Übung 1

Zahlencodierung & binäre Grundlagen

Lösung

V2.1

Hinweise:

- Übungen sind mit Vorteil alleine zu lösen
- Benutzen Sie die Musterlösung nur zur Korrektur
- Die Übungen sind wichtige Vorbereitungen für die Prüfung. Lösen Sie die Übungen sorgfältig und stellen Sie die Lösungswege übersichtlich dar.
- (Ergänzte) Vorlesungsunterlagen und Fachbücher helfen beim Lösen von Übungen und bringen gleichzeitig eine erweiterte Ansicht auf die Problemstellung.
- Wenn Sie die Übungen nicht verstehen, fragen Sie!

Ziele

In diesen Übungen geht es darum, Ihnen den Umgang mit Binär- und Hexadezimalzahlen natürlich werden zu lassen, da diese Zahlen in der Informatik ständig vorkommen. Dezimalzahlen hingegen spielen so gut wie keine Rolle. Verwenden Sie keine Hilfsmittel.

Aufgabe 1 Dezimalzahlen als Polynom

Stellen Sie die folgenden Dezimalzahlen als Polynom dar.

Z.B. 111 = $1 \times 10^2 + 1 \times 10^1 + 1 \times 10^0$

a) 345 $3 \times 10^2 + 4 \times 10^1 + 5 \times 10^0$

b) 9999 $9 \times 10^3 + 9 \times 10^2 + 9 \times 10^1 + 9 \times 10^0$

c) 10000 1×10^4 da die anderen Stellen 0 sind, tragen sie nicht zur Summe bei

d) 701 $7 \times 10^2 + 0 \times 10^1 + 1 \times 10^0$ oder $7 \times 10^2 + 1 \times 10^0$

e) 0 0×10^0

Aufgabe 2 Dualzahlen als Polynom

Stellen Sie die folgenden Dualzahlen zunächst als Polynom mit Zweierpotenzen und dezimalem Exponenten, dann als Summe von Dezimalzahlen und schliesslich als Dezimalzahl dar

Z.B.: 0101 = $2^2 + 2^0 = 4 + 1 = 5_d$

a) 0001 $= 2^0 = 1_d$

b) 0010 $= 2^1 = 2_d$

c) 1000 $= 2^3 = 8_d$

d) 0100 $= 2^2 = 4_d$

e) 0011 $= 2^1 + 2^0 = 2 + 1 = 3_d$

f) 0110 $= 2^2 + 2^1 = 4 + 2 = 6_d$

g) 1100 $= 2^3 + 2^2 = 8 + 4 = 12_d$

h) 1111 $= 2^3 + 2^2 + 2^1 + 2^0 = 8 + 4 + 2 + 1 = 15_d$

i) 1010 $= 2^3 + 2^1 = 8 + 2 = 10_d$

j) Was ist der Unterschied zwischen der Binärzahl 0000 und 0000'0000?
Beide Zahlen stellen denselben Wert dar, nämlich 0. Allerdings bezieht sich 0000 auf 4 Bit,

0000’0000 jedoch auf 8 Bit. Diese Unterscheidung ist wichtig, da durch die zusätzlichen Nullen auch mehr Speicherplatz impliziert wird.

Aufgabe 3 Dezimalzahl in Dualzahl umformen

Wandeln Sie folgende Dualzahlen in Dezimalzahlen um. Stellen Sie die Zahl zuerst als Polynom dar

a) 1011 $1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 8 + 0 + 2 + 1 = 11$

b) 0011 0101 $0 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 0 + 0 + 32 + 16 + 0 + 4 + 0 + 1 = 53$

c) 1 $1 \times 2^0 = 1$

d) 1111 1111 $1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255$

e) 0 $0 \times 2^0 = 0$

f) 0001 0000 0000 $0 \times 2^{11} + 0 \times 2^{10} + 0 \times 2^9 + 1 \times 2^8 + 0 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 0 + 0 + 0 + 256 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 = 256$

g) 1010 1010 $1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 128 + 0 + 32 + 0 + 8 + 0 + 2 + 0 = 170$

Diese Methode zeigt, wie jede Position in einer Dualzahl einem bestimmten Wert in der Dezimalzahl entspricht – basierend auf Zweierpotenzen.

Aufgabe 4 Dualzahl in Dezimalzahl umformen

Wandeln Sie folgende Dezimalzahlen in Dualzahlen um.

a) 2024 0111 1110 1000

b) 255 1111 1111

c) 11 1011

d) 658345 1010 0000 1011 1010 1001

e) 00067340 0001 0000 0111 0000 1100

f) 5 0101

g) 257 0001 0000 0001

Diese Dualzahlen sind das Ergebnis der schrittweisen Teilung der Dezimalzahlen durch 2 und der Bildung der Binärzahl durch Anordnung der Reste in umgekehrter Reihenfolge.

Aufgabe 5 Zaubertrick (aus der zweiten Klasse Primarschule)

Der Zauberkünstler bittet einen Zuschauer, sich eine Zahl n zwischen 0 und 31 auszudenken (und ggf. zu notieren). Danach zeigt er ihm nacheinander folgende fünf Tabellen mit der Bitte jeweils zu sagen, ob n in der jeweiligen Tabelle enthalten sei.

Tabelle 1:

1	3	5	7
9	11	13	15
17	19	21	23
25	27	29	31

Tabelle 2:

2	3	6	7
10	11	14	15
18	19	22	23
26	27	30	31

Tabelle 3:

4	5	6	7
12	13	14	15
20	21	22	23
28	29	30	31

Tabelle 4:

8	9	10	11
12	13	14	15
24	25	26	27
28	29	30	31

Tabelle 5:

16	17	18	19
20	21	22	23
24	25	26	27
28	29	30	31

Zur Überraschung der Zuschauer kann der Zauberkünstler jedes Mal die Zahl richtig erraten. Erklären Sie mithilfe des Binärsystems, wie dieser Trick funktioniert. Versuchen Sie dazu zunächst Gemeinsamkeiten der Zahlen innerhalb einer Tabelle zu erkennen (im Zweifelsfall stellen Sie alle Zahlen einer Tabelle als Binärzahlen dar). Entwickeln Sie ein Verfahren, wie der Zauberkünstler aus der gegebenen Information (ja/nein = 1 Bit) die Zahl ermitteln kann.

Wir können jede Tabelle als Bit einer 5-Bit-Dualzahl betrachten. In Tabelle 1 sind alle Zahlen, deren LSB auf 1 ist. In der Tabelle 2 sind alle Zahlen, die an der Stelle 2 in Binärdarstellung ein gesetztes Bit haben usw.

Das Verfahren für den Zauberkünstler besteht also darin, immer dann 2^{m-1} zu addieren, wenn der Zuschauer bei Tabelle m «Ja» sagt. Er nimmt also eine Umformung der fünfstelligen Binärzahl in das Dezimalsystem vor, die dadurch entsteht, das jedem «Ja» eine 1 und jedem «Nein» eine 0 zugeordnet wird.

Aufgabe 6 gebrochene Dezimalzahl in Dualzahl umwandeln

Wandeln Sie folgende Dezimalzahlen in Dualzahlen um. Zeigen Sie den Lösungsweg für die Umwandlung.

a) 0.625 0.101

b) 0.375 0.011

c) 0.3 $0.0\overline{1001}$

d) 0.16 $0.\overline{00101000111101011100}$

e) 0.4 $0.\overline{0110}$

f) 0.9 $0.1\overline{1100}$

Bei diesen Umwandlungen ist zu beachten, dass einige Dezimalzahlen keine exakte Darstellung im Binärsystem haben und daher als periodische Binärzahlen dargestellt werden. Die angegebenen Binärzahlen sind daher Näherungen.

Aufgabe 7 gebrochene Dualzahl in Dezimalzahl umwandeln

Wandeln Sie folgende Dualzahlen in Dezimalzahlen um. Stellen Sie zuerst die Zahl zuerst als Polynom dar. Berechnen Sie anschliessend die Lösung.

a) 0.101 $= 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} = 0.5 + 0.125 = 0.625$

b) 0.011 $= 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} = 0.25 + 0.125 = 0.375$

c) 0.0011 $= 0 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4} = 0.125 + 0.0625 = 0.1875$

d) 0.1101 $= 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4} = 0.5 + 0.25 + 0.0625 = 0.8125$

e) 0.110011 $= 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 0 \times 2^{-4} + 1 \times 2^{-5} + 1 \times 2^{-6}$
$= 0.5 + 0.25 + 0.03125 + 0.015625 = 0.796875$

f) 0.11011011 $= 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4} + 1 \times 2^{-5} + 0 \times 2^{-6} + 1 \times 2^{-7} + 1 \times 2^{-8}$
$= 0.5 + 0.25 + 0.0625 + 0.03125 + 0.0078125 + 0.00390625 = 0.85546875$

Diese Berechnungen zeigen, wie jede Position einer gebrochenen Dualzahl als negative Zweierpotenz betrachtet wird, um die entsprechende Dezimalzahl zu erhalten.

Aufgabe 8 Addition von Dualzahlen

Führen Sie folgende Rechnungen durch.

a) $0101\ 1101_2 + 1001\ 0110_2$ $1111\ 0011$

b) $111_2 + 101_2$ 1100

c) $0100\ 0110_2 + 1010_2$ 0101 0000

d) $111_2 + 111_2$ 1110

Aufgabe 9 Welche Zahl verbirgt sich dahinter

Ein Computersystem verwendet eine 8-Bit-Architektur und arbeitet mit dem 2er-Komplement-Verfahren zur Darstellung von vorzeichenbehaftetet Zahlen. Welche Dezimalzahl verbirgt sich hinter der gegebenen Dualzahl?

a) 1100 1111 1 Bit = 1 -> negative Zahl
Umkehrung des 2er-Komplements: N(1100 1111 – 1) = N(1100 1110) = 0011 0001 = 0011 0001 = 49_{10}, Lösung: -49

b) 1000 0010 1 Bit = 1 -> negative Zahl
Umkehrung des 2er-Komplements: N(1000 0010 – 1) = N(1000 0001) = 0111 1110 = 0111 1110 = 126_{10}, Lösung: -126

c) 0100 1111 1 Bit = 0 -> positive Zahl 0100 1111 = 79_{10}

Hinweis:
Die Lösung verwendet den Algorithmus: 1 von der neg. Zahl subtrahieren und anschliessend Flip der Bits.
In der Vorlesung haben wir folgendes verwendet: Flip der Bits und anschliessend 1 addieren.
Es ist zu sehen, dass beide Algorithmen zum korrekten Ergebnis führen.

Aufgabe 10 Subtraktion durch Addition

Führen Sie folgende Subtraktionen durch, indem Sie diese durch eine Addition ersetzen. Bestimmen Sie zuerst das entsprechende Komplement (additives Inverse) und führen Sie anschliessend die Addition durch.

a) $75_{10} – 26_{10}$ 10er-Komplement von 26 = 99 – 26 + 1 = 74
75 + 74 = 149 Überlauf ignorieren = 49

b) $134_{10} – 58_{10}$ 10er-Komplement von 058 = 999 – 58 + 1 = 942
134 + 942 = 1076 Überlauf ignorieren = 076 = 76

c) $25207_{10} – 993_{10}$ 10er-Komplement von 00993 = 99999 – 993 + 1 = 99007
25207 + 99007 = 124214 Überlauf ignorieren = 24214

d) $1111_2 – 1010_2$ 2er-Komplement von 1010 = N(1010) + 1 = 0101 + 1 = 0110
1111 + 0110 = 10101 Überlauf ignorieren = 0101 = 101

e) $101111_2 – 11001_2$ 2er-Komplement von 011001 = N(011001) + 1 = 100110 + 1 = 100111
101111 + 1001111 = 1010110 Überlauf ignorieren = 010110 = 10110

f) $1101_2 - 1101_2$ 2er-Komplement von 1101 = N(1101) + 1 = 0010 + 1 = 0011
1101 + 0011 = 10000 Überlauf ignorieren = 0000 = 0

g) $71_8 - 25_8$ 8er-Komplement von 25 = 77 - 25 + 1 = 52 + 1 = 53
71 + 53 = 144 Überlauf ignorieren = 44

h) $174_8 - 54_8$ 8er-Komplement von 054 = 777 - 54 + 1 = 723 + 1 = 724
174 + 724 = 1120 Überlauf ignorieren = 120

Aufgabe 11 Multiplikation

Führen Sie folgende signed Multiplikationen (4-Bit) durch.

a) 1100 x 1001 = 0110 1100, da wir mit 4 Bit rechnen, entsteht ein Überlauf.
Das effektive Ergebnis ist 1100 = -4

b) 1010 x 0011 = 0001 1110, 4Bit: 1110 = -2

c) 0010 x 1111 = 1111 1110, 4Bit: 1110 = -2

d) 0111 x 0010 = 0000 1110, 4Bit: 1110 = -2

Feststellung: Die Multiplikation funktioniert ohne explizite Bildung des 2er-Komplements, weil die Operanden bereits im 2er-Komplement vorliegen und die Operation als wiederholte Addition im Ring $\mathbb{Z}/2^n\mathbb{Z}$ ausgeführt wird.
Korrekterweise müssten die Resultate als 8-Bit-Ergebnisse betrachtet werden.

