist die Anzahl Elementarentscheidungen zur Bestimmung dieses Zeichens.

$$I(x_k) = \log_2\left(\frac{1}{p(x_k)}\right) \quad [\text{bit}] \quad (36)$$

6.2.1 Tatsächliche Codewortlänge L

$$L(x_k) = \lceil I(x_k) \rceil \quad [\text{Bit}] \quad (37)$$

Vorsicht: Wenn die Codewortlänge bekannt ist (z.B. nach Durchführung der Huffman-Codierung), muss die effektive Codewortlänge verwendet werden anstelle dieser Formel. Diese Formel gilt nur für die Quelle, nicht für die Codierung dieser.

6.3 Entropie

Unter der Entropie versteht man den mittleren Informationsgehalt einer Quelle. Also wie viele Elementarentscheidungen die Quelle/Senke im Mittel pro Zeichen treffen muss. Anderes Wort ist auch die «Ungewissheit» einer Quelle, d.h. wie unsicher ist das Erraten eines Codes.

$$H(X) = \sum_{k=1}^N p(x_k) I(x_k) = \sum_{k=1}^N p(x_k) \log_2\left(\frac{1}{p(x_k)}\right) \quad [\text{bit}/\text{Zeichen}] \quad (38)$$

Die Entropie ist maximal, wenn alle Zeichen dieselbe Wahrscheinlichkeit besitzen. Wenn jedoch nur ein Zeichen vorkommt, d.h. ein Zeichen hat die Wahrscheinlichkeit 1, dann ist die Entropie = 0.

6.3.1 Mittlere Codewortlänge

$$L = H_c(X) = \sum_{i=1}^N p(x_i) L(x_i) = \sum_{i=1}^N p(x_i) \lceil I(x_i) \rceil \quad [\text{bit}/\text{Zeichen}] \quad (39)$$

6.3.2 Quelle mit Gedächtnis

$$H(X, Y) = \sum_{i=1}^N \sum_{k=1}^N p(x_i) \cdot p(y_k|x_i) \cdot \log_2\left(\frac{1}{p(x_i) \cdot p(y_k|x_i)}\right) = H(X) + H(Y|X) \quad (40)$$

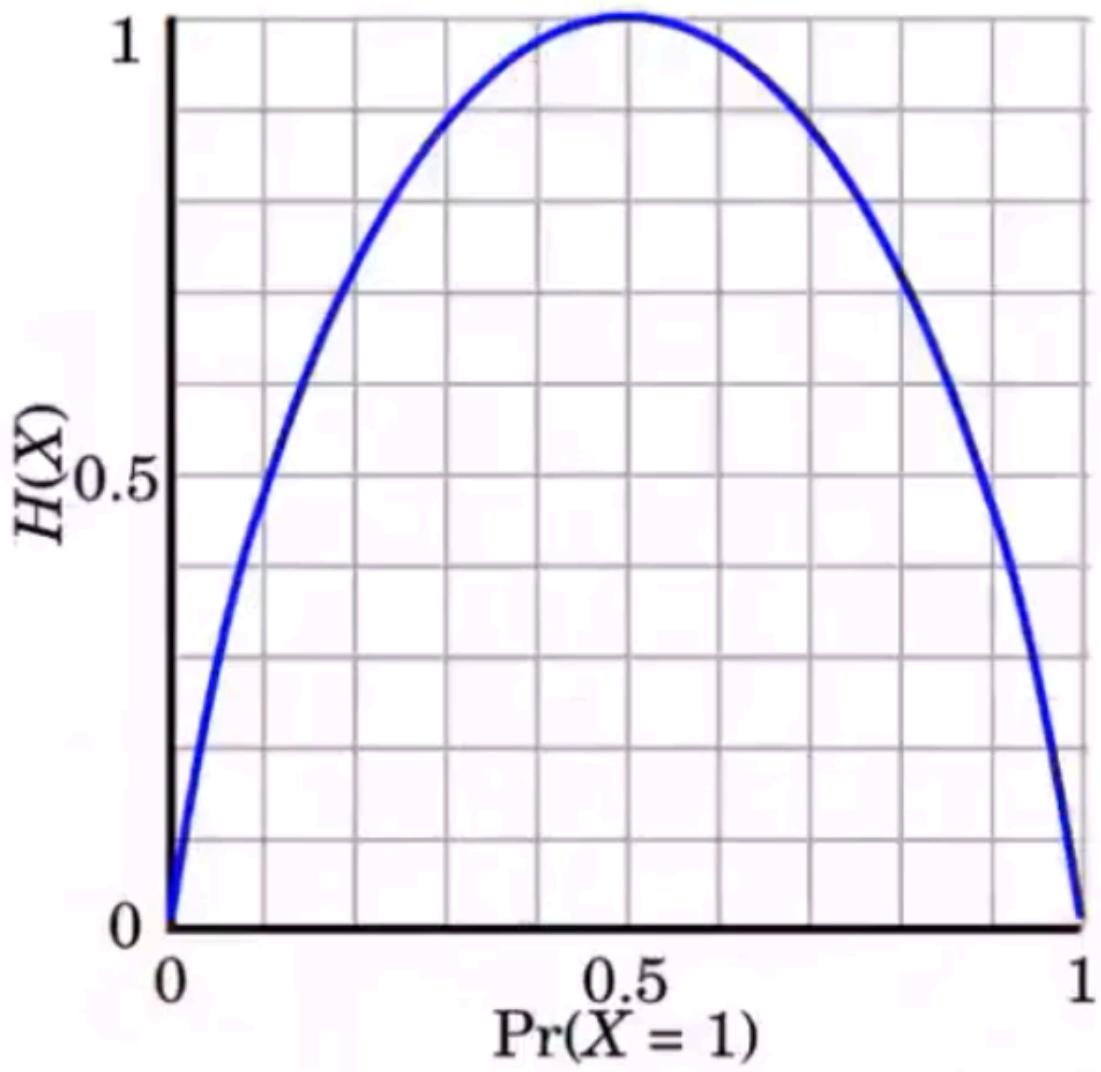
Redundanz:

$$R_Q = H_0 - H_{oG}(X) \leq H_0 - H_{mG}(x) = H_0 - H(Y|X) \quad (41)$$

6.3.3 Wann wird die Entropie einer Quelle/Senke maximal?

Die Entropie ist maximal, wenn alle Wahrscheinlichkeiten gleich sind.

$$\begin{aligned} X &= \{x_1, x_2\} \\ p(x_1) &= p \\ p(x_2) &= 1 - p \\ \Rightarrow H(X) &= p \cdot \log_2\left(\frac{1}{p}\right) + (1-p) \cdot \log_2\left(\frac{1}{1-p}\right) \end{aligned} \quad (42)$$



6.3.4 Shannon'sches Codierungstheorem

$$H(X) \leq L \leq H(X) + 1 \quad (43)$$

Wobei L = Mittlere Codewortlänge

6.4 Redundanz einer Quelle

Absolut:

$$R_{Q_{\text{absolut}}} = H_0 - H(X) \quad [\text{bit}/\text{Zeichen}] \quad (44)$$

Relativ:

$$R_{Q_{\text{relativ}}} = \frac{R_{Q_{\text{absolut}}}}{H_0} = 1 - \frac{H(X)}{H_0} \quad [\cdot 100\%] \quad (45)$$

Redundanz der Quelle:

$$R_Q = H_0 - H(X) \quad (46)$$

Redundanz des Codes:

$$R_c = L - H(X) \quad (47)$$

6.5 Präfixeigenschaft

Kodierung nach Binärtree, wobei Zeichen nur bei den Leafs sind, ist kommafrei (siehe Abbildung 4).

