Version 1.1

June 11, 2025

Zusammenfassung

Digitale Codierung

Katharina Tschirky

katharina@tschirky.ch

Inhaltsverzeichnis

[Inhaltsverzeichnis 1](#_Toc200652691)

[1 Exzess, Fest- und Gleitkommazahlen - W3 2](#_Toc200652692)

[1.1 Exzess (Überschusscode) 2](#_Toc200652693)

[1.2 Gleitkommazahlen 2](#_Toc200652694)

[2 Boolsche Logik – W5 2](#_Toc200652695)

[3 Wahrscheinlichkeit - W6 3](#_Toc200652696)

[4 Informations- und Codierungstheorie – W7 3](#_Toc200652697)

[5 Quellencodierung und Verschlüsselung – W9 4](#_Toc200652698)

[6 Quellencodierung und Komprimierung – W8 4](#_Toc200652699)

[7 Kanalmodell – W10 5](#_Toc200652700)

[8 Blockcodes – W11 5](#_Toc200652701)

[9 Faltungscode – W12 6](#_Toc200652702)

[10 Glossar 6](#_Toc200652703)

[10.1 Gruppen, Ring und Körper / Text, Festkomma- und Gleitkommazahlen (U3 & U4) 6](#_Toc200652704)

[10.2 Informationstheorie / Quellencodierung und Komprimierung (U7/U8) 7](#_Toc200652705)

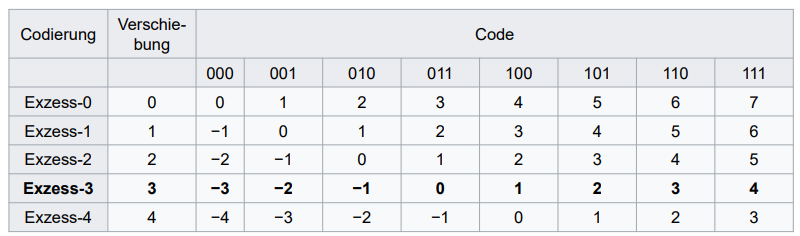
[10.3 Blockcodes und Faltungscodes (U11/12) 8](#_Toc200652706)

[10.4 Kanalmatrix 8](#_Toc200652707)

# Exzess, Fest- und Gleitkommazahlen - W3

## Exzess (Überschusscode)

* Exzess-127 erlaubt Darstellung von -127 bis 128.
* Exzesscodierung verschiebt den Nullpunkt in den positiven Bereich. 🡪 Code basiert auf einer Wertebereichverschiebung
* Nutzt **nicht** das Zweierkomplement intern
* Exzesscodierung funktioniert nicht nur mit 4 Bit
* Wenn negative Zahlen dargestellt werden wollen, muss man den Nullpunkt verschieben
* In einer Exzesscodierung wird der Bias zur darzustellenden Zahl addiert, um den Binärcode zu erhalten



**Cex−4,5(x):**

Bias: -4 🡪 steht für den niedrigsten negativen Wert

Länge der binären Schreibweise: 5

Zu Codierende Zahl: x

## Gleitkommazahlen

* Der Exponent wird im Exzessformat gespeichert.
* Das Vorzeichenbit 1 bedeutet negative Zahl.
* Bei Gleitkommazahlen wird zusätzlich zum Bitmuster z der eigentlichen Zahl auch noch die Stelle k mitgeführt, an der das Komma steht
* Gleitkommazahlen sind Näherungen
* Mantisse wird **nicht** im Zweierkomplement dargestellt

**CGK k, n (z) =z \* 2k :**

K = Exponent (Positiv oder negativ, verwendet Exzess)

Z = Signifikand (Mantisse = Nachkommateil) 🡪 Eigentlicher Wert der Zahl (rechts vom Binärpunkt