Aufgabe 12 Rechnen mit Polynom

Führen Sie folgende Operationen aus, indem Sie die Zahlen zuerst in Polynome umwandeln und die Operation anschliessend mit den Polynomen durchführen.

Bsp: 1101 + 110 $= (2^3 + 2^2 + 2^0) + (2^2 + 2^1) = 2^3 + 2 \times 2^2 + 2^1 + 2^0 = 8 + 2 \times 4 + 2 + 1 = 19$

a) 101 + 110 $(2^2 + 2^0) + (2^2 + 2^1) = 2 \times 2^2 + 2^1 + 2^0 = 8 + 2 + 1 = 11$

b) 110 – 011 $(2^2 + 2^1) - (2^1 + 2^0) = 2^2 - 2^0 = 4 - 1 = 3$

c) 101 x 111 $(2^2 + 2^0) \times (2^2 + 2^1 + 2^0) = 2^4 + 2^3 + 2^2 + 2^2 + 2^1 + 2^0$
$= 16 + 8 + 4 + 4 + 2 + 1 = 35$

Aufgabe 13 Stellenwertsystem und römisches Zahlensystem

Erklären Sie die Unterschiede zwischen einem Stellenwertsystem und dem römischen Zahlensystem. Überlegen Sie sich zuerst die Eigenschaften eines Stellenwertsystems und vergleichen Sie diese mit dem römischen Zahlensystem.

Das Stellenwertsystem basiert auf der Position der Ziffern und verwendet eine Basis (wie 10), während das römische System feste Symbole für bestimmte Werte verwendet, ohne eine Basis oder Stellenwerte zu berücksichtigen. Die Position dient dort für die Entscheidung ob Addition oder Subtraktion der Zahl.

Aufgabe 14 Interpretation einer Menge von Bits

Interpretieren Sie folgende Menge an Bits als Binärzahl, Tupel, Menge, Vektor, Polynom sowie Zustand:

1101 0101

a) Binärzahl 11010101

b) Tupel $(1, 1, 0, 1, 0, 1, 0, 1)$

c) Menge $\{0, 1\}$

d) Vektor $\begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$

e) Polynom $1 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$

f) Zustand Könnte ein Zustand eines Speicherregisters (Adressierung, Anweisung, usw.) sein.

Aufgabe 15 Hexadezimalzahl als Polynom

Stellen Sie folgende Hexadezimalzahlen als Polynom zur Basis 16 dar.

a) $2B3_h$ $2 \cdot 16^2 + 11 \cdot 16^1 + 3 \cdot 16^0 = 691$

b) $C4E_h$ $12 \cdot 16^2 + 4 \cdot 16^1 + 14 \cdot 16^0 = 3150$

c) $3F4C9_h$ $3 \cdot 16^4 + 15 \cdot 16^3 + 4 \cdot 16^2 + 12 \cdot 16^1 + 9 \cdot 16^0 = 259273$

Aufgabe 16 Hexadezimal und Binär

Bestimmen Sie die angegebenen Bits der gegebenen Hexadezimalzahl als Binärzahl. Stellen Sie diese dann als Hexadezimalzahl dar.
Beispiel: Bits 8 bis 2 von 1234_h
Lösung: Lösung: $b = 1234_h = 0001'001$**0'0011'01**00_b, $b8 \ldots b2 = 000'1101b = 0D_h$

f) Bits 14 bis 3 von $9AE3_h$
$b = 9AE3_h = 1$**001'1010'1110'0**011_b, $b14 \ldots b3 = 0011'0101'1100b = 35C_h$

a) Bits 20 bis 6 von $A1B2C3_h$
$b = A1B2C3_h = 101$**0'0001'1011'0010'11**$00'0011_b$, $b20 \ldots b6 = 000'0110'1100'1011_b = 06CB_h$

b) Bits 11 bis 4 von $CAFE_h$
$b = CAFE_h = 1100'$**1010'1111**$'1110_b$, $b11 \ldots b4 = 1010'1111_b = AF_h$

Aufgabe 17 Hexadezimal ohne Binär

k) An welchen Stellen und an welchen Bits stehen die beiden Ziffern 4 und A in der Zahl $78A489_h$?
4 steht an Stelle 2 bzw. Bits 11 bis 8. A steht an Stelle 3 bzw. Bits 15 bis 12.

Ziffer:	7	8	A	4	8	9
Stelle:	5	4	3	2	1	0
Bits:	23 ... 20	19 ... 16	15 ... 12	11 ... 8	7 ... 4	3 ... 0

l) Bestimmen Sie direkt, also ohne ins Binärsystem umzurechnen, die Bits 11 bis 4 von $CAFE_h$.
Es sind die Bits der beiden Stellen 2 (Bits 11 bis 8) und 1 (Bits 7 bis 4), also AF_h.

m) Bestimmen Sie direkt, also ohne ins Binärsystem umzurechnen, die Bits 23 bis 8 von $6709AFFE_h$.
Es sind die Bits der Stellen 5 (Bits 23 bis 20) bis 2 (Bits 11 bis 8), also $09AF_h$.

Aufgabe 18 Rechnen mit Zweierpotenzen in Präfixschreibweise

Berechnen Sie ohne Hilfsmittel und stellen Sie das Ergebnis in Präfixschreibweise dar.

h) 128 x 64 $2^7 \times 2^6 = 2^{7+6} = 2^{13} = 8K$

i) 4G x 2K $2^{32} \times 2^{11} = 2^{32+11} = 2^{43} = 8T$

j) 256 x 2M $2^8 \times 2^{21} = 2^{8+21} = 2^{29} = 512M$

k) 8T / 64K $2^{43} / 2^{16} = 2^{43-16} = 2^{27} = 128M$

l) 128K / 512 $2^{17} / 2^9 = 2^{17-9} = 2^8 = 256$

m) 4M / 256 $2^{22} / 2^8 = 2^{22-8} = 2^{14} = 16K$

n) Ein Computer habe 32GB Hauptspeicher. Wieviele Datenstrukturen zu je 512K würden in diesen Speicher passen? $32GB / 512K = 2^{35} / 2^{19} = 2^{35-19} = 2^{16} = 64K$

Aufgabe 19 Zahlenbereiche je Bits

Wie viele Zahlen kann man jeweils mit den folgenden Bits darstellen? Stellen sie die Ergebnisse in Präfixschreibweise dar. Geben Sie auch die kleinste und grösste Zahl in Hexadezimalschreibweise an.

h) 2 Bit 2^2 = 4 Zahlen – Zahlenbereich von 0 bis 3.

i) 4 Bit 2^4 = 16 Zahlen – Zahlenbereich von 0 bis F.

j) 7 Bit 2^7 = 128 Zahlen – Zahlenbereich von 00 bis 7F.

k) 8 Bit 2^8 = 256 Zahlen – Zahlenbereich von 00 bis FF.

l) 9 Bit 2^9 = 512 Zahlen – Zahlenbereich von 000 bis 1FF.

m) 10 Bit 2^{10} = 1K Zahlen – Zahlenbereich von 000 bis 3FF.

n) 11 Bit 2^{11} = 2K Zahlen – Zahlenbereich von 000 bis 7FF.

o) 12 Bit 2^{12} = 4K Zahlen – Zahlenbereich von 000 bis FFF.

p) 13 Bit 2^{13} = 8K Zahlen – Zahlenbereich von 0000 bis 1FFF.

q) 16 Bit 2^{16} = 64K Zahlen – Zahlenbereich von 0000 bis FFFF.

r) 24 Bit 2^{24} = 16M Zahlen – Zahlenbereich von 00’0000 bis FF’FFFF.

s) 32 Bit 2^{32} = 4G Zahlen – Zahlenbereich von 0000’0000 bis FFFF’FFFF.

a) 64 Bit 2^{64} = 16E Zahlen (sehr ungewöhnlich, wird so kaum verwendet) – Zahlenbereich von 0000’0000’0000’0000 bis FFFF’FFFF’FFFF’FFFF.

Aufgabe 20 Bits per Zahlenbereich

Wie viele Bits benötigt man, um die folgenden Anzahlen an verschiedenen Zahlen darstellen zu können?

a) 15d $8 < 15 \leq 16 = 2^4 \Rightarrow$ 15 Zahlen benötigen 4 Bit

b) 16d $8 < 16 \leq 16 = 2^4 \Rightarrow$ 16 Zahlen benötigen 4 Bit

c) 17d $16 < 17 \leq 32 = 2^5 \Rightarrow$ 17 Zahlen benötigen 5 Bit

d) 100d $64 < 100 \leq 128 = 2^7 \Rightarrow$ 100 Zahlen benötigen 7 Bit

e) 1000d $512 < 1000 \leq 1K = 2^{10} \Rightarrow$ 1000 Zahlen benötigen 10 Bit

f) 10'000d $8K < 10'000 \leq 16K = 2^{14} \Rightarrow$ 10'000 Zahlen benötigen 14 Bit

g) 100'000d $64K < 100'000 \leq 128K = 2^{17} \Rightarrow$ 100'000 Zahlen benötigen 17 Bit

h) 1'000'000d $512K < 1'000'000 \leq 1M = 2^{20} \Rightarrow$ 1'000'000 Zahlen benötigen 20 Bit

Die benötigte Anzahl an Bits ergibt sich aus dem Logarithmus zur Basis 2 der Anzahl der darzustellenden Zahlen: $Anzahl\ Bits = \lceil \log_2(Anzahl\ Zahlen) \rceil$
Der Wert wird dabei auf die nächste ganze Zahl aufgerundet, da die Anzahl der Bits nur in ganzen Zahlen vorliegen kann.

Aufgabe 21 Datum und Zeit im UNIX-System

In UNIX Systemen wird - aus historischen Gründen - die Zeit in Sekunden seit dem 1. Januar 1970, 00:00 Uhr gezählt. In welchem Jahr gibt es Probleme mit 32-Bit-Maschinen, wenn die Zahl vorzeichenbehaftet gespeichert ist? Wann würde das Problem auftreten, wenn die Zahl vorzeichenlos gespeichert wird?

Vorzeichenbehaftete 32bit-Zahl = -2^{31} bis $+2^{31}$.
Entsprechend können mit 31Bit 2^{31} = 2'147'483'647 Sekunden gezählt werden. Dies entspricht 2'147'483'647/(60*60*24*365) = 68 Jahre und 35 Tage. – Somit tritt das Problem im Jahre 2038 auf. Unter Berücksichtigung der Schaltjahre sind es 68 Jahre und 19 Tage. Entsprechend tritt das Problem genau am 19. Januar 2038 auf. Dieses bekannte Problem wird auch als das "Jahr-2038-Problem" oder "Y2K38" bezeichnet.