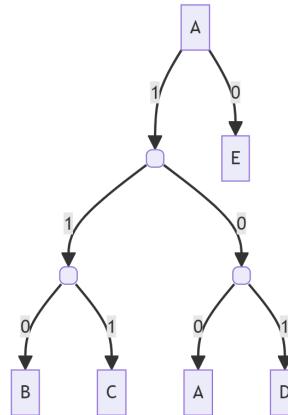


Abbildung 4: Kommafreie Codierung nach Binary Tree

6.6 Quelle mit Gedächtnis

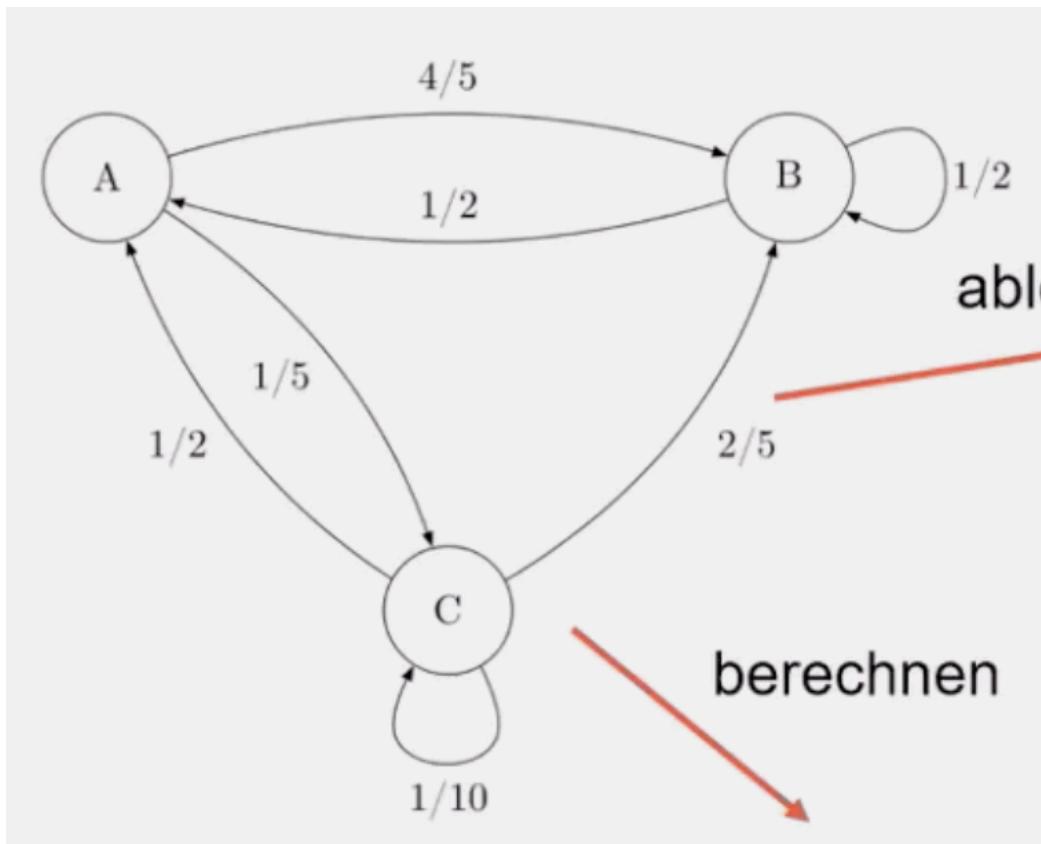


Abbildung 5: Quelle mit Gedächtnis

Folgende Tabelle kann man ablesen:

	A	B	C
A	0	4/5	1/5
B	1/2	1/2	0

	A	B	C
C	1/2	2/5	1/10

Nun hat man folgende drei Gleichungen:

$$\begin{aligned}
 p(A) &= \frac{1}{2} \cdot p(B) + \frac{1}{2} \cdot p(C) \\
 p(B) &= \frac{4}{5} \cdot p(A) + \frac{1}{2} \cdot p(B) + \frac{2}{5} \cdot p(C) \\
 p(C) &= \frac{1}{5} \cdot p(A) + \frac{1}{10} \cdot p(C)
 \end{aligned} \tag{48}$$

Diese sind jedoch linear abhängig. Somit nimmt man zwei von den obigen drei Gleichungen und die Gleichung

$$p(A) + p(B) + p(C) = 1 \tag{49}$$

um ein lösbares lineares Gleichungssystem zu erhalten.

7 Kompressionsmethoden

7.1 Lauflängenkomprimierung

Ähnliches Prinzip wie «AAAbbccccc» = «3A2bc3e», aber mit Bits. Hierzu muss jeweils nur klar sein welcher Bit als Erstes kommt, 1 oder 0. Beispiel:

$$\begin{aligned}
 &1111'1100'0001'1001 \\
 &= [6 \cdot 1, 5 \cdot 0, 2 \cdot 1, 2 \cdot 0, 1] \\
 &= 6, 5, 2, 2, 1 \\
 &= 110'101'010'010'001
 \end{aligned} \tag{50}$$

Start mit einer 1 → 1'110'101'010'010'001

7.2 Lempel-Ziv-Verfahren

Eine Datenstruktur besteht aus:

- Einem Textfenster, dem **search buffer**
 - Schon kodierte Symbole
 - Wird dynamisch aufgebaut
- Nach vorne gerichteten Puffer, dem **look-ahead buffer**

Beispiel:

Search Buffer/Textfenster	Look-ahead buffer	Coding (Position, Länge, Zeichen)
	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, «_»)
sir_	sid_eastmam	(4, 2, «d») 4 Stellen zurück, 2 Zeichen lang
sir_sid	_eastman	

7.2.1 Lempel-Ziv-Welch (LZW)

Startet im Normalfall mit dem Wörterbuch:

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

Wir codieren die Zeichenfolge «36363»:

Buffer	Erkannte Zeichenfolge (Index)	Neuer Eintrag
36363	3 (3)	10: 3
36363	6 (6)	11: 63
36363	36 (10)	12: 363
36363	3 (3)	-

Codiert wird «3 6 10 3»

7.2.2 Lempel-Ziv für Binärzahlen



"0" "01" "1" "010" "10" "101" "11" "0100" "10"
 $(0,0) (1,1) (0,1) (2,0) (3,0) (5,1) (3,1) (4,0) (5,\text{eof})$

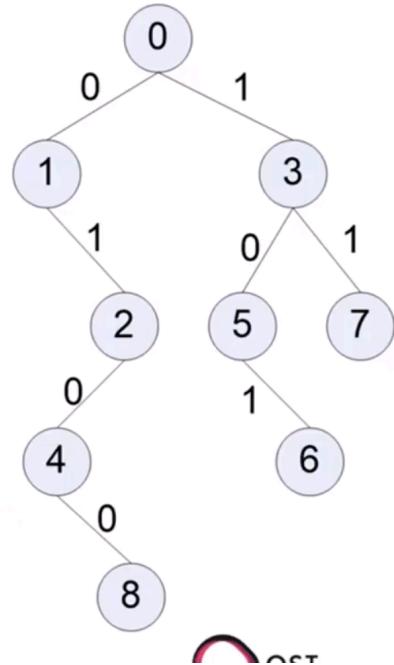


Abbildung 6: Lempel-Ziv-Algorithmus für Binärzahlen

7.3 Kombination Lempel-Ziv und Huffman

Es ist möglich, zunächst die Nachricht mit Lempel-Ziv zu komprimieren und dann mit der Huffman-Codierung das entstandene Wörterbuch weiter zu komprimieren.