# Boolsche Logik – W5

Konjunktion: AND

Disjunktion: OR

Negation: NOT

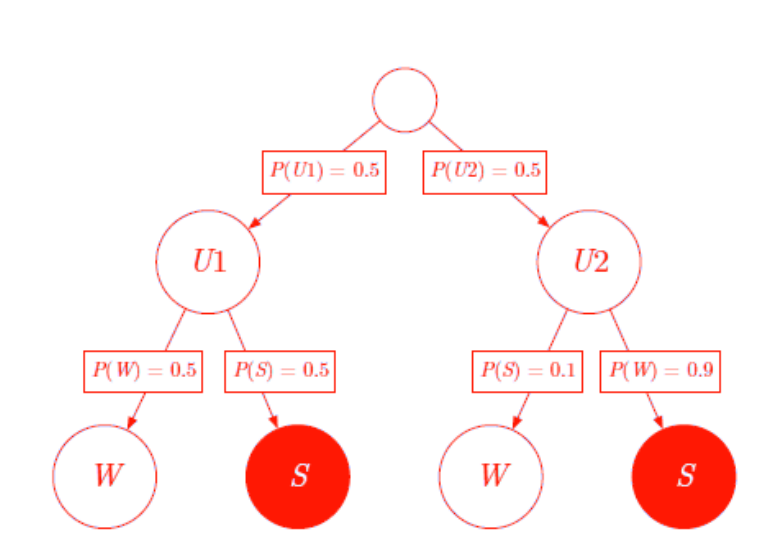
Entweder-Oder: XOR

# Wahrscheinlichkeit - W6

**Auftrittswahrscheinlichkeit:**

1. Anzahl aller Ereignisse berechnen
2. P(A1) = Anzahl / Anzahl Ereignisse

**Ereignisbaum:**

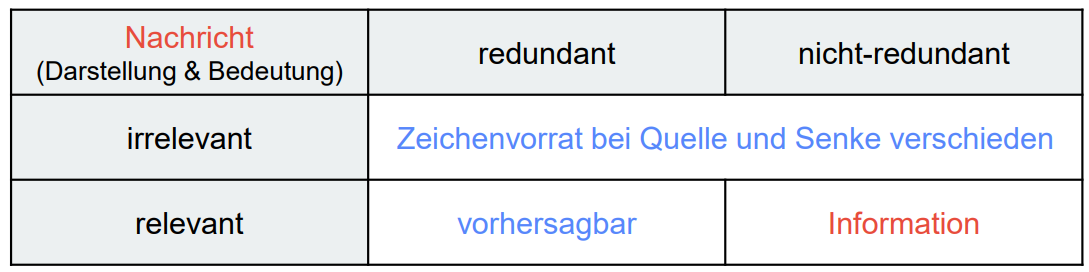
****

**Wahrscheinlichkeiten zusammenrechnen:**

Bei oder: P(A1) + P(A2)

Bei und dann: P(A1) x P(A2)

# Informations- und Codierungstheorie – W7



**Entscheidungsgehalt:**

* Die Auftrittswahrscheinlichkeiten der Zeichen einer Quelle haben keinen Einfluss auf den Entscheidungsgehalt der Quelle.
* Mass für den Aufwand, der zur Bildung einer Nachricht bzw. für die Entscheidung einer Nachricht notwendig ist, ist der Entscheidungsgehalt

**Informationsgehalt:**

* Je seltener ein Zeichen einer Quelle, desto grösser ist sein Informationsgehalt.
* Der Informationsgehalt eines Zeichens sagt aus, wie viele Elementarentscheidungen zur Bestimmung dieses Zeichens zu treffen sind.

**Entropie:**

* Die Entropie bezeichnet den mittleren Informationsgehalt der Quelle. Sie zeigt also auf, wie viele Elementarentscheidungen die Quelle/Senke im Mittel pro Zeichen treffen muss
* Die Entropie ist maximal, wenn beide Symbole mit gleicher Wahrscheinlichkeit auftreten oder wenn die Entropie 1 bit pro Symbol beträgt, d.h die Unsicherheit über das nächste Symbol ist am grössten

**Absolute Redundanz (Redundanz der Quelle):** Entscheidungsgehalt (H0) – Entropie (H(X))

Relative Redundanz:

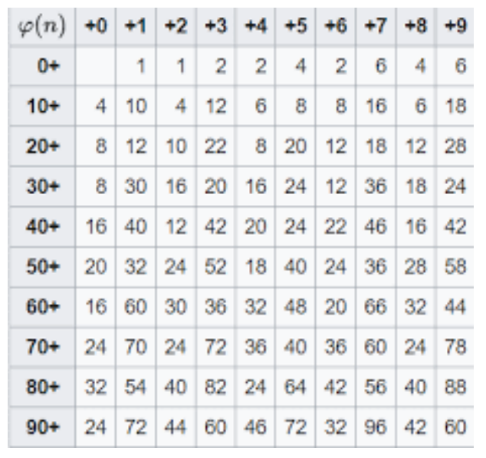
Redundanz des Codes:

**Mittlere Codewortlänge:**

* Bei der Quellencodierung werden die diskreten Zeichen der Quelle auf binäre CW abgebildet.
* Günstig ist, wenn die mittlere Codewortlänge 𝐿 möglichst klein ist.
* Ist abhängig von den Auftrittswahrscheinlichkeiten der Zeichen und der Codierung der Quelle