Vorzeichenlose 32bit-Zahl = 0 bis $+2^{32}$.
2^{32} s entsprechen 136 Jahre und 31 Tage (unter Berücksichtigung der Schaltjahre). Somit tritt das Problem am 07. Februar 2106 auf. Ohne Berücksichtigung der Schaltjahre: 135 Jahre und 42 Tage.

Aufgabe 22 Oktal- und Dezimalzahlen

Die Ziffern von Oktalzahlen sind zur Basis 8 zu interpretieren anstelle der Basis 10 bei Dezimalzahlen.

a) Wandeln Sie die Oktalzahl 144_8 in eine Dezimalzahl um.
$1 * 8 * 8 + 4 * 8 + 4 = 64 + 32 + 8 = 100_d$

b) Wandeln Sie die Dezimalzahl 135_d in eine Oktalzahl um.
$135/8 = 16 * 8 + 7 = 2 * 8 * 8 + 0 * 8 + 7 = 207_8$

Aufgabe 23 Oktal- und Hexadezimalzahlen

Oktal- und Hexzahlen lassen sich einfach von und nach Binärzahlen umrechnen. Verwenden Sie diesen Umweg für die folgenden Umrechnungen.

a) Wandeln Sie die Oktalzahl 65_8 in eine Hexadezimalzahl um.
$65_8 = 110'101_2 = 35_h$

b) Wandeln Sie die Hexzahl $7b_h$ in eine Oktalzahl um.
$0111'1011_2 = 173_8$

a) Wieso sind die Umrechnungen von und nach Binärzahlen für Oktal- und Hexzahlen einfacher als für die meisten anderen Zahlensysteme?
Weil die Basis eine ganzzahlige Potenz der Binärbasis 2 ist und somit direkt mehrere Binärterme zusammengefasst werden können.

Übung 2

Spezielle Codierungen: Vorzeichen, Gleitkomma und Text

Lösung

V2.0

Hinweise:

- Übungen sind mit Vorteil alleine zu lösen
- Benutzen Sie die Musterlösung nur zur Korrektur
- Die Übungen sind wichtige Vorbereitungen für die Prüfung. Lösen Sie die Übungen sorgfältig und stellen Sie die Lösungswege übersichtlich dar.
- (Ergänzte) Vorlesungsunterlagen und Fachbücher helfen beim Lösen von Übungen und bringen gleichzeitig eine erweiterte Ansicht auf die Problemstellung.
- Wenn Sie die Übungen nicht verstehen, fragen Sie!

Aufgabe 1 Darstellung negativer Zahlen

Vervollständigen Sie die Tabelle.

Bin	Betrag	Betrag mit Vorzeichen	Exzess-4	(b-1) 1erKompl.	(b) 2erKompl.
000	0	+0	-4	0	0
001	1	1	-3	1	1
010	2	2	-2	2	2
011	3	3	-1	3	3
100	4	-0	0	-3	-4
101	5	-1	1	-2	-3
110	6	-2	2	-1	-2
111	7	-3	3	0	-1

Aufgabe 2 (b-1)-Komplement

Um das Konzept des (b-1)-Komplement-Systems auf ein anderes Zahlensystem als das Binärsystem zu übertragen, verwenden wir das Dezimalsystem (b=10). Berechnen Sie -1234+5000 unter Verwendung des 9er-Komplements (4 Ziffern).

Wir ersetzen jede Ziffer einer Zahl durch (9) minus den Wert der Ziffer. Dies ähnelt dem Vorgehen im Binärsystem, wo jede Binärziffer (Bit) durch (1) minus den Wert des Bits ersetzt wird.

9 - 1 = 8
9 - 2 = 7
9 - 3 = 6
9 - 4 = 5

9er-Komplement von 1234 = 8765
8765 + 5000 = 13765

Da das Ergebnis 13765 über 9999 liegt (dem maximalen Wert für vier Dezimalstellen im Neunerkomplement), führen wir einen End-Around Carry durch und schneiden den Überlauf ab.

13765 + 1 – 10000 = 3766

Aufgabe 3 (b)-Komplement

Führen Sie die Berechnung von Aufgabe 2 mittels (b)-Komplement (d.h. 10er-Komplement) durch.

Wir berechnen das 9er-Komplement und addieren +1 und erhalten so das 10er-Komplement: 8765+1 = 8766
Berechnung mittels 10er-Komplement: 8766 + 5000 = 13766, Überlauf abschneiden: 3766

Aufgabe 4 Exzess-Darstellung

Wir verwenden eine Wortlänge von 8Bit. Führen Sie folgende Berechnungen durch:

Dezimal zu binär

a) 34 nach Exzess-2 $d=|34-(-2)|=36$ $C_{Ex,-2,8}(34)=0010\ 0100_{Exzess-2}$

b) -2 nach Exzess-4 $d=|-2-(-4)|=2$ $C_{Ex,-4,8}(-2)=0000\ 0010_{Exzess-4}$

c) -5 nach Exzess-16 $d=|-5-(-16)|=11$ $C_{Ex,-16,8}(-5)=0000\ 1011_{Exzess-16}$

d) -21 nach Exzess-32 $d=|-21-(-32)|=11$ $C_{Ex,-32,8}(-21)=0000\ 1011_{Exzess-32}$

e) -21 nach Exzess-128 $d=|-21-(-128)|=107$ $C_{Ex,-128,8}(-21)=0110\ 1011_{Exzess-128}$

Binär zu Dezimal

f) $1111\ 1111_{Exzess-2}$ 1111 1111 = 255 255+(-2) = 253

g) $0000\ 0000_{Exzess--4}$ 0000 0000 = 0 0+(-4) = -4

h) $1111\ 1111_{Exzess-16}$ 1111 1111 = 255 255+(-16) = 239

i) $0000\ 0000_{Exzess-32}$ 0000 0000 = 0 0+(-32) = -32

j) $0111\ 1011_{Exzess-128}$ 0111 1011 = 123 123+(-128)=-5

Aufgabe 5 Berechnung mit Exzess-Codierung

Führen Sie die Berechnung von Aufgabe 2 mittels Exzess-Darstellung durch. Verwenden Sie 16Bit und den Bias mit gleichmässiger Verteilung. Was stellen Sie fest? Wie können Sie das Problem lösen?

$b = -2^{n-1} -1 = -2^{16-1} = -32767$

Codierung von -1234

$d = |-1234 - (-32767)| = 31533_{10} = 0111\ 1011\ 0010\ 1101_2$

$C_{Ex,-32767,16}(-1234) = 0111\ 1011\ 0010\ 1101_{Exzess-32767}$

Codierung von 5000

$d = |5000 - (-32767)| = 37767_{10} = 1001\ 0011\ 1000\ 0111_2$

$C_{Ex,-32767,16}(5000) = 1001\ 0011\ 1000\ 0111_{Exzess-32767}$

$0111\ 1011\ 0010\ 1101_{Exzess-32767}$
$+1001\ 0011\ 1000\ 0111_{Exzess-32767}$
$1\ 0000\ 1110\ 1011\ 0100$

Wir verwenden nur 16-Bit und haben entsprechend einen Überlauf.

Lösungen:

- Verwendung von 32Bit.
- Den Bias anders wählen (z.B. -2000)

Aufgabe 6 Effizienz der Rechenoperationen

Berechnen Sie die Addition -1 + 2.

Führen Sie diese Addition jeweils mit folgenden Darstellungsarten durch:

- Betrag mit Vorzeichen
- Exzess-4
- (b-1)-Komplement
- (b)-Komplement

Verwenden Sie jeweils 3-Bit.

Was stellen Sie fest?

Betrag mit Vorzeichen:

-1 = 101

+2 = 010

Da eine direkte Addition hier nicht sinnvoll ist, müssen wir die Operationen basierend auf dem Vorzeichen und Betrag durchführen:

1) Da wir eine negative Zahl mit einer größeren positiven Zahl addieren, wird das Ergebnis positiv sein. Also, wir subtrahieren den Betrag von −1 (also 1) vom Betrag von +2 (also 2), was 11 ergibt.
2) Das Ergebnis +1 wird demnach als 001 dargestellt.

Dies erfordert zusätzliche Schritte zur Handhabung von Überträgen und zur Bestimmung des Ergebnisvorzeichens.

Exzess-4:

-1 = 011

+2 = 110

011 + 110 = 1001

Da der Exzess in der Addition 2x drin ist, müssen wir diesen nun auch 2x wieder rausrechnen.
1001 = 9 – 2x4 = +1
Da wir aber nur 3 Bits verwenden, funktioniert das eigentlich nicht. Wobei es hier zufällig stimmt, weil das abschneiden des 4. Bit ein -8 entspricht – das ist aber wie erwähnt nur Zufall. Wenn wir 4 Bit verwenden würden, würde es korrekt funktionieren.

Verdeutlichung mit Dezimalzahlen:
A=2
B=-1
Exzess e = 4
Ae = A+e = (2+4)=6
Be = B+e = (-1+4)=3
Ae+Be = (A+e) + (B+e) = (2+4) + (-1+4) = 9
-> so ist zu sehen, dass der Exzess 2x drin ist und und entpsrechend für die Interpretation auch 2x rausgerechnet werden muss.

Die Addition ist einfach, jedoch ist die Berechnung der Exzess-Darstellung «aufwändig».

(b-1)-Komplement

-1 = 110 (001 flip = 110)

2 = 010

110 + 010 = 1000

Wir verwenden nur 3-Bit und schneiden dementsprechend den Übertrag ab. Allerdings müssen wir in (b-1)-Komplement-Systemen den Übertrag zur ursprünglichen Summe hinzufügen, was uns 001 = +1 ergibt.

(b)-Komplement

-1 = 111 (001 flip und +1 = 111)

2 =010

111 + 010 = 1001

Wir verwenden nur 3-Bit und ignorieren dementsprechend den Übertrag, was als Resultat 001 = +1 ergibt. Dies ist die effizienteste Variante, da die Berechnung des Komplements einfach ist und das Resultat nicht speziell behandelt werden muss. D.h. diese Operation kann direkt durchgeführt werden, ohne das Vorzeichen gesondert zu berücksichtigen, was die Hardware vereinfacht und beschleunigt.

Fazit:

Im Allgemeinen ist das **Zweierkomplement** die effizienteste Methode für binäre arithmetische Operationen, vor allem wegen seiner direkten Unterstützung von Addition, Subtraktion und einfachen Übertragungsregeln. Es reduziert die Komplexität bei der Hardware-Implementierung und ist weit verbreitet in Computerarchitekturen.

Die **Betrag-und-Vorzeichen-Darstellung** und das **1er-Komplement** sind weniger effizient für die Addition, da sie zusätzliche Schritte oder spezielle Behandlungen erfordern, was ihre Verwendung in modernen Systemen einschränkt.

Die **Exzess-3-Darstellung** bietet eine interessante Alternative für spezifische Anwendungen, insbesondere in Situationen, in denen ein begrenzter Wertebereich mit einer symmetrischen Darstellung um Null herum erwünscht ist, aber sie ist weniger verbreitet für allgemeine arithmetische Operationen.

Insgesamt bevorzugen die meisten digitalen Systeme das Zweierkomplement für die Darstellung und Manipulation von ganzen Zahlen, aufgrund seiner Effizienz und Einfachheit bei der Durchführung von arithmetischen Operationen.