8 Kryptologie

Assymetrische Verschlüsselung ist nicht performant. Deshalb verwendet man oft nur zu Beginn einer Kommunikation asymmetrische Verschlüsselung, damit ein sicherer Kanal entsteht. Ein gemeinsamer Schlüssel für symmetrische Verschlüsselung wird dann über die asymmetrische Verschlüsselung sicher übertragen.

8.1 RSA

Durch $\text{mod } b$ wird ein Ereignisraum von 0 bis $b - 1$ möglich. Somit kann man eine Zahl c ausserhalb dieses Ereignisraums finden, die in dieser Rechenvorschrift eine Inverse Zahl zu a ist.

$$\begin{array}{ll} m^e \text{ mod } N & \text{Verschlüsseln} \\ (m^e)^d \text{ mod } N & \text{Entschlüsseln} \\ = m^{ed} \text{ mod } N & \end{array} \quad (51)$$

Dabei muss folgendes gelten:

$$e \cdot d \equiv 1 \text{ mod } N \quad (52)$$

Für zwei teilerfremde Zahlen a, b existiert eine Zahl c , so das gilt:

$$a \cdot c \equiv 1 \text{ mod } b \quad (53)$$

8.1.1 Eulerfunktion

$\varphi(n) = \text{Anzahl der relativ primen (teilerfremden) Zahlen zu } n$

Theorem 8.1.1.1:

$$a^y \text{ mod}(p \cdot q) = a^{y \text{ mod } \varphi(p \cdot q)} \text{ mod}(p \cdot q) \quad (54)$$

Beweis:

$$\begin{aligned} a^y \text{ mod}(p \cdot q) &\stackrel{!}{=} a^{y \text{ mod } \varphi(p \cdot q)} \text{ mod}(p \cdot q) \\ &= a^{y-n \cdot \varphi(p \cdot q)} \text{ mod}(p \cdot q) \\ &= a^y \cdot a^{-n \cdot \varphi(p \cdot q)} \text{ mod}(p \cdot q) \\ &= (a^y \text{ mod}(p \cdot q)) \cdot \underbrace{(a^{\varphi(p \cdot q)} \text{ mod}(p \cdot q))^{-n}}_{=1} \\ &= a^y \text{ mod}(p \cdot q) \end{aligned} \quad (55)$$

■

Beispiel (RSA): Seien Primzahlen $p = 47, q = 59$ gegeben. Daraus folgt: $p \cdot q = 2773$ und $\varphi(n) = (p - 1) \cdot (q - 1) = 2668$.

Wir wählen den Wert $e = 17$, relativ prim zu $\varphi(n)$, wobei $1 < e < \varphi(n)$. Und dann berechnen wir den zu e inversen Wert $d = 157$.

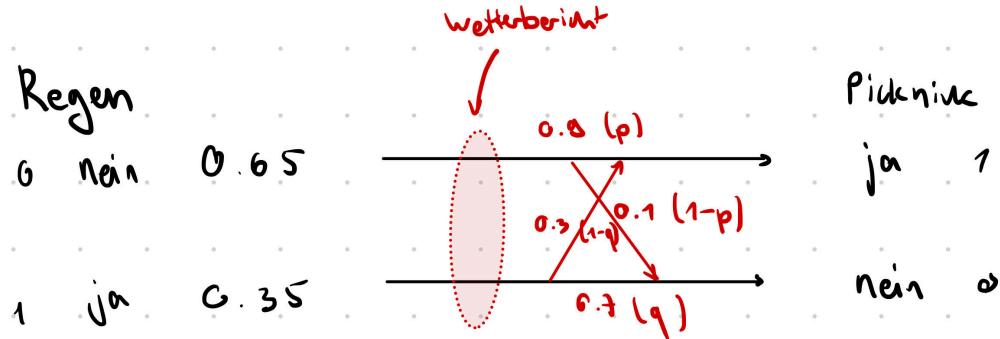
Privater Schlüssel: (d, n) , Öffentlicher Schlüssel: (e, n)

$$\begin{array}{ll} m = 12 \rightarrow c = m^e \bmod n = 336 & \text{Encrypt} \\ c = 336 \rightarrow m = c^d \bmod n = 12 & \text{Decrypt} \end{array} \quad (56)$$

8.1.2 Euklidischer Algorithmus

x	y	q	r	u	s	v	t
		x div y	x mod y	$u_{k+1} = s_k$	$s_{k+1} = u_k - q_k \cdot s_k$	$v_{k+1} = t_k$	$t_{k+1} = v_k - q_k \cdot t_k$

9 Kanalmodell



$$\begin{aligned} p(y_1) &= p(x_1) \cdot p + p(x_2) \cdot (1 - q) \\ p(y_2) &= p(x_1) \cdot (1 - p) + p(x_2) \cdot q \end{aligned} \quad (57)$$

$$p(Y|X) = \begin{pmatrix} p & 1-p \\ 1-q & q \end{pmatrix} \quad (58)$$

In diesem Beispiel wäre

$$p(Y|X) = \begin{pmatrix} 0.9 & 0.1 \\ 0.3 & 0.7 \end{pmatrix} \quad (59)$$

Fehlerfrei:

$$p(Y|X) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad (60)$$

$$p(P = 1, R = 1) = p(R = 1) \cdot p(P = 1|R = 1) = 0.35 \cdot 0.3 = 0.105 \quad (61)$$

9.1.1 3x3-Matrix

Eine 3x3-Matrix ist also folgendermassen aufgebaut:

$$\begin{pmatrix} p(y_1|x_1) & p(y_1|x_2) & p(y_1|x_3) \\ p(y_2|x_1) & p(y_2|x_2) & p(y_2|x_3) \\ p(y_3|x_1) & p(y_3|x_2) & p(y_3|x_3) \end{pmatrix} \quad (62)$$

$$\begin{pmatrix} p(y_1) \\ p(y_2) \\ p(y_3) \end{pmatrix} = \begin{pmatrix} p(x_1) \cdot p(y_1|x_1) + p(x_2) \cdot p(y_1|x_2) + p(x_3) \cdot p(y_1|x_3) \\ p(x_1) \cdot p(y_2|x_1) + p(x_2) \cdot p(y_2|x_2) + p(x_3) \cdot p(y_2|x_3) \\ p(x_1) \cdot p(y_3|x_1) + p(x_2) \cdot p(y_3|x_2) + p(x_3) \cdot p(y_3|x_3) \end{pmatrix} \quad (63)$$

9.1.2 Symmetrischer Kanal

Fehlerwahrscheinlichkeit des Kanals ist unabhängig von der Auftrittswahrscheinlichkeit der Zeichen der Quelle.

$$\begin{pmatrix} 0.95 & 0.025 & 0.025 \\ 0.025 & 0.95 & 0.025 \\ 0.025 & 0.025 & 0.95 \end{pmatrix} \quad (64)$$