# Quellencodierung und Verschlüsselung – W9

Phi Tabelle:



Bei RSA ist die Faktorisierung grosser Primzahlen sehr aufwändig

# Quellencodierung und Komprimierung – W8

**Huffman-Codierung:**

* Berücksichtigt die Historie der Zeichen **nicht**
* Berücksichtigt den Informationsgehalt und die Auftrittswahrscheinlichkeiten der Zeichen der Quelle
* Ziel ist es Redundanz und Irrelevanz zu entfernen
* Ist statisch und adaptiv
* Minimale mittlere Codewortlänge (Niemals kleiner als die Entropie und kann grösser als die Entropie sein)

**Lempel Ziv (LZ77):**

* Dynamisches verfahren
* Erkennt wiederkehrende Phrasen
* Verwendet ein Sliding Window
* Kann grösser sein als der Originalcode

**RLE/RLC (Run Length Encoding/Run Length Coding):**

* Verkürzung von Wiederholungen (z.b bei TIFF)
* Agggbbehfffgggg => |w|=15
* A3g2b1e1h3f4g => | wc |= 13

**Präfixeigenschaft:**

* Ist kommafrei
* Kein Codewort ist der Anfang eines anderen Codeworts
* Die Zeichenfolge ist eindeutig dekodierbar, auch ohne Trennzeichen

**Shannon’sches Codierungstheorem:**

* Das Shannon'sche Codierungstheorem besagt, dass jede Quellensprache mit einer Entropie H, mit einer durchschnittlichen Codewortlänge L codiert werden kann, wobei L ≥ H.
* Für jede beliebige zugehörige Binärcodierung mit Präfixeigenschaft ist die mittlere Codewortlänge nicht kleiner als die Entropie 𝐻(𝑋): H(X) =< L
* Für jede beliebige Quelle kann eine Binärcodierung gefunden werden, so dass die folgende Ungleichung gilt: H(X) =< L =< H(X) + 1

# Kanalmodell – W10

* Ein nicht gestörter Kanal entspricht der Einheitsmatrix und überträgt den mittleren Informationsfluss ohne Verlust 🡪 Transinformation wird durch Quelle bestimmt
* Veränderte Entropie = Veränderte Transinformation
* Höhere Fehlerwahrscheinlichkeit = verringerte Transinformation
* Transinformation: Gibt den maximalen (ergo fehlerfreien) Informationsfluss über den gestörten Kanal an
* Ein vollständig gestörter Kanal ist, wenn alles gleich ist in der Matrix:
* Äquivokation: Ungewissheit über das gesendete Zeichen bei bekanntem Empfangszeichen
  + Ist der Kanal fehlerfrei, so ist die Äquivokation (Rückschlussentropie) gleich 0
* Irrelevanz/Streuentropie: Ungewissheit der empfangenen Zeichen bei vorgegebenen Sendezeichen

# Blockcodes – W11

Prüfmatrix auslesen:

A number of numbers on a white background

AI-generated content may be incorrect.

Kontrollstellen k = 4, Gültige Codewörter 2^m = 2^11 = 2048,

**Dichtgepackter Code:**

* Alle gültigen und ungültigen Codewörter sind in einer Korrigierkugel

**CRC:**

* Nutzt Polynomdivision
* Das Ergebnis ist eine Prüfsumme

# Faltungscode – W12

* Speichert vergangene Eingaben mit Hilfe von Schieberegistern
* Zum Decodieren wird oft der Viterbi-Algorithmus eingesetzt