Aufgabe 7 Fixkommazahlen

In dieser Aufgabe arbeiten wir mit unsigned 10-Bit-Fixkommazahlen, die im Bereich [0, 4) liegen. Beantworten Sie folgende Fragen:

a) Wie viele Stellen stehen links und rechts vom Komma?
Vor dem Komma müssen die Zahlen 0 bis 3 darstellbar sein (da 4 gemäss Vorgabe nicht mehr enthalten ist), also benötigen wir 2 Bit links vom Komma. Somit sind 8 Bit rechts davon.

b) Wie lauten die kleinste Zahl k > 0 und die grösste Zahl g in diesem Format? Geben Sie beide als (gemischten) Bruch und als Dezimalkommazahl an.
Um die Skalierung zu erreichen, nutzen wir die Tatsache, dass 1024 Werte (0 bis 2^{10}-1=1023) repräsentiert werden können und teilen den max. Wertebereich (4) durch die max. Anzahl repräsentierbarer Werte (1024). Diese kleinste Einheit wäre dann die gesuchte Zahl k.
Anderer Weg wäre, dass wir die kleinste Auflösung der Nachkomma-Bits verwende:

$$k = 2^{-8} = \frac{1}{256} = 0.00390625$$

Die grösse Zahl ist erreicht, wenn alle 10 Bits auf 1 gesetzt sind. Der grösste Wert ist 1023. Somit können wir berechnen

$$g = 1023 \times \frac{1}{256} = 3.99609375$$

oder

$$g = 4 - 2^{-8} = 3\frac{255}{256} = 3.99609375$$

c) Wie würden nun die Zahlen 0.001, 0.002, 0.003, 0.005 und 0.006 dargestellt?
$00.0000\ 0000 = 0$
$00.0000\ 0001 = 2^{-8} = 0.00390625$
$00.0000\ 0010 = 2 \times 2^{-8} = 2^{-7}$=0.0078125
0.0000 0001 stellt die Zahlen $\left[2^{-8} - \frac{2^{-8}}{2}, 2^{-8} + \frac{2^{-8}}{2}\right)$ dar = [0.001953125,0.005859375)
0.001 ist unter der Grenze, somit = 0.0000 0000
0.002 ist im Bereich, somit 0.0000 0001
0.003 ist im Bereich, somit 0.0000 0001
0.005 ist im Bereich, somit 0.0000 0001
0.006 ist über der Grenze aber < als $2^{-8} + \frac{2^{-8}}{2} + 2^{-8} (= obere\ Grenze\ von\ 0.0000\ 0010) =$ 0.009765625, somit 0.0000 0010

d) Was ergibt 0.002+0.003 in diesem Binärformat? Formen Sie das Ergebnis ins Dezimalsystem um. Führen Sie die Rechnung dezimal durch und bestimmen Sie die Differenz zwischen beiden Ergebnissen.
$0.002 + 0.003 \triangleq 00.0000\ 0001 + 00.0000\ 0001 = 00.0000\ 0010 = 2^{-7} = 0.0078125$
$0.002 + 0.003 = 0.005 = 00.0000\ 0001$
$0.0078125 - 0.005 = 0.0028125 = 00.0000\ 0001$

e) Was ergibt 0.005-0.003 in diesem Binärformat? Formen Sie das Ergebnis ins Dezimalsystem um. Führen Sie die Rechnung dezimal durch und bestimmen Sie die Differenz zwischen beiden Ergebnissen.
$0.005 - 0.003 \triangleq 00.0000\ 0001 - 00.0000\ 0001 = 00.0000\ 0000 = 0$
$0.005 - 0.003 = 0.002 = 00.0000\ 0001$
$0 - 0.002 = 0.00390625 = 00.0000\ 0001$

Aufgabe 8 Fixkommazahlen: Fehlerfortpflanzung

Ein Mikrocontroller steuert eine industrielle Dosierpumpe, die Flüssigkeit in einen Behälter pumpt. Die Pumpe dosiert in Millilitern (ml), aber der Mikrocontroller speichert Werte in $C_{FK,4,16}$-Fixkommadarstellung.

Der Durchfluss F wird als Sensorwert erfasst, und die Zeit t gibt die Pumpdauer in Sekunden an. Das Mikrocontroller-System berechnet die gesamte gepumpte Flüssigkeitsmenge mit der Formel:

$$V = F \cdot t$$

wobei:

- $\boldsymbol{F = 1.3}\ ml/s$ (Durchflussrate des Sensors)
- $\boldsymbol{t = 3.7}\ s$ (Pumpdauer)

a) Wandeln Sie F und t in $C_{FK,4,16}$-Fixkommadarstellung um und berechnen Sie für beide Werte den absoluten sowie den relativen Rundungsfehler

$$F = 1.3 \approx 0001.0100 = 1.25$$
$$t = 3.7 \approx 0011.1011 = 3.6875$$
$$E_{abs}(F) = |1.3 - 1.25| = 0.05$$
$$E_{abs}(t) = |3.7 - 3.6875| = 0.0125$$
$$E_{rel}(F) = \frac{0.05}{1.3} = 0.0384615 = 3.84615\%$$
$$E_{rel}(t) = \frac{0.0125}{3.7} = 0.0033783 = 0.33783\%$$

b) Berechnen Sie die gesamte gepumpte Flüssigkeitsmenge V binär mit den gerundeten Fixkomma-Werten.
Berechnen Sie den absoluten und den relativen Fehler von V als Fixkomma-Wert (jedoch mit 8 Nachkommastellen um den gewünschten Effekt nicht zu verfälschen) im Vergleich zu den korrekten Werten F und t. Was stellen Sie fest?

$$V_{gerundet} = F \cdot t = 0001.0100 \cdot 0011.1011 = 100.10011100 = 4.609375$$
$$V_{korrekt} = 1.3 \cdot 3.7 = 4.81$$
$$E_{abs}(V) = |4.81 - 4.609375| = 0.200625$$
$$E_{rel}(F) = \frac{0.200625}{4.81} = 0.0417099 = 4.17099\%$$

Feststellung: Der relative Fehler addiert sich wg. der Multiplikation.

c) Angenommen, das System soll eine hochpräzise Medikamentendosierung steuern. Welche Konsequenzen könnte der Rundungsfehler haben?
Wie könnte man die Fehlerfortpflanzung reduzieren? Würde eine Gleitkommazahl das Problem lösen? Könnte man das Problem algorithmisch lösen?

Durch die Fixkommadarstellung entstehen Rundungsfehler, die sich bei Berechnungen verstärken.
Um die Fehler zu reduzieren, gibt es verschiedene Ansätze:

- Höhere Fixkomma-Genauigkeit (d.h. mehr Nachkommastellen). Das erhöht die Präzision aber auch den Speicherbedarf.
- Gleitkommazahlen statt Fixkommazahlen verwenden. IEEE 754 Fliesskommazahlen (z. B. float, double) bieten höhere Genauigkeit. Sie verursachen jedoch höhere Rechenkosten auf Mikrocontrollern.
- Fehlerkompensation im Algorithmus: Eine Möglichkeit wäre, die Rundungsfehler über mehrere Messungen auszugleichen.
- Lösung der Multiplikationsfehler durch alternative Algorithmen:
  Da sich relative Fehler bei Multiplikationen addieren, könnten alternative Berechnungsverfahren helfen. Beispiel: Statt $V = F \cdot t$ direkt zu berechnen, könnte man eine iterative Summierung nutzen: $V = F + F + \cdots + F(t - mal)$. Falls t eine ganze Zahl ist, vermeidet man dadurch eine Multiplikation mit Rundungsfehlern.

Aufgabe 9 Gleitkommazahlen: Float-Addition

Ein Milliardär hat ein spezielles Bankkonto, das sein Geld in Milliardeneinheiten abspeichert, d.h. 1 Milliarde CHF = 1. Dazu nutzt die Bank den Typ 32-Bit-Float. Der Milliardär besitzt aktuell 5.1 Milliarden CHF. Er verkauft nun seine Firma für 2.8 Milliarden CHF, die auf sein Bankkonto einbezahlt werden. Wir wollen diese Operation nun untersuchen.

a) Stellen Sie 2.8 und 5.1 als 32-Bit Float dar. Formen Sie dazu zunächst 2.8 und 5.1 in Fixkommazahlen um. Gehen Sie dabei wie bei den Ganzzahlen vor, verwenden Sie zusätzlich aber auch Zweierpotenzen mit negativen Exponenten. Bestimmen Sie dann den Exponenten so, dass genau eine 1 vor dem Komma steht. Diese 1 ist dann das hidden Bit.

$2 = 10_b$
$2.8 - 2 = 0.8$
$0.8 - 2^{-1} = 0.8 - 0.5 = 0.3$
$0.3 - 2^{-2} = 0.3 - 0.25 = 0.05$
$0.05 - 2^{-5} = 0.05 - 0.03125 = 0.01875$
$0.01875 - 2^{-6} = 0.01875 - 0.015625 = 0.003125$
...
$z_{2.8} = 10.\overline{1100}_b = 1.\overline{0110}_b \cdot 2^1 = 1.\overline{0110}_b \cdot 2^{128-127}$
= 0|1000 0000|0110 0110 0110 0110 0110 011
$z_{5.1} = 101.0\overline{0011}_b = 1.010\overline{0011}_b \cdot 2^2 = 1.010\overline{0011}_b \cdot 2^{129-127}$
= 0|1000 0001|0100 0110 0110 0110 0110 011

b) Bestimmen Sie nun andersherum den Wert, den die Zahlen tatsächlich als float haben. Nutzen Sie die 4er-Periodizität der Zahlen, d.h. Sie können ein Zwischenergebnis immer wieder durch 24 dividieren. Ein Taschenrechner ist zu empfehlen. Bestimmen Sie jeweils die Differenz zum ursprünglichen Wert in Milliarden CHF und einzelnen CHF.

$W(z_{2.8}) = 2 + 2{-}1 + 2{-}2 + 2{-}5 + 2{-}6 + \cdot \cdot \cdot$
$= 2 + 0.75 + 0.75/16 + \cdot \cdot \cdot$
= 2.799 999 952 316 284 179 687 5
$2.8 - W(z_{2.8}) = 0.000\,000\,047\,683\,715\,820\,312\,5 \approx 47.68 CHF$
$W(z_{5.1}) = 5.099\,999\,904\,632\,568\,359\,375$
$5.1 - W(z_{5.1}) = 0.000\,000\,095\,367\,431\,640\,625 \approx 95.37 CHF$

c) Berechnen Sie schliesslich die Summe und die Abweichung zur erwarteten Summe. Betrachten Sie ausserdem die Summe der Fehler der beiden Summanden.

$z_{5.1} + z_{2.8}$ = 0|1000 0001|(1)0100 0110 0110 0110 0110 011
+ 0|1000 0001|(0)1011 0011 0011 0011 0011 001(1) um 1 nach rechts geschoben
= 0|1000 0001|(1)1111 1001 1001 1001 1001 100(1)
= 0|1000 0001|(1)1111 1001 1001 1001 1001 100(0) gerundet (round-to-even)
= 0|1000 0001|1111 1001 1001 1001 1001 100

$W(z_{5.1} + z_{2.8}) = 7.899\,999\,618\,530\,273\,437\,5$
$7.9 - W(z_{5.1} + z_{2.8}) = 0.000\,000\,381\,469\,726\,562\,5 \approx 381.47 CHF$
$5.1 - W(z_{5.1}) + 2.8 - W(z_{2.8}) = 7.9 - W(z_{5.1}) - W(z_{2.8})$
$\approx 47.68 CHF + 95.37 CHF = 143.05 CHF \neq 381.47 CHF$

Aufgabe 10 Float-Addition implementieren

In dieser Aufgabe implementieren Sie die Addition für Gleitkommazahlen. Die folgenden Schritte führen Sie zum Resultat in Java (nachfolgend finden Sie auch einige Hinweise für die Umsetzung in C# oder Python).

Die Bit-Darstellung der Gleitkommazahl wird in dieser Aufgabe als Integer gespeichert. Sie können in Java einen `float f` mit `Float.floatToIntBits(float value)`[1] als `int` reinterpretieren – d.h. Sie erhalten so die Bit-Darstellung des `float`. Das Gegenstück dazu lautet `Float.intBitsToFloat`.