9.1.3 Verbundentropie

$$H(X, Y) = - \sum_i^n \sum_j^n p(x_i, y_j) \cdot \log_2(p(x_i, y_j)) \quad (65)$$

9.1.4 Äquivokation (Verlust)

$$\begin{aligned} H(X|Y) &= - \sum_i^n \sum_j^n p(x_i, y_j) \cdot \log_2(p(x_i|y_j)) \\ &= - \sum_i^n \sum_j^n p(y_j) \cdot p(x_i|y_j) \cdot \log_2(p(x_i|y_j)) \end{aligned} \quad (66)$$

- Ähnliche Formel wie die Verbundentropie (Rückschlusstropie)
- Ungewissheit über das gesendete Zeichen bei bekanntem Empfangszeichen
- Merke: Ist der Kanal fehlerfrei, so ist die Äquivokation gleich 0

9.1.5 Irrelevanz (Rauschen)

$$\begin{aligned} H(Y|X) &= - \sum_i^n \sum_j^n p(x_i, y_j) \cdot \log_2(p(y_j|x_i)) \\ &= - \sum_i^n \sum_j^n p(x_i) \cdot p(y_j|x_i) \cdot \log_2(p(y_j|x_i)) \end{aligned} \quad (67)$$

- Auch Streuentropie genannt
- Ungewissheit der empfangenen Zeichen bei vorgegebenen Sendezeichen

9.1.6 Transinformation

$$\begin{aligned} T &= H(X) - H(X|Y) \\ T &= H(Y) - H(Y|X) \end{aligned} \quad (68)$$

Fazit:

- Ein nicht gestörter Kanal (Einheitsmatrix) überträgt den mittleren Informationsfluss ohne weiteren Verlust, d.h. die Transinformation wird nur durch die Quelle beeinflusst
- Verändert sich die Entropie der Quelle, so verändert sich auch die Transinformation
- Nimmt die Fehlerwahrscheinlichkeit zu, nimmt die Transinformation ab
- Sind alle Positionen der Kanalmatrix gleich besetzt, so wird die Transinformation $T = 0$ und $H(Y) = H(Y|X) = 1$ und zwar unabhängig von der Entropie am Kanaleingang
- Die Transinformation gibt den maximalen und somit fehlerfreien Informationsfluss über einen gestörten Kanal an.

9.1.7 Maximum-Likelihood-Verfahren

Gegeben sei eine Kanalmatrix $P(Y|X)$. Man wählt in jeder Spalte die höchstwahrscheinliche Zahl:

$$P(Y|X) = \begin{pmatrix} 0.5 & 0.4 & 0.1 \\ 0.2 & 0.1 & 0.7 \\ 0.3 & 0.5 & 0.2 \end{pmatrix} \quad (69)$$

Der Entscheider ist hier somit:

$$\begin{aligned} y_1 &\rightarrow x_1 \\ y_2 &\rightarrow x_3 \\ y_3 &\rightarrow x_2 \end{aligned} \quad (70)$$

Die Spalten gelten für die Y 's, die Zeilen für die X 's in der Kanalmatrix.

9.2 Fehlerkorrektur

Idee: Wir fügen Redundanz hinzu, damit sich der Coderaum in gültige und ungültige Codewörter aufteilt (Siehe Abbildung 7).

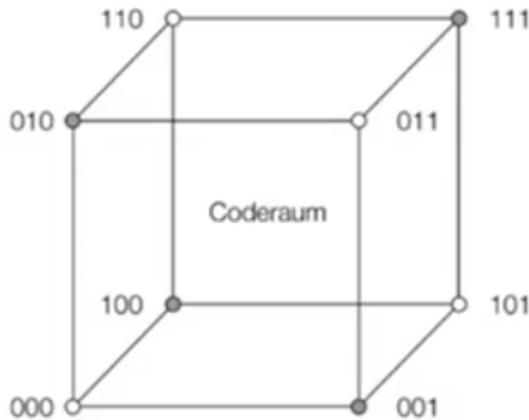


Abbildung 7: Coderaum. Weisse Punkte sind gültige Codewörter, dunkle Punkte sind ungültig. Hier ist nur Fehlererkennung möglich, keine Fehlerkorrektur.

Die Distanz zwischen zwei gültigen Codewörtern nennt man Distanz d .

Hammingdistanz h : Die kleinste Distanz zwischen zwei gültigen Codewörtern (\rightarrow Flachdrücken des Coderaums).

$$h = \min_{i,j}(d(x_i, x_j)) \quad (71)$$

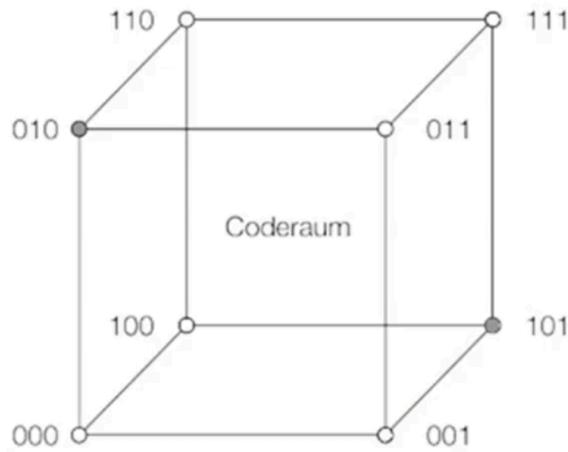


Abbildung 8: Coderaum mit Fehlerkorrektur möglich, Hammingdistanz 3.

Anzahl der sicher erkennbaren Fehler $e^* = h - 1$. Im Beispiel Abbildung 8 ist die Hammingdistanz $h = 3$. Die Anzahl erkennbaren Fehler ist somit $e^* = 3 - 1 = 2$.

Anzahl sicher korrigierbaren Fehler $e = \left\lfloor \frac{h-1}{2} \right\rfloor$

Bei Hamming-Blockcode ist $h = 3$.

9.2.1 Korrigierkugel

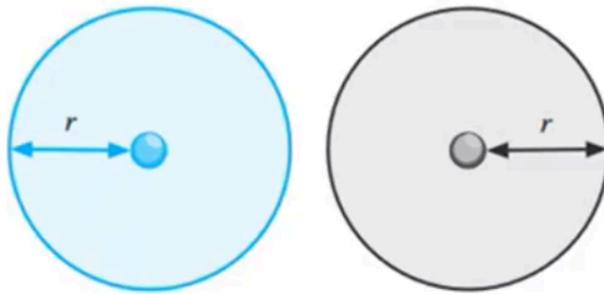


Abbildung 9: Korrigierbare Korrigierkugeln dürfen sich nicht berühren.

Eine Korrigierkugel hat das Zentrum bei einem gültigen Punkt und umfasst alle korrigierbaren Fehler in ihrem Radius. Der Radius ist somit die Anzahl sicher korrigierbarer Fehler e . Bei einer Skizze kann man dann noch die Hammingdistanz h und die erkennbaren Fehler e^* einzeichnen.

Beispiel (Quersummencode):

$m = 2$		$k = 1$
x_1	x_2	x_3
0	0	0
0	1	1
1	0	1
1	1	0
0	0	1
0	1	0
1	0	0

1	1	1
---	---	---

Ein solcher Code hat die Hammingdistanz $d = 2$, da der Abstand zwischen zwei korrekten Kombinationen **zwei Bits Unterschied** hat.