# Glossar

## Gruppen, Ring und Körper / Text, Festkomma- und Gleitkommazahlen (U3 & U4)

File: conversions.py

|  |  |  |
| --- | --- | --- |
| Thema | Funktion | Output |
| Division von Binärzahlen | Notation: u⁴ + u² + u + 1 → [1, 0, 1, 1, 1]  div([1, 0, 0, 0, 1], [1, 0, 1]) | ([Quotient], [Rest]) |
| Addition von Binärzahlen | bad("10111111", "11000011") |  |
| Subtraktion von Binärzahlen | bs("1111", "1010") |  |
| Polynom-Addition in Z\_2 | ad2([1, 0, 0, 0, 1], [0, 1, 0, 1, 1]) | [1, 1, 1, 0, 0] d.h Codewort: 11100 |
| Polynom-Multiplikation in Z\_2 | mu2([1, 0, 0, 0, 1], [0, 1, 0, 1, 1]) | [1, 1, 1, 0, 0] d.h Codewort: 11100 |
| Polynom-Division in Z\_2 (long division) | divl([1, 0, 0, 0, 1], [0, 1, 0, 1, 1]) | ([Quotient], [Rest]) |
| Vektoraddition in Z\_2 | addv([[1, 1, 0], [0, 0, 1], [1, 1, 1], [1, 0, 1], [0, 0, 1]]) | [1, 0, 0] |
| Erweiterungskörper (elemente die durch polynom erzeugt werden können) | gfe([1, 0, 0, 1, 1]) -> x^4 + x + 1 | Aufkettung von den Elementen  Zykluslänge |
| Reduzible Polynome  (Ist ein polynom reduzibel?) | isr([1, 1, 0]) -> x² + x | true/false |
| Darstellung negativer Zahlen (bin) | cvb("11111") | Betrag, Betrag mit Vorzeichen, Exzess-4, b-1 (1erKompl.), b (2erKompl.) |
| Komplement von Dezimalzahlen | 9er Komplement: nk(1234)  10er: nk(1234) + 1 |  |
| Addition von Komplementen (dez) | 9er: ank(-2, 1)  10er: azk(-2, 1) |  |
| Exzess Darstellung | 1. Zahl: Zahl  2. Zahl: Exzess (Bias)  3. Wortlänge (default=8)  Dezimal zu Binär: de(34, 2, 4)  Binär zu Dezimal: ed("11111111", 2)  Achtung: Das Minus ist implizit, für Exzess--4 also -4eingeben |  |
| Kleinste Festkommazahl | kfi(8, 2)  Gesamt Bits: 8 (default=8)  Vorkomma bits: 2 (default=2) |  |

## Informationstheorie / Quellencodierung und Komprimierung (U7/U8)

File: informationstheorie.py

|  |  |  |
| --- | --- | --- |
| Thema | Funktion | Output |
| Diskrete Quelle ohne Gedächtnis (Achtung: Funktioniert nicht mit nur einer Wahrscheinlichkeit, dann ist Entropie = 1) | bq([0.3, 0.1, 0.1, 0.2, 0.3]) | Entropie  Informationsgehalt  Entscheidungsgehalt  Redundanz |
| Diskrete Quelle mit Gedächtnis | bqg([      [0.1, 0.5, 0.4],      [0.4, 0.2, 0.4],      [0.3, 0.3, 0.4]  ]) |  |
| Codierung - bei gegebener Wahrscheinlichkeit | ac([      ["a", "0", 0.3],      ["b", "110", 0.1],      ["c", "1111", 0.1],      ["d", "1110", 0.2],      ["e", "10", 0.3]  ]) |  |

## Blockcodes und Faltungscodes (U11/12)

File: Block-Fatlungs\_codes.py

|  |  |  |
| --- | --- | --- |
| Thema | Funktion | Output |
| Infos zu Blockcodes: | binf(10, 5, 4)  1. m=Nachrichtenstellen  2. k=Kontrollstellen  3. h=Hammingdistanz | -Anzahl gültige/mögliche Codewörter  -Sicher erkennbare Fehlerzahl und sicher korrigierbare Fehler  -Dichtgepackt (Ja/Nein) |
| Zyklischer Hammingcode für Generatorpolynom | haut([1, 0, 1, 1]) entspricht g(x) = 1 + x + x³ |  |
| CRC Code | rcc([1, 1, 0, 0, 1]) -> 1 + x + x^4 |  |
| Faltungscodes: GSM | gsm([1, 1, 0], [1, 1, 1], 185)  1 + x^3 + x^4 => [1, 0, 0, 1, 1] |  |

## Kanalmatrix

File: Kanalmodell.py

|  |  |  |
| --- | --- | --- |
| Thema | Funktion | Output |
| Kanalmatrix basic berechnung | bk([[0.9, 0.1],[0.1, 0.9]], [0.3, 0.7], 1000)  1. Kanalmatrix, 2. Auftrittswahrscheinlichkeit, 3. Übertragungsrate (1kbit/s = 1000) |  |
| Entscheider und Fehlerwahrscheinlichkeit | euf([[0.2, 0.5, 0.3], [0.7, 0.2, 0.1], [0.4, 0.0, 0.6]], [550, 1200, 3000]) |  |
| Kanalmatrix bei gegebenen Wahrscheinlichkeiten (Symmetrisch) | btk([0.3, 0.7], [0.34, 0.66], 140, 500)  p\_x = [0.3, 0.7] p\_y = [0.34, 0.66] kanalrate\_kbps = 140 blocksize\_mbit = 500 | Irrelevanz  Ausgangsentropie  Transinformation  P(Y|X) |