Wenn Sie die Umsetzung mit C# durchführen, können Sie das mit folgendem Code erreichen:

```
BitConverter.SingleToInt32Bits();
BitConverter.Int32BitsToSingle();
```

a) Rahmen

1. Schreiben Sie eine Klasse `FloatingPointAddition`, die die statischen Methoden aus den folgenden Aufgabenschritten enthält.
2. Schreiben Sie eine statische Funktion `extractBits`, die von einem Startwert `start` aus eine Anzahl Bits `numberOfBits` extrahiert. `start` entspricht dabei dem höchstwertigen Bit, das extrahiert werden soll.
3. Schreiben Sie eine Funktion `shiftMantissa`, die eine Mantisse um eine Anzahl Bits verschiebt. Wenn die Anzahl Bits grösser als Null ist, soll die Mantisse nach rechts und sonst nach links geschoben werden.

b) Vorbereitende Hilfsfunktionen

1. Schreiben Sie eine Funktion `prependLeadingOneForMantissa`, die die vorstehende 1 an die Mantisse anhängt. Neben den Kommastellen der Mantisse benötigen Sie auch den dazugehörigen Exponenten als Parameter. Berücksichtigen Sie dabei auch die 0.
2. Schreiben Sie eine Funktion `getSignFromBit`, die aus dem Vorzeichenbit in +1 bzw. -1 ableitet.

c) Nachbereitende Hilfsfunktionen

1. Schreiben Sie eine Funktion `getNumberOfBitShiftsForMantissaNormalization`, die berechnet, um wie viele Bits die Mantisse verschoben werden muss, damit sie normalisiert wird. Als Hilfe können Sie die Funktionen `Integer.highestOneBit(int mantissa)` und `Integer.numberOfTrailingZeros(int highestOneBit)` verwenden. Behandeln Sie auch den Fall, wenn die Mantisse den Wert 0 hat.
2. Schreiben Sie eine Funktion `normalizeMantissa`, die die Mantisse so verschiebt, dass man sie wieder im Float einsetzen kann. Behandeln Sie auch den Fall, dass die Mantisse negativ ist.
3. Schreiben Sie eine Funktion `adjustExponent`, die anhand der Mantisse den Exponenten so anpasst, dass er in den Float eingesetzt werden kann. Behandeln Sie den Fall, dass die Mantisse negativ ist und den Fall, dass der Exponent 0 ist.
4. Schreiben Sie eine Funktion `removeLeadingOneFromMantissa`, um die vorstehende 1 der Mantisse wieder zu entfernen
5. Schreiben Sie eine Funktion `assembleBitsToFloat`, die drei `int` Parameter `sign`, `exponent` und `mantissa` entgegennimmt und diese zu den Integer-Bits des entsprechenden `floats` zusammensetzt. Entfernen Sie dabei die vorstehende 1 der Mantisse.

[1] https://docs.oracle.com/javase/8/docs/api/java/lang/Float.html#floatToIntBits-float-

d) Addition
Führen Sie mithilfe der obengenannten Funktionen die Addition durch. Sie können folgendermassen vorgehen:
1. 1. Schreiben Sie die statische Funktion `addition`, die zwei `int` entgegennimmt und einen `int` zurückgibt.
2. 2. Extrahieren Sie zu Beginn das Vorzeichen, den Exponenten und die Mantisse aus den Inputparametern.
3. 3. Passen Sie die kleinere Mantisse der grösseren an, wenn der Exponent unterschiedlich ist. Für den Fall, dass der Exponent den Wert 0 hat, müssen Sie ihn wie den Wert 1 behandeln.
4. 4. Führen Sie die Addition der Mantissen durch und beachten Sie dabei das Vorzeichen.
5. 5. Die resultierende Mantisse müssen Sie nun normalisieren. Falls beide Exponenten der Inputwerte den Wert Null haben, passen Sie die Mantisse nicht an.
6. Nehmen Sie den Exponenten der grösseren Mantisse und passen Sie diesen entsprechend der resultierenden Mantisse (vor der Normalisierung) an. Falls einer der Input Exponenten den Wert Null hat, verwenden Sie den anderen.
7. Berechnen Sie das Vorzeichen des Resultats
8. 8. Setzen Sie die erhaltenen Werte wieder zu den Integer Bits des `floats` zusammen.

Lösung siehe «Aufgabe 9 – Implementierungen» auf Moodle.

Aufgabe 11 Codierung von Text

Lesen Sie das Essay *«Von ASCII zu Unicode»* (Moodle) und verschaffen Sie sich einen Überblick über das Unicode-Konzept sowie dessen Umsetzung in UTF-8.

Führen Sie anschliessend die folgenden Codierungen durch.

a) Codieren Sie «Café €» nach UTF-8
C = U+0043 = (ASCII) 0100 0011
a = U+0061 = (ASCII) 0110 0001
f = U+0066 = (ASCII) 0110 0110
é = U+00E9 = (2 Byte) 1100 0011 1010 1001
Space = U+0020 = (ASCII) 0010 0000
€ = U20AC = (3 Byte) 1110 0010 10000 0010 1010 1100

01000011 01100001 01100110 11000011 10101001 00100000 11100010 10000010 10101100

b) Decodieren Sie folgenden UTF-8-Code wieder nach Text
01001000 01101001 00100000 11110000 10011111 10011000 10001010 00100001

01001000 (1 Byte)
Dies ist ein einzelnes Byte, beginnt mit 0, was darauf hinweist, dass es sich um ein ASCII-Zeichen handelt.
Die direkte Umwandlung ergibt das Zeichen "H".

01101001 (1 Byte)
Auch dieses Byte beginnt mit 0, was ein ASCII-Zeichen anzeigt.
Die Umwandlung ergibt das Zeichen "i".

00100000 (1 Byte)
Ein weiteres Byte, das mit 0 beginnt, steht für ein ASCII-Zeichen, in diesem Fall das Leerzeichen " ".

11110000 10011111 10011000 10001010 (4 Bytes)
Diese Byte-Sequenz beginnt mit 11110, was auf ein UTF-8-kodiertes Zeichen hinweist, das vier Bytes umfasst.
Die folgenden Bytes beginnen jeweils mit 10, was bestätigt, dass sie zur Kodierung desselben Zeichens gehören.
Um den Unicode-Codepunkt zu finden, entfernen wir die für UTF-8 spezifischen Bits:

i. Entferne die ersten fünf Bits des ersten Bytes (11110) und die ersten beiden Bits (10) von jedem der folgenden drei Bytes.
ii. Die verbleibenden Bits sind 0001 1111 1001 1000 001010, zusammengefügt zu 00001111100110000001010, was dem Hexadezimalwert 1F60A entspricht.

U+1F60A ist der Unicode-Codepunkt für das Emoji "😊".

00100001 (1 Byte)
Ein einzelnes Byte, das mit 0 beginnt, steht für das ASCII-Zeichen "!".

Übung 3

Algebra der Bits - von Logik zu Polynomen

Lösung

V2.0

Hinweise:

- Übungen sind mit Vorteil alleine zu lösen
- Benutzen Sie die Musterlösung nur zur Korrektur
- Die Übungen sind wichtige Vorbereitungen für die Prüfung. Lösen Sie die Übungen sorgfältig und stellen Sie die Lösungswege übersichtlich dar.
- (Ergänzte) Vorlesungsunterlagen und Fachbücher helfen beim Lösen von Übungen und bringen gleichzeitig eine erweiterte Ansicht auf die Problemstellung.
- Wenn Sie die Übungen nicht verstehen, fragen Sie!

Aufgabe 1 Wahrheitstabellen

Evaluieren Sie folgende logische Funktionen mittels Wahrheitstabellen.

a)

x	y	$(x \rightarrow y) \oplus (y \rightarrow x)$
0	0	0
0	1	1
1	0	1
1	1	0

Zu welcher Binärfunktion lässt sich der Ausdruck vereinfachen?

Die Funktion lässt sich zu XOR vereinfachen, wie man der Wahrheitstabelle direkt entnehmen kann.

b) Beweisen Sie das erste De Morgan Law $\overline{x \vee y} = \overline{x} \wedge \overline{y}$ mit Hilfe von zwei Wahrheitstabellen.

x	y	$\overline{x \vee y}$	$\overline{x} \wedge \overline{y}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

Beide Wahrheitstabellen sind identisch, d.h. die Seite links vom Gleichheitszeichen kann immer für die rechte Seite (oder umgekehrt) in anderen Formeln ersetzt werden, ohne das sich irgendein Wert der Formel ändert.

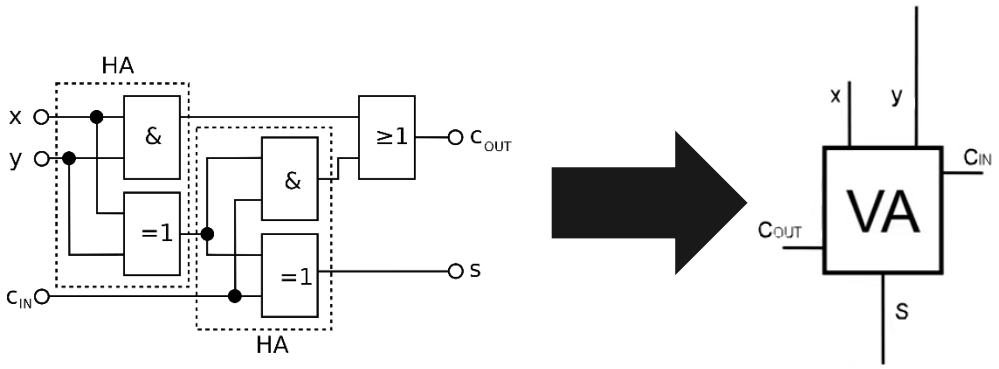
c) Zeichen Sie mit Hilfe von zwei Wahrheitstabellen, dass $x \mid (y \mid y) \neq (x \mid y) \mid y$ ist.

x	y	x | (y | y)	(x | y) | y
0	0	1	1
0	1	1	0
1	0	0	1
1	1	1	1

Aufgabe 2 Paralleladdierer

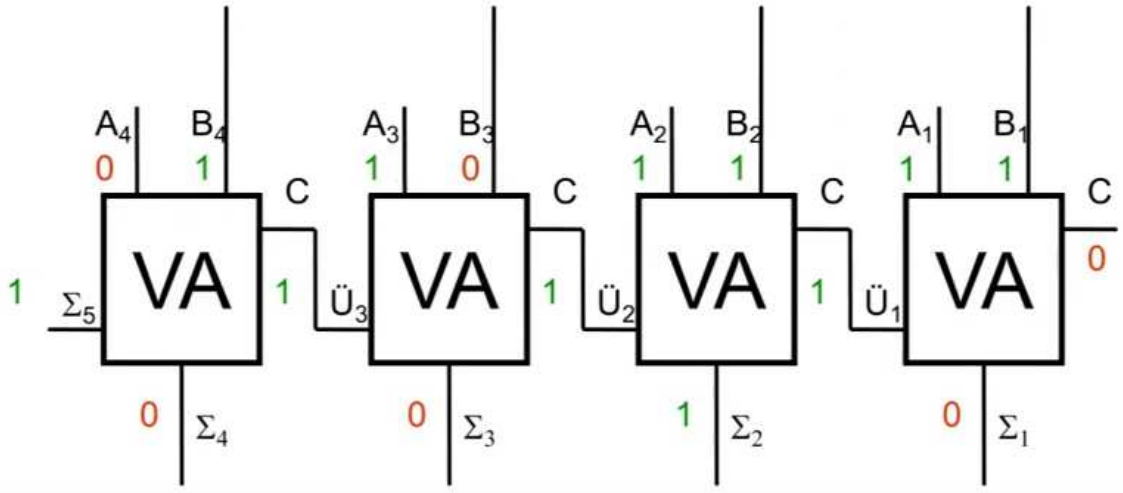
Sie haben in der Vorlesung den Halbaddierer und den Volladdierer kennengelernt. Kombinieren Sie den Volladierer nun so, damit Sie zwei 4-Bit-Zahlen addieren können (=Paralleladdierer).

Um die Übersicht zu behalten, vereinfachen Sie den Volladdierer



a) Führen Sie anschliessend mit Ihrem Volladdierer die Berechnung 0111 + 1011 durch und überprüfen Sie das Resultat.