9.2.2 Paritätsbit-Code

0100	1111	1
0101	0011	0
0101	0100	1
0100	1000	0

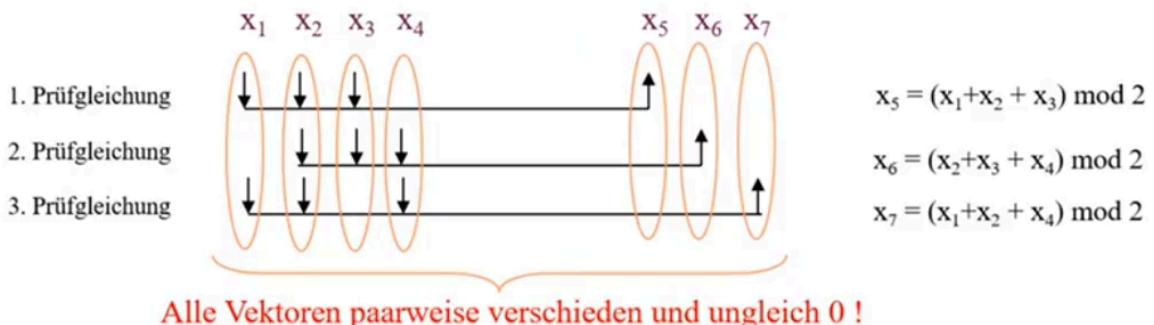
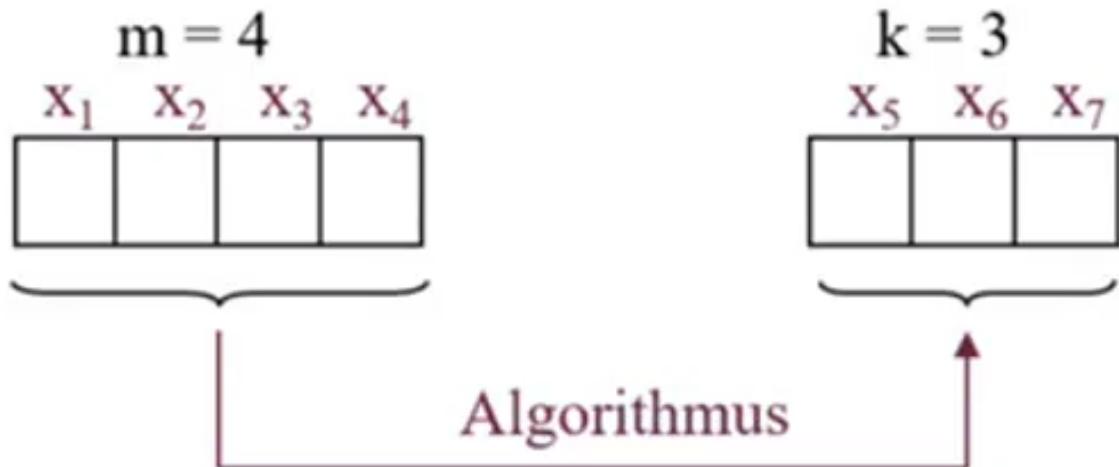
Es können mehrere Fehler gefunden und korrigiert werden.

9.2.3 Hammingcode

k -Kontrollstellen:

$$\begin{aligned} n &= 2^k - 1 \\ n &= m + k \end{aligned} \tag{72}$$

Die Notation des Hammingcode ist immer ein Tupel (n, m) , z.B. $n = 7, m = 4 \rightarrow (7, 4)$



Es wird pro Kontrollbit ein Vektor mit jeweils zwei Stellen gebildet. Hammingdistanz ist drei, somit können zwei Fehler erkannt und ein Fehler korrigiert werden. Daraus bildet sich nun folgende **Generatormatrix**:

$$\left(\begin{array}{cccc|ccc} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{array} \right) \quad (73)$$

$$\overrightarrow{P_1} \overrightarrow{P_2} \overrightarrow{P_3} \overrightarrow{P_4} \overrightarrow{P_5} \overrightarrow{P_6} \overrightarrow{P_7}$$

Hieraus folgt die Codebedingung: $\sum_i x_i \cdot \vec{P}_i \equiv \vec{0} \pmod{2}$

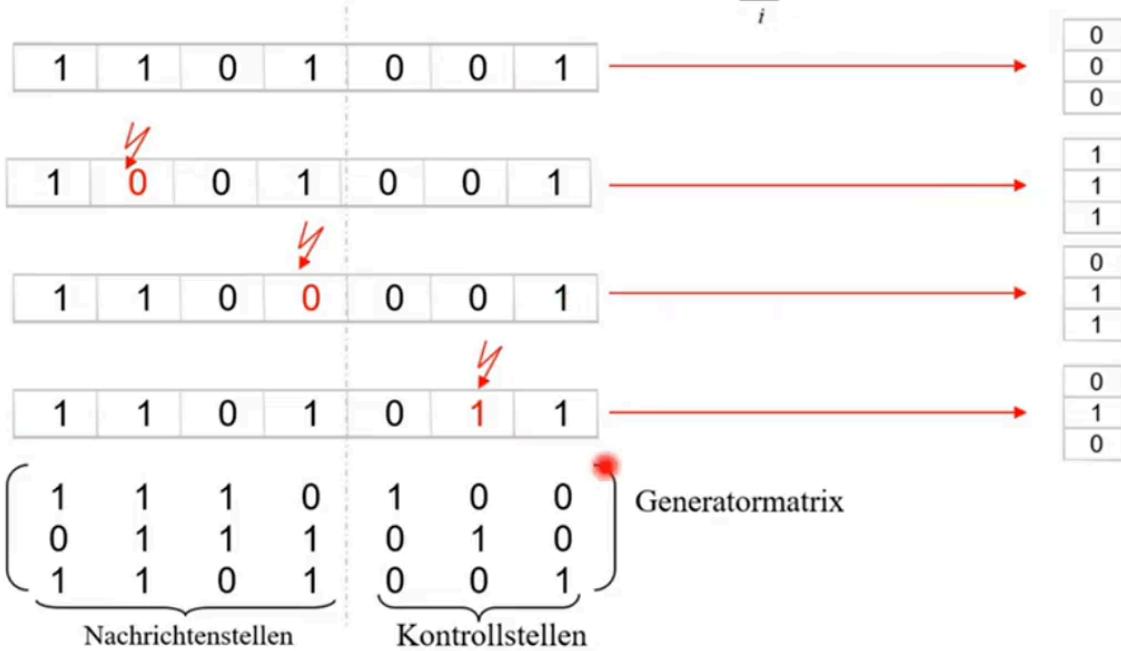
Beispiel: Dieselbe Matrix, erhaltenes Codewort: 1010011.

$$1 \cdot \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} + 1 \cdot \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} + 1 \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + 1 \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \equiv \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \pmod{2} \quad (74)$$

✓ ✓

Das Syndrom Z:

$$\vec{Z} = \sum_i x_i \cdot \vec{P}_i \pmod{2}$$



→ Das entstandene Fehlersyndrom ist derselbe Vektor wie der Spaltenvektor an der Stelle des Fehlers!

Fehlersyndrom:

Aus der Codebedingung folgt das Syndrom

$$\vec{Z} = \sum_i x'_i \cdot \vec{P}_i = \sum_i (x_i + f_i) \cdot \vec{P}_i$$

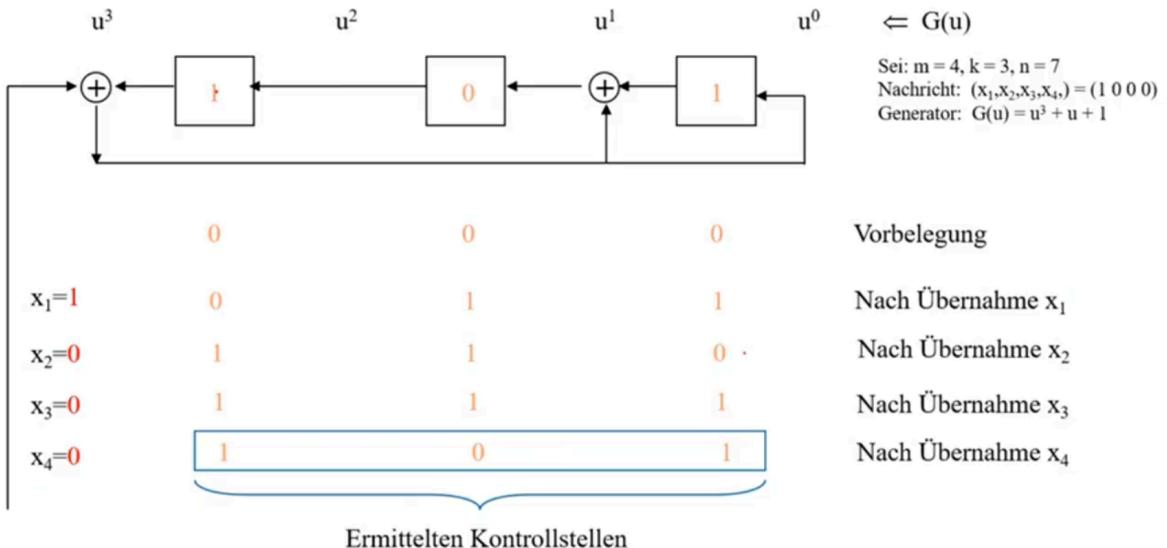
$$= \sum_i x_i \cdot \vec{P}_i + \sum_i f_i \cdot \vec{P}_i$$

Codebedingung = 0

$$\Rightarrow \vec{Z} = \sum_i f_i \cdot \vec{P}_i$$

Das heisst, bei genau einem Fehler markiert die Prüfspalte den Fehlerort.

9.2.4 Ermitteln der Kontrollstellen durch rückgekoppeltes Schieberegister



9.2.5 Abramson- bzw. CRC-Code

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

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

CRC-Codes haben immer die Hammingdistanz $h = 4$ (Multiplikation \rightarrow eins mehr als zyklische Hammingcodes). Sie haben immer die Länge $n = 15$ (Andernfalls ist die Formel bei CRC $n = 2^{k-1} - 1$).

Beispiel:

$$\begin{aligned} g(x) &= (1 + x + x^3) \cdot (1 + x) \\ g(x) &= 1 + x^2 + x^3 + x^4 \end{aligned} \tag{75}$$

Wenn die Frage nach einem CRC-Code ist, reicht es, dieses Polynom anzugeben.

Anzahl Kontrollstellen ist dieselbe, wie der Grad des Polynoms, in diesem Beispiel 4.

9.3 Codierungsdichte

Sei:

- n die Dimension des Codes (Anzahl aller CW 2^n)
- m die Dimension der Nachrichten (Anzahl aller gültigen CW 2^m)
- k die Dimension der Kontrollstellen mit $n = m + k$

Codeabschätzung (Dann ist der Code **nicht** dichtgepackt):

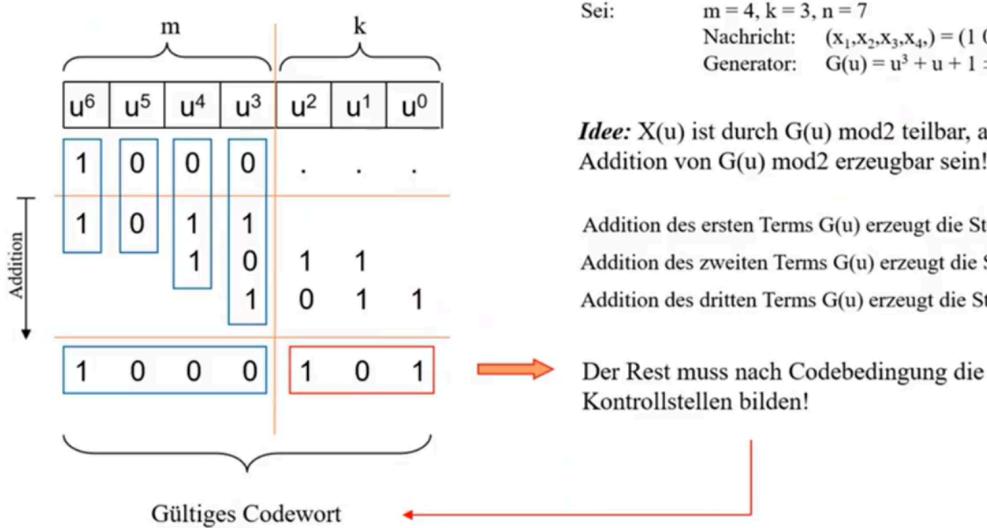
$$2^m \cdot \sum_{w=0}^e \binom{n}{w} \leq 2^n \tag{76}$$

9.4 Fehlersyndrom

$$\vec{Z} = \sum_i f_i \cdot \vec{P}_i \tag{77}$$

9.5 Zyklischer Hamming-Code

Ermittlung der Kontrollstellen durch Mehrfachaddition (alternativ zu Polynomdivision):



Sei:
 $m = 4, k = 3, n = 7$
Nachricht: $(x_1, x_2, x_3, x_4) = (1, 0, 0, 0)$
Generator: $G(u) = u^3 + u + 1 \Rightarrow (g_3, g_2, g_1, g_0) = (1, 0, 1, 1)$

Idee: $X(u)$ ist durch $G(u) \bmod 2$ teilbar, also muss $X(u)$ durch Addition von $G(u) \bmod 2$ erzeugbar sein!

Addition des ersten Terms $G(u)$ erzeugt die Stellen u^6, u^5 von $X(u)$

Addition des zweiten Terms $G(u)$ erzeugt die Stelle u^4 von $X(u)$

Addition des dritten Terms $G(u)$ erzeugt die Stelle u^3 von $X(u)$

Der Rest muss nach Codebedingung die Kontrollstellen bilden!

- $X(u)$: Codewort
- $G(u)$: Generatorpolynom

Beispiel: $X(u) = 1000101, G(u) = 1011$. Durch Mehrfachaddition kann berechnet werden, ob $X(u)$ ein gültiges Codewort ist (siehe Abbildung 10).

$$\begin{array}{r}
1 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \\
1 \ 0 \ 1 \ 1 \\
1 \ 0 \ 1 \ 1 \\
1 \ 0 \ 1 \ 1 \\
\hline
0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0
\end{array}$$

Abbildung 10: Polynomdivision ohne Rest

Wenn man ein gültiges Codewort hat, kann man jeweils einen Fehler einbauen für jede Position und dann wieder durch Polynomdivision die Generatormatrix herleiten (siehe Abbildung 11). Der übrigbleibende Rest entspricht immer dem **Fehlersyndrom**!