Lösung:



b) Überlegen Sie sich einen allfälligen Nachteil des Paralleladdierers (Stichwort: Carry-Bit). Informieren Sie sich über den Carry-Lookahead-Addierer.

Ein spezifischer Nachteil von Paralleladdierern, der oft diskutiert wird, betrifft die Handhabung des Übertrags (Carry). In einem Paralleladdierer muss der Übertrag von jeder Bitstelle zur nächsten weitergeleitet werden. Dies kann, insbesondere bei Addierern, die einfach die Überträge von einer Stelle zur nächsten weiterreichen (sogenannte Ripple-Carry-Addierer), zu signifikanten Verzögerungen führen. Hier sind die Hauptprobleme im Zusammenhang mit dem Übertrag (Carry) in Paralleladdierern:

1. **Geschwindigkeitsbegrenzung**: In Ripple-Carry-Addierern muss der Übertrag durch jede Bitposition hindurchgeleitet werden, beginnend bei der niedrigsten bis hin zur höchsten Bitstelle. Das bedeutet, dass die Zeit, die benötigt wird, um eine Addition durchzuführen, direkt proportional zur Anzahl der Bits ist. Dies begrenzt die Geschwindigkeit des Addierers erheblich, besonders bei Operationen mit hoher Bitbreite.
2. **Verzögerung durch Carry-Propagation**: Die Fortpflanzungszeit des Übertrags von der niedrigsten zur höchsten Bitstelle ist der limitierende Faktor für die Geschwindigkeit des gesamten Addiervorgangs. Diese Verzögerung wird als Carry-Propagation-Delay bezeichnet und kann die Leistung in zeitkritischen Anwendungen beeinträchtigen.

Um diese Probleme zu mindern, wurden verschiedene Arten von Paralleladdierern entwickelt, die eine schnellere Carry-Verarbeitung ermöglichen, wie z.B.: Übertragsvorausberechnung (carry-lookahead)

Aufgabe 3 Vereinfachung

Eine Select-Komponente selektiert eines von zwei Input-Bits. Das s (select) Bit zeigt, welches Input-Bit selektiert werden soll. Wenn 0, wird d0 selektiert, wenn 1 wird d1 selektiert.

Die Wahrheitstabelle für eine Select-Komponente sieht entsprechend folgendermassen aus:

s	d1	d0	Ouput
0	0	0	0
0	1	0	0
0	0	1	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

a) Notieren Sie KDNF.

(¬s ∧ ¬d1 ∧ d0) ∨ (¬s ∧ d1 ∧ d0) ∨ (s ∧ d1 ∧ ¬d0) ∨ (s ∧ d1 ∧ d0)

b) Vereinfachen Sie die KDNF mit Hilfe eines KV-Diagramms.

	¬d1	**¬d1**	**d1**	**d1**
¬s	0	1	1	0
s	0	0	1	1
	¬d0	**d0**	**d0**	**¬d0**

Wir können nun zwei Blöcke bilden:
¬s ∧ ¬d1 ∧ d0 ∧ d1 ∧ d0 = ¬s ∧ d0
s ∧ d1 ∧ d0 ∧ s ∧ d1 ∧ ¬d0 = s ∧ d1
Resultat: (¬s ∧ d0) ∨ (s ∧ d1)

c) Überführen Sie Ihr Resultat in die Form, welche ausschliesslich NAND verwendet.
Erinnerung: NAND(A, B) = NOT(AND(A,B)) = ¬(A ∧ B) = ¬A ∨ ¬B

(s | d1) | (¬s | d0) oder das ¬ noch ersetzt: (s | d1) | ((s | s) | d0)

Aufgabe 4 NAND-Game

Spielen Sie das NAND-Game[1] bis und mit Level 2 (Bit Addierer) durch. Sie können Ihre Resultate aus Aufgabe 3 für die Select-Component wieder verwenden.

Aufgabe 5 Vereinfachung von Programmen

Der Code von Methoden mit dem Rückgabetyp *boolean* ist oft komplizierter als nötig. Schauen Sie sich beispielsweise die folgende Java Methode an (äquivalent in C#):

```
public boolean hasWallsOnBothSides () {
    if (isWallToLeft()) {
        if (isWallToRight()) {
            return true;
        } else {
            return false;
        }
    } else {
        return false;
    }
}
```

In obigem Code kann man z.B. das innere `if-else`-Statement durch `return isWallToRight();` ersetzen.

a) Vereinfachen Sie das obige Beispiel weiter.

Lösung:

```
public boolean hasWallsOnBothSides () {
    return isWallToLeft() && isWallToRight();
}
```

b) Vereinfachen Sie:

```
if (boolExp1) {
    return false;
} else {
    return true;
}
```

```
return !boolExp1;
```

[1] https://nandgame.com/

c) Vereinfachen Sie:

```
if (boolExp1) {
    return true;
} else {
    return boolExp3;
}
```

```
return boolExp1 || boolExp3;
```

d) Vereinfachen Sie:

```
if (boolExp1) {
    return boolExp2;
} else {
    return false;
}
```

```
return boolExp1 && boolExp2;
```

e) Weiterhin können oft auch logische Ausdrücke weiter vereinfacht werden. Z. B. kann `boolExpr == true` durch `boolExpr` oder `boolExpr == false` durch `!boolExpr` ersetzt werden. Vor allem DeMorgan's Laws sind sehr hilfreich um logische Ausdrücke, die aus &&, || und ! aufgebaut sind zu vereinfachen. Wenden Sie letzteres auf folgende Beispiele an:

1) `(!boolExp1) || (!boolExp2)`

   `!(boolExp1 && boolExp2)`

2) `(!boolExp1) && (!boolExp2)`

   `!(boolExp1 || boolExp2)`

3) `boolExp1 || ((!boolExp1) && boolExp2)`

   `boolExp1 || boolExp2`

4) `((!boolExp1) && boolExp2) || boolExp1`

   `boolExp1 || boolExp2`

5) `boolExp1 && ((!boolExp1) || boolExp2)`

   `boolExp1 && boolExp2`

f) Erstellen Sie die Wahrheitstabelle von folgendem Ausdruck, leiten Sie die KDNF ab, vereinfachen Sie von dort und geben Sie einen möglichst kompakten Ausdruck an.

```
1 !(!(x && y) && z)
```

x	y	z	Ausdruck	Minterm
0	0	0	1	¬x ∧ ¬y ∧ ¬z
0	0	1	0	
0	1	0	1	¬x ∧ y ∧ ¬z
0	1	1	0	
1	0	0	1	x ∧ ¬y ∧ ¬z
1	0	1	0	
1	1	0	1	x ∧ y ∧ ¬z
1	1	1	1	x ∧ y ∧ z

(¬x ∧ ¬y ∧ ¬z) ∨ (¬x ∧ y ∧ ¬z) ∨ (x ∧ ¬y ∧ ¬z) ∨ (x ∧ y ∧ ¬z) ∨ (x ∧ y ∧ z)

= (¬x ∧ (¬y ∨ y) ∧¬z) ∨ (x ∧ (¬y ∨ y) ∧¬z) ∨ (x ∧ y ∧ z)

= (¬x ∧ ¬z) ∨ (x ∧¬z) ∨ (x ∧ y ∧ z)

= ((¬x ∨ x)∧ ¬z) ∨ (x ∧ y ∧ z)

= ¬z ∨ (x ∧ y ∧ z)

Damit können wir also schreiben : `(x && y && z) || !z`
Das ist für manche Menschen leichter verständlich. Die Ausdrücke sind aber nur dann äquivalent, wenn es keine Seiteneffekte bei der Auswertung von `z` gibt.

Der Ausdruck könnte noch weiter umgeformt werden zu (¬z ∨ x) ∧ (¬z ∨ y) wobei nun Ermessenspielraum ist, ob dieser Ausdruck einfacher ist, als der obere.

Aufgabe 6 Modulorechnung

Berechnen Sie x.

a) 17 mod 7 = x x = 3

b) -17 mod -7 = x x = -3

c) 17 mod x = 7 x = 10

d) 17 mod x = -7 x = -12 oder x = -24

e) 17 mod x = 17 x > 17

f) x mod 17 = 7 x= … -10, 7, 24, 41, …

Aufgabe 7 Vektoraddition in $\mathbb{Z}_2$

Berechnen Sie

$$\begin{pmatrix}1\\1\\0\end{pmatrix}+\begin{pmatrix}0\\0\\1\end{pmatrix}+\begin{pmatrix}1\\1\\1\end{pmatrix}+\begin{pmatrix}1\\0\\1\end{pmatrix}+\begin{pmatrix}0\\0\\1\end{pmatrix}$$

$$\begin{pmatrix}1\\0\\0\end{pmatrix}$$

Aufgabe 8 Polynomrechnungen in $\mathbb{Z}_2$

a) Gegeben sind folgende zwei Polynome
A(x)=u^4+u^2+u+1
B(x)=$u^3+u^1+u^0$

1) Addieren Sie die zwei Polynome A(x) und B(x). Interpretieren Sie das Resultat als Codewort.

$(u^4+u^2+u+1) + (u^3+u^1+u^0) = (u^4+u^2+u^1+u^0) + (u^3+u^1+u^0) = u^4+u^3+u^2+2u^1+2u^0 = u^4+u^3+u^2$
Codewort: 11100

2) Multiplizieren Sie die zwei Polynome A(x) und B(x)

$(u^4+u^2+u+1) * (u^3+u^1+u^0) = (u^4+u^2+u^1+u^0) * (u^3+u^1+u^0) =$
$u^7+u^5+u^4+u^5+u^3+u^2+u^4+u^2+u^1+u^3+u^1+u^0=u^7+2u^5+2u^4+2u^3+2u^2+2u^1+u^0=u^7+u^0$
Codewort: 10000001

3) Notieren Sie die zwei Polynome als Vektoren und führen Sie die Addition nochmals in dieser Darstellung durch.

$$\begin{pmatrix}1\\0\\1\\1\\1\end{pmatrix}+\begin{pmatrix}0\\1\\0\\1\\1\end{pmatrix}=\begin{pmatrix}1\\1\\1\\0\\0\end{pmatrix}$$

b) Berechnen Sie $\frac{x^6}{x^3+x+1}$

$$\begin{array}{rrrrrr} x^6 & & & & & : x^3+x+1 = x^3+x+1 \\ -\ x^6 & +x^4 & +x^3 & & & \\ \hline & x^4 & +x^3 & & & \\ -\ & x^4 & & +x^2 & +x & \\ \hline & & x^3 & +x^2 & +x & \\ & -\ x^3 & & & +x & +1 \\ \hline & & & x^2 & & +1 \end{array}$$

$$\frac{x^6}{x^3+x+1} = (x^3+x+1)\ Rest\ (x^2+1)$$

c) Als Resultat einer Polynomdivision haben Sie (x^4+1) Rest (x+1) erhalten. Was war vermutlich die ursprüngliche Division?

$$(x^4+1)+\frac{x+1}{x^4+1}=\frac{(x^4+1)^2+x+1}{x^4}=\frac{x^8+x}{x^4+1}$$

Übung 4

Wahrscheinlichkeit als Modell von Unsicherheit

Lösung

V2.0

Hinweise:

- Übungen sind mit Vorteil alleine zu lösen
- Benutzen Sie die Musterlösung nur zur Korrektur
- Die Übungen sind wichtige Vorbereitungen für die Prüfung. Lösen Sie die Übungen sorgfältig und stellen Sie die Lösungswege übersichtlich dar.
- (Ergänzte) Vorlesungsunterlagen und Fachbücher helfen beim Lösen von Übungen und bringen gleichzeitig eine erweiterte Ansicht auf die Problemstellung.
- Wenn Sie die Übungen nicht verstehen, fragen Sie!

Aufgabe 1 Wahrscheinlichkeit: Experiment

Bei der wiederholten Durchführung eines Experimentes treten die Ergebnisse A1 bis A4 (exklusiv) mit den unten angegebenen Häufigkeiten auf.