9.5.1 Ermitteln der Kontrollstellen durch Mehrfachaddition

Wenn wir z.B. wissen, dass wir ein 4-stelliges Codewort ($m = 4$) kodieren und $k = 3$, und das Generatorpolynom 1010 beträgt, können wir verschiedene Additionen vornehmen, um die korrekten Codeworte zu erhalten:

$$\begin{array}{r|l}
0001 & \\
1 & 010 \\
\hline
0 & 010
\end{array}$$

→ 0001010 ist ein gültiges Codewort.

$$\begin{array}{r|l}
1000 & \\
1010 & \\
\hline
1010 &
\end{array}$$

0010	
10	10
00	100

→ 1000100 ist ein gültiges Codewort

Gültiges Codewort: **1 0 0 0 1 0 1**

$\begin{array}{r} \textcolor{red}{0} \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \\ \hline 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \end{array}$	$\begin{array}{r} 1 \ \textcolor{red}{1} \ 0 \ 0 \ 1 \ 0 \ 1 \\ 1 \ 0 \ 1 \ 1 \\ 1 \ 0 \ 1 \ 1 \\ \hline 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \end{array}$	$\begin{array}{r} 1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \\ 1 \ 0 \ 1 \ 1 \ 1 \\ 1 \ 0 \ 1 \ 1 \\ \hline 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 0 \end{array}$	$\begin{array}{r} 1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 1 \\ 1 \ 0 \ 1 \ 1 \\ 1 \ 0 \ 1 \ 1 \\ \hline 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \end{array}$
$\begin{array}{r} 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \\ 1 \ 0 \ 1 \ 1 \\ 1 \ 0 \ 1 \ 1 \\ 1 \ 0 \ 1 \ 1 \\ \hline 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \end{array}$	$\begin{array}{r} 1 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \\ 1 \ 0 \ 1 \ 1 \\ 1 \ 0 \ 1 \ 1 \\ 1 \ 0 \ 1 \ 1 \\ \hline 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \end{array}$	$\begin{array}{r} 1 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \\ 1 \ 0 \ 1 \ 1 \\ 1 \ 0 \ 1 \ 1 \\ 1 \ 0 \ 1 \ 1 \\ \hline 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \end{array}$	

Generatormatrix $\xrightarrow{\hspace{1cm}}$
$$\begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Abbildung 11: Herleitung der Generatormatrix

9.6 Faltungscodes

Ein Faltungscode wird als Tupel (a, e, s) angegeben wobei:

- a: Anzahl Ausgänge
- e: Anzahl Eingänge
- s: Anzahl Speicherplätze

Beispiel für einen Faltungscode $(3, 1, 2)$ mit:

- $g_1(x) = 1 + x$
- $g_2(x) = 1 + x^2$
- $g_3(x) = 1 + x + x^2$

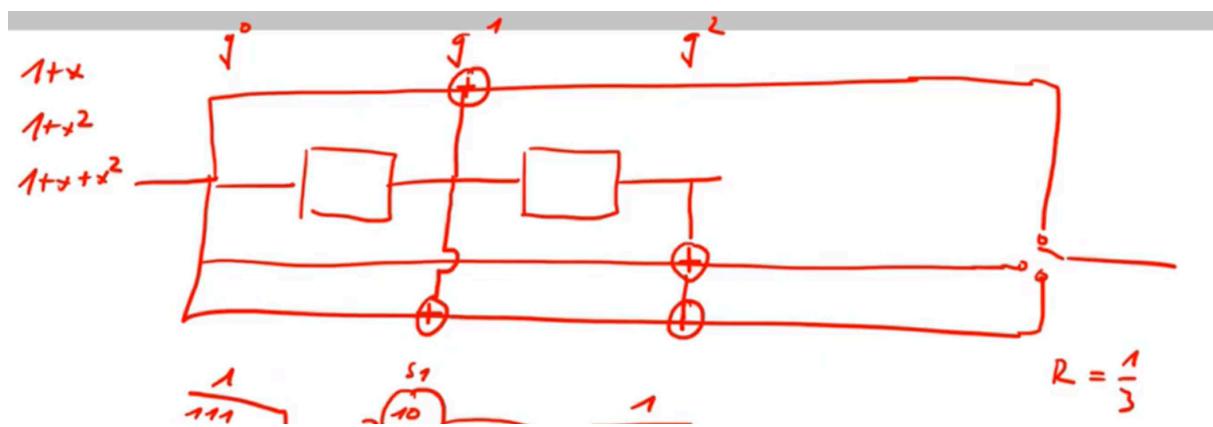
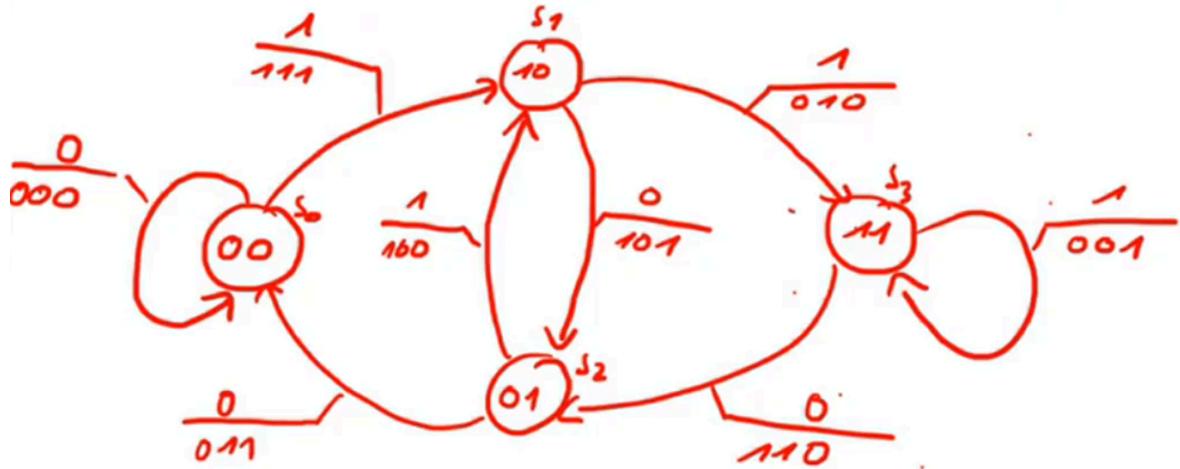
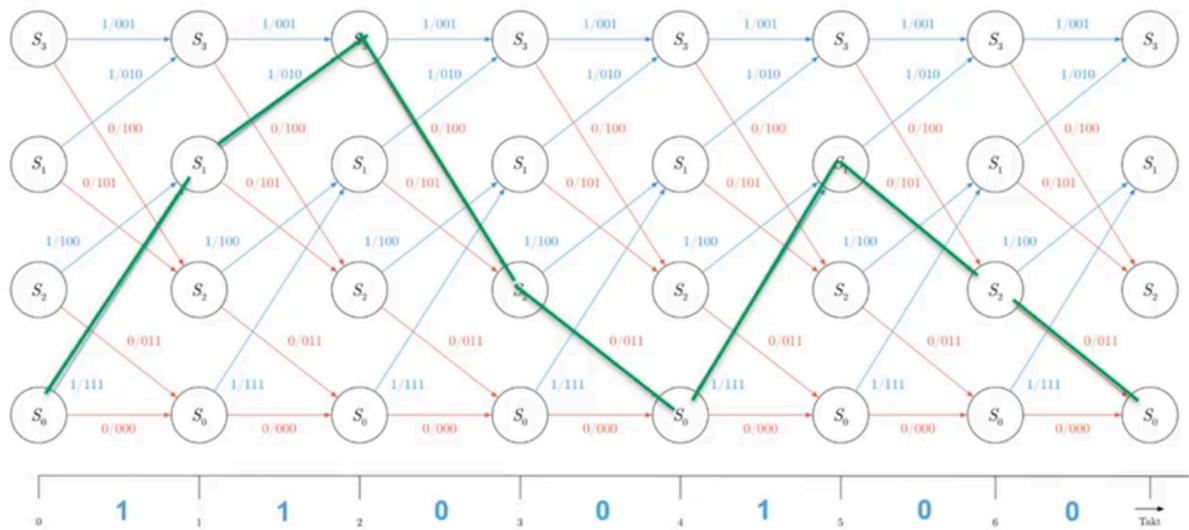


Abbildung 12: Faltungscode

9.6.1 Zustandsdiagramm

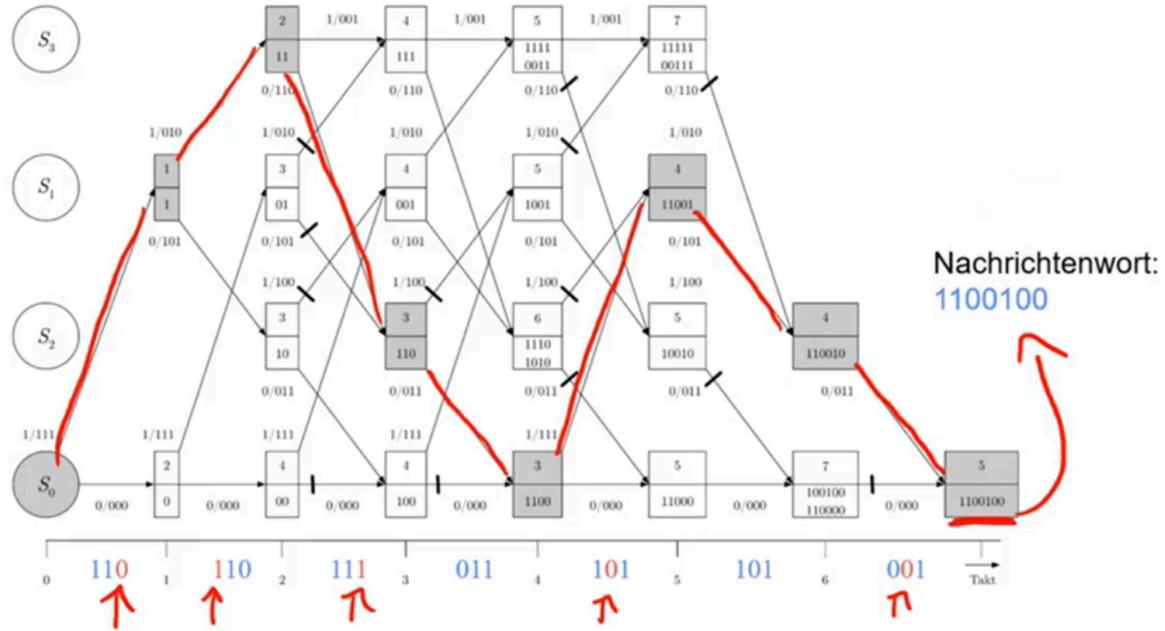
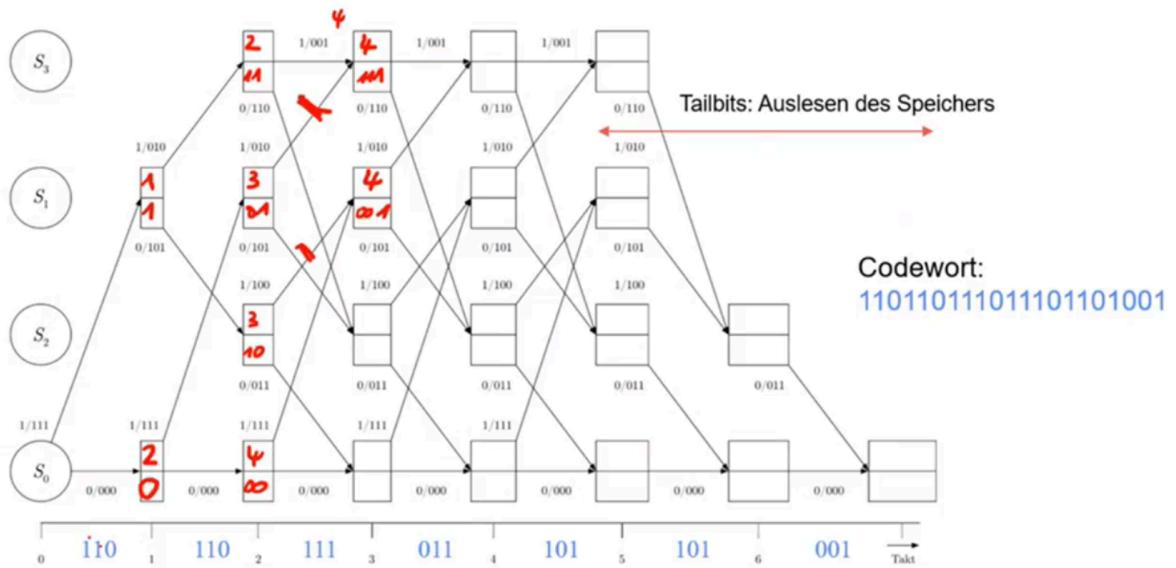


Netzdiagramm:



Ergibt die Kodierung: 111 010 100 011 111 101 011 → Bitrate: $\frac{1}{3}$ (1 bit wird in 3 bit übersetzt)

Decodierung des (3, 1, 2) Faltungscodes:



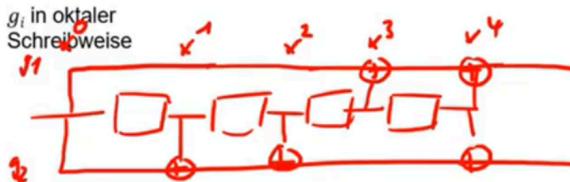
(2,1,4)-Code

$$g_1 = 23_{10} = 010\ 011 \rightarrow x^4 + x^3 + 1$$

$$g_2 = 35_{10} = 011\ 101 \rightarrow x^4 + x^2 + x + 1$$

m	R = $\frac{1}{2}$			R = $\frac{1}{3}$				R = $\frac{1}{4}$				
	g_1	g_2	d_f	g_1	g_2	g_3	d_f	g_1	g_2	g_3	g_4	d_f
2	5	7	5	5	7	7	8	5	7	7	7	10
3	15	17	6	13	15	17	10	13	15	15	17	15
4	23	35	7	25	33	37	12	25	27	33	37	16
5	53	75	8	47	53	75	13	53	67	71	75	18
6	133	171	10	133	145	175	15	135	135	147	163	20
7	247	371	10	225	331	367	16	235	275	313	357	22
8	561	753	12	557	663	711	18	463	535	733	745	24

g_i in oktaler Schreibweise



[Martin Werner, Information und Codierung, vieweg]