Ereignis	A1	A2	A3	A4
Anzahl	5550	3567	2234	4443

1. Bestimmen Sie näherungsweise die Auftrittswahrscheinlichkeiten der Ergebnisse.

Anzahl aller Ereignisse: 15797

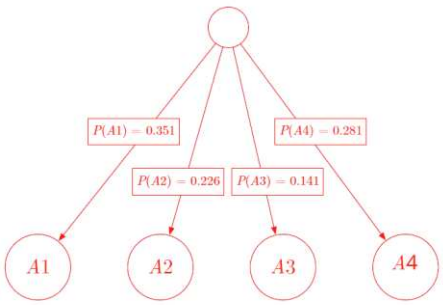
$$P(A1) = \frac{5550}{15797} = 0.351$$

$$P(A2) = \frac{3567}{15797} = 0.226$$

$$P(A3) = \frac{2234}{15797} = 0.141$$

$$P(A4) = \frac{4443}{15797} = 0.281$$

Ereignisbaum:



2. Wie gross ist die Wahrscheinlichkeit, dass bei …
   a. … einem Experiment das Ergebnis A2 oder A4 auftritt?

$$P(A2 \vee A4) = P(A2) + P(A4) = 0.226 + 0.281 = 0.507$$

   b. Bei zwei aufeinander folgenden Experimenten zuerst das Ergebnis A1 und dann A3 auftritt?

$$P(A1 \text{ dann } A3) = P(A1) \times P(A3) = 0.351 \times 0.141 = 0.049$$

Aufgabe 2 Wahrscheinlichkeit: Lotto

Berechnen Sie beim Lotto 6 aus 49 die Wahrscheinlichkeiten 3, 4, 5 oder 6 Richtige zu erhalten.

Als Grundlage für diese Berechnung wird die Wahrscheinlichkeitsdefinition von Laplace benötigt:

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

Anzahl aller Ergebnisse:

$$\binom{49}{6} = \frac{49!}{6! \cdot (49-6)!}$$

Anzahl günstige Ergebnisse unter der Verwendung von zwei Mengen (gezogene und nicht gezogene Zahlen):

$$\binom{\text{Gezogene}}{\text{Treffer}} \cdot \binom{\text{Nicht-Gezogene}}{\text{Falsche}}$$

Bemerkung: Man nennt diese Verteilung eine hypergeometrische Verteilung. Sie beschreibt das Auftreten eines qualitativen Merkmals einer Stichprobe ohne Zurücklegen.

$$P(3 \text{ Richtige}) = \frac{\binom{6}{3} \cdot \binom{43}{3}}{\binom{49}{6}} = 0.01765040$$

$$P(4 \text{ Richtige}) = \frac{\binom{6}{4} \cdot \binom{43}{2}}{\binom{49}{6}} = 0.00096862$$

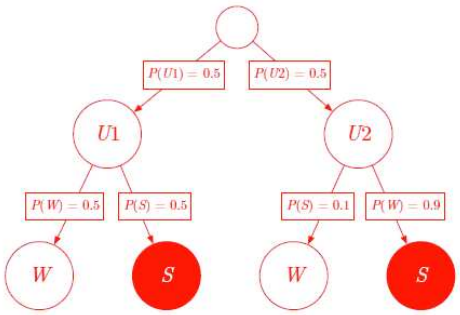
$$P(5 \text{ Richtige}) = \frac{\binom{6}{5} \cdot \binom{43}{1}}{\binom{49}{6}} = 0.00001845$$

$$P(6 \text{ Richtige}) = \frac{\binom{6}{6} \cdot \binom{43}{0}}{\binom{49}{6}} = 0.00000007$$

Aufgabe 3 Wahrscheinlichkeit: Urnen

Es seien zwei Urnen vorhanden. In der ersten Urne sind 5 schwarze und 5 weisse Kugeln. In der zweiten Urne seien 9 schwarze und 1 weisse Kugel. Wie gross ist die Wahrscheinlichkeit, eine weisse Kugel zu ziehen, wenn die Auswahl der Urne für beide Urnen gleich wahrscheinlich ist?

Ereignisbaum:



Aus dem Ereignisbaum kann die Formel für die Lösung abgeleitet werden:

$$P(U1) \wedge P(W|U1) \vee P(U2) \wedge P(W|U2) = 0.5 \cdot 0.5 + 0.5 \cdot 0.1 = 0.3$$

Bemerkung: Der $\wedge$-Operator bindet stärker als der $\vee$-Operator.

Aufgabe 4 Wahrscheinlichkeit: Bitfehler 1

Die Wahrscheinlichkeit eines Bitfehlers sei mit 10^{-3} gegeben.

1. Wie gross ist die Wahrscheinlichkeit, dass ein Datenblock der Grösse von 150 kbit[1] fehlerhaft ist?

$$\begin{aligned} P(\text{fehlerhaft}) &= 1 - P(0\ \text{Fehler}) \\ &= 1 - 0.999^{150000} \\ &= 1 - 6.66 \cdot 10^{-66} \\ &= 1 \end{aligned}$$

2. Sie haben ein Verfahren zur Vorwärtskorrektur entwickelt und können bis zu 3 Fehler richtig korrigieren. Die Grösse eines Datenblocks sei 150 bit. Wie gross ist die Wahrscheinlichkeit, dass höchstens 3 Fehler auftreten. Wie gross ist die Restfehlerwahrscheinlichkeit für einen Datenblock (Verwenden Sie eine Bitfehler-Wahrscheinlichkeit von 10^{-2})?

[1] Nach SI: 1 Kilobit = 10^3 Bit = 1000 Bit

$$
\begin{aligned}
P(\leq 3 \text{ Fehler}) &= P(0 \text{ Fehler}) + P(1 \text{ Fehler}) + P(2 \text{ Fehler}) + P(3 \text{ Fehler}) \\
&= \binom{150}{0} \cdot 0.01^0 \cdot 0.99^{150} + \binom{150}{1} \cdot 0.01^1 \cdot 0.99^{149} \\
&\quad + \binom{150}{2} \cdot 0.01^2 \cdot 0.99^{148} + \binom{150}{3} \cdot 0.01^3 \cdot 0.99^{147} \\
&= 0.221 + 0.336 + 0.252 + 0.126 \\
&= 0.935
\end{aligned}
$$

Die Restfehlerwahrscheinlichkeit beträgt folglich:

$$P(\text{Restfehler}) = 1 - P(\leq 3 \text{ Fehler}) = 1 - 0.935 = 0.065$$

Diese Art der Verteilung nennt man Binomialverteilung. Sie beschreibt das k-fache Auftreten eines qualitativen Merkmals mit der Wahrscheinlichkeit p in einer Stichprobe der Grösse n mit Zurücklegen:

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

Aufgabe 5 Wahrscheinlichkeit: Würfel

Man bestimme die Wahrscheinlichkeit dafür, dass bei zwei Würfen eines unverfälschten Würfels wenigstens einmal die 4 erscheint.

Die Wahrscheinlichkeit, dass bei einem Wurf keine 4 gewürfelt wird, ist $\frac{5}{6}$. D.h. die Wahrscheinlichkeit im ersten und im zweiten Wurf keine 4 zu würfeln ist $\frac{25}{36}$ (oder $\frac{5}{6} \cdot \frac{5}{6}$). Daraus folgt die Wahrscheinlichkeit mit zwei Würfen mindestens eine 4 zu würfeln:

$$P(\text{mind. einmal 4}) = 1 - \frac{25}{36} = \frac{11}{36} = 0.306$$

Mit der Binomialverteilung und der Gegenwahrscheinlichkeit:

$$\begin{aligned} P(\text{mind. einmal 4}) &= 1 - \binom{n}{k} p^k (1-p)^{n-k} \\ &= 1 - \binom{2}{0} \left(\frac{1}{6}\right)^0 \left(\frac{5}{6}\right)^2 = 0.306 \end{aligned}$$

Mit der Binomialverteilung:

$$\begin{aligned} P(\text{mind. einmal 4}) &= P(\text{1x eine 4}) + P(\text{2x eine 4}) \\ &= \binom{2}{1} \left(\frac{1}{6}\right)^1 \left(\frac{5}{6}\right)^1 + \binom{2}{2} \left(\frac{1}{6}\right)^2 \left(\frac{5}{6}\right)^0 = 0.306 \end{aligned}$$

Aufgabe 6 Wahrscheinlichkeit: Schach

A und B spielen 12 Partien Schach, von denen A 6 und B 4 gewinnt und zwei unentschieden ausgehen. Sie vereinbaren, ein aus 3 Partien bestehendes Turnier zu spielen. Man bestimme basierend auf diesen 12 vergangenen Partien, die Wahrscheinlichkeit dafür, dass …

1. … A alle drei Partien gewinnt.

Die Wahrscheinlichkeit der Einzelereignisse wird über die empirische Häufigkeit angenähert zu:

$$P(\text{A gewinnt}) = \frac{6}{12} = \frac{1}{2}$$
$$P(\text{B gewinnt}) = \frac{4}{12} = \frac{1}{3}$$
$$P(\text{unentschieden}) = \frac{2}{12} = \frac{1}{6}$$
$$P(\text{A gewinnt alles}) = \left(\frac{1}{2}\right)^3 = 0.125$$

2. … genau zwei Partien unentschieden ausgehen.

$$\begin{aligned} P(\text{genau 2 unentschieden}) &= P(2 \text{ unentschieden}) \wedge P(\text{A gewinnt}) \\ &\quad \vee P(2 \text{ unentschieden}) \wedge P(\text{B gewinnt}) \\ &= \binom{3}{2} \cdot \left(\frac{1}{6}\right)^2 \cdot \left(\frac{1}{2}\right) + \binom{3}{2} \cdot \left(\frac{1}{6}\right)^2 \cdot \left(\frac{1}{3}\right) \\ &= 0.042 + 0.028 \\ &\approx 0.07 \end{aligned}$$

Oder direkt mit der Binomialverteilung:

$$\begin{aligned} P(\text{genau 2 unentschieden}) &= \binom{n}{k} p^k (1-p)^{n-k} \\ &= \binom{3}{2} \left(\frac{1}{6}\right)^2 \left(\frac{5}{6}\right)^1 \approx 0.07 \end{aligned}$$

3. … A und B abwechselnd gewinnen.

$$\begin{aligned} P(\text{abwechselnd}) &= P(\text{A gewinnt}) \wedge P(\text{B gewinnt}) \wedge P(\text{A gewinnt}) \\ &\quad \vee P(\text{B gewinnt}) \wedge P(\text{A gewinnt}) \wedge P(\text{B gewinnt}) \\ &= \frac{1}{2} \cdot \frac{1}{3} \cdot \frac{1}{2} + \frac{1}{3} \cdot \frac{1}{2} \cdot \frac{1}{3} \\ &= \frac{5}{36} \end{aligned}$$

4. … B mindestens eine Partie gewinnt.

Idee: P(B gewinnt mind. eine Partie) = 1 - P(B gewinnt keine Partie)

$$\begin{aligned} P(\text{B gewinnt keine Partie}) &= P(\text{3x unentschieden}) \\ &\quad \vee P(\text{2x unentschieden}) \wedge P(\text{A gewinnt}) \\ &\quad \vee P(\text{1x unentschieden}) \wedge P(\text{A gewinnt 2x}) \\ &\quad \vee P(\text{A gewinnt 3x}) \\ &= \left(\frac{1}{6}\right)^3 + 3 \cdot \left(\left(\frac{1}{6}\right)^2 \cdot \frac{1}{2}\right) + 3 \cdot \left(\frac{1}{6} \cdot \left(\frac{1}{2}\right)^2\right) + \left(\frac{1}{2}\right)^3 \\ &= \frac{8}{27} = 0.296 \end{aligned}$$

D.h.: P(B gewinnt mind. eine Partie) = 1 - 0.296 = 0.704

Die Multiplikation mit 3 bei P(2x unentschieden und A gewinnt) und P(1x unentschieden und A gewinnt 2x) kommt von der Anzahl der möglichen Anordnungen dieser Ergebnisse. Es gibt jeweils drei verschiedene Möglichkeiten, wie diese Ergebnisse auftreten können und jede hat dieselbe Wahrscheinlichkeit. Deshalb multiplizieren wir die Wahrscheinlichkeit einer spezifischen Reihenfolge dieser Ergebnisse mit 3, um die Gesamtwahrscheinlichkeit für dieses Szenario zu erhalten.

Aufgabe 7 Wahrscheinlichkeit: Anzahl Kinder

Man bestimme die Wahrscheinlichkeit von Knaben und Mädchen in Familien mit drei Kindern, wenn die Wahrscheinlichkeit für Knaben und Mädchen als gleich angenommen wird.

# Knaben	# Mädchen	Berechnung	Wert
3	0	$\binom{3}{0} \times 0.5^3$	0.125
2	1	$\binom{3}{1} \times 0.5^3$	0.375
1	2	$\binom{3}{2} \times 0.5^3$	0.375
0	3	$\binom{3}{3} \times 0.5^3$	0.125

Aufgabe 8 Kombinatorik: Anordnung Murmeln

Auf welche Arten können 5 verschieden farbige Murmeln in einer Reihe angeordnet werden (Permutation)?

Anzahl aller möglichen Permutationen = 5! = 120

Aufgabe 9 Kombinatorik: Sitzplatz-Anordnung

5 Männer und 4 Frauen sollen in einer Reihe sitzen, und zwar so, dass die Frauen auf den geraden Plätzen sitzen. Wie viele solche Anordnungen sind möglich?

Anzahl möglicher Anordnungen = Permutationen Frauen × Permutationen Männer
= 4! × 5! = 24 × 120 = 2880

Aufgabe 10 Wahrscheinlichkeit: Ethernet-Kollision

An einem Ethernet-Strang hängen 17 Maschinen, mit jeweils einer Wahrscheinlichkeit für einen Zugriff von p = 0.1. Wie gross ist die Wahrscheinlichkeit, dass es zu einer Kollision kommt?

$$P(X \leq 1) = 1 - P(X = 0) - P(X = 1)$$
$$P(X \leq 1) = 1 - \binom{n}{0} p^0 (1-p)^{n-0} - \binom{n}{1} p^1 (1-p)^{n-1}$$
$$P(X \leq 1) = 1 - \binom{17}{0} p^0 (1-0.1)^{17-0} - \binom{17}{1} (0.1)^1 (1-0.1)^{17-1}$$
$$P(X \leq 1) = 1 - (1-0.1)^{17} - 17 \cdot 0.1 (1-0.1)^1 6$$
$$P(X \leq 1) = 1 - 0.167 - 0.315 = 0.518$$

Aufgabe 11 Wahrscheinlichkeit: Bitfehler 2

Ein Übertragungskanal überträgt Bits (0 oder 1). Die Bitfehlerwahrscheinlichkeit beträgt $p = 0.02$
Das bedeutet: Ein gesendetes Bit wird mit Wahrscheinlichkeit 0.02 falsch übertragen.
Die gesendeten Bits sind gleich wahrscheinlich:

$$P(gesendet = 0) = 0.5$$
$$P(gesendet = 1) = 0.5$$

Der Empfänger liest ein Bit.

1. Gesamtwahrscheinlichkeit: Wie gross ist die Wahrscheinlichkeit, dass der Empfänger eine 1 liest?
   Gesucht: $P(empfangen = 1)$

   Eine empfangene 1 kann auf zwei Arten entstehen:
   Eine 1 wurde gesendet und korrekt übertragen

   $$P(empfangen = 1 \mid gesendet = 1) = 0.98$$

   Eine 0 wurde gesendet und durch einen Fehler zu 1

   $$P(empfangen = 1 \mid gesendet = 0) = 0.02$$

   Nach dem Satz der totalen Wahrscheinlichkeit:

   $$P(empfangen = 1) = P(empfangen = 1 \mid gesendet = 1) \cdot P(gesendet = 1)$$
   $$+P(empfangen = 1 \mid gesendet = 0) \cdot P(gesendet = 0)$$
   $$= 0.98 \cdot 0.5 + 0.02 \cdot 0.5$$
   $$= 0.49 + 0.01$$
   $$= 0.50$$

2. Bedingte Wahrscheinlichkeit (Bayes): Der Empfänger liest eine 1. Wie gross ist die Wahrscheinlichkeit, dass tatsächlich eine 1 gesendet wurde?
   Gesucht: $P(gesendet = 1 \mid empfangen = 1)$
   Hinweis: Verwenden Sie das Bayes-Theorem.

   Gesucht: $P(gesendet = 1 \mid empfangen = 1)$
   Bayes:

   $$P(gesendet = 1 | empfangen = 1)$$
   $$= (P(empfangen = 1 \mid gesendet = 1) \cdot P(gesendet = 1)) / P(empfangen = 1)$$

   Einsetzen:

   $$= (0.98 \cdot 0.5) / 0.5$$
   $$= 0.98$$

   Wenn der Empfänger eine 1 liest, beträgt die Wahrscheinlichkeit, dass tatsächlich eine 1 gesendet wurde, etwa 98 %.
   Die restlichen 2 % entstehen durch Übertragungsfehler.

Übung 5

Information & Entropie
Wie viel Information steckt in Daten
Lösung

V1.0

Hinweise:

- Übungen sind mit Vorteil alleine zu lösen
- Benutzen Sie die Musterlösung nur zur Korrektur
- Die Übungen sind wichtige Vorbereitungen für die Prüfung. Lösen Sie die Übungen sorgfältig und stellen Sie die Lösungswege übersichtlich dar.
- (Ergänzte) Vorlesungsunterlagen und Fachbücher helfen beim Lösen von Übungen und bringen gleichzeitig eine erweiterte Ansicht auf die Problemstellung.
- Wenn Sie die Übungen nicht verstehen, fragen Sie!

OST Ostschweizer Fachhochschule

Aufgabe 1 Diskrete Quelle ohne Gedächtnis

1) Sie haben eine Quelle mit folgenden Auftrittswahrscheinlichkeiten der Zeichen:

Zeichen	p
a	0.3
b	0.1
c	0.1
d	0.2
e	0.3

a) Berechnen Sie den Entscheidungsgehalt

$$H_0 = log_2(N) = log_2(5) = 2.32\ bit$$

b) Berechnen Sie den Informationsgehalt

$$I(x_k) = -log_2(p(x_k))$$

Zeichen	p	I
a	0.3	1.74
b	0.1	3.32
c	0.1	3.32
d	0.2	2.32
e	0.3	1.74

c) Berechnen Sie die Entropie

$$H(x) = \sum^{N} p(x_k) \cdot I(x_k) = 2.16\ bit/Zeichen$$

Zeichen	p	I	$p \cdot I$
a	0.3	1.74	0.52
b	0.1	3.32	0.33
c	0.1	3.32	0.33
d	0.2	2.32	0.46
e	0.3	1.74	0.52
		$\sum$	2.16

d) Berechnen Sie die Redundanz der Quelle

$$R_Q = H_0 - H(x) = 2.32 - 2.16 = 0.16\ bit$$

2) Sie codieren Ihre Zeichen mit folgenden Codeworten:

Zeichen	Codewort
a	0
b	110
c	1111
d	1110
e	10

a) Berechnen Sie die mittlere Codewortlänge

$$L = \sum^{N} p(x_k) \cdot L(x_k) = 2.4\ bit$$

Zeichen	p	L	$p \cdot L$
a	0.3	1	0.3
b	0.1	3	0.3
c	0.1	4	0.4
d	0.2	4	0.8
e	0.3	2	0.6
		$\sum$	2.4

b) Berechnen Sie die Redundanz des Codes

$$R_C = L - H(x) = 2.4 - 2.16 = 0.24\ bit$$

3) Finden Sie eine bessere Codierung, d.h. eine mit weniger Redundanz?

Ziel ist es, für häufigen Zeichen möglichst kurzen Codeworte zu verwenden. Das erreichen wir z.B., indem wir die Codeworte für b und d tauschen. Wir erhalten damit eine mittlere Codewortlänge von:

$$L = \sum^{N} p(x_k) \cdot L(x_k) = 2.3\ bit$$

Zeichen	Codewort	p	L	$p \cdot L$
a	0	0.3	1	0.3
b	1110	0.1	4	0.4
c	1111	0.1	4	0.4
d	110	0.2	3	0.6
e	10	0.3	2	0.6
			$\sum$	2.3

Und eine Redundanz von:

$$R_C = L - H(x) = 2.3 - 2.16 = 0.14\ bit$$

Eine noch bessere Codierung wäre z.B.:

Zeichen	Codewort	p	L	$p \cdot L$
a	00	0.3	2	0.6
b	110	0.1	3	0.3
c	111	0.1	3	0.3
d	01	0.2	2	0.4
e	10	0.3	2	0.6
			$\sum$	2.2

$$R_C = L - H(x) = 2.2 - 2.16 = 0.04\ bit$$

4) Was sagt Ihnen das Codierungstheorem von Shannon in diesem Fall?

$$H(x) < L < H(x) + 1$$
$$2.16 < 2.2 \ / \ 2.3 < 3.16$$

Für die Quelle sind wir mit dieser Codierung nahe an der minimale mittlere Codewortlänge angelangt.

Aufgabe 2 Binäre Quelle ohne Gedächtnis

1) Zeigen Sie rechnerisch, dass die Entropie der binären Quelle maximal wird bei p = 0.5.

Wir berechnen das Maximum, indem wir H ableiten und gleich Null setzen:

$$\begin{aligned}
\frac{dH}{dp} &= \frac{d}{dp}\left(-p \cdot log_2(p) - (1-p) \cdot log_2(1-p)\right) \\
&= \frac{d}{dp}\left(-\frac{p \cdot ln(p)}{ln(2)} - \frac{(1-p)ln(1-p)}{ln(2)}\right) \\
&= \frac{1}{ln(2)}\left(-1 \cdot ln(p) - p\frac{1}{p} - (-1) \cdot ln(1-p) - (1-p)\frac{1}{1-p} \cdot (-1)\right) \\
&= \frac{1}{ln(2)}\left(-ln(p) - 1 + ln(1-p) + 1\right) \\
&= \frac{1}{ln(2)}\left(-ln(p) + ln(1-p)\right) \stackrel{!}{=} 0
\end{aligned} \tag{1}$$

Und erhalten:

$$\begin{aligned}
1 - p &= p \\
p &= \frac{1}{2}
\end{aligned}$$

2) Wann wird die Entropie maximal bei einer Quelle mit drei Zeichen?

Wir berechnen das Maximum, indem wir H partiell ableiten und jeweils gleich Null setzen:

$$\begin{aligned} H &= -p_1 log_2(p_1) - p_2 log_2(p_2) - (1-p_1-p_2) log_2(1-p_1-p_2) \\ &= \frac{1}{ln(2)} \left(-p_1 ln(p_1) - p_2 ln(p_2) - (1-p_1-p_2) ln(1-p_1-p_2)\right) \\ \frac{\delta H}{\delta p_1} &= \frac{1}{ln(2)} \left(ln(1-p_1-p_2) - ln(p_1)\right) \stackrel{!}{=} 0 \\ \frac{\delta H}{\delta p_2} &= \frac{1}{ln(2)} \left(ln(1-p_1-p_2) - ln(p_2)\right) \stackrel{!}{=} 0 \end{aligned}$$

Wir erhalten die beiden Gleichungen:

$$\begin{aligned} ln(p_1) &= ln(1-p_1-p_2) \\ ln(p_2) &= ln(1-p_1-p_2) \end{aligned}$$

Und damit:

$$\begin{aligned} 2p_1 + p_2 &= 1 \\ p_1 + 2p_2 &= 1 \end{aligned}$$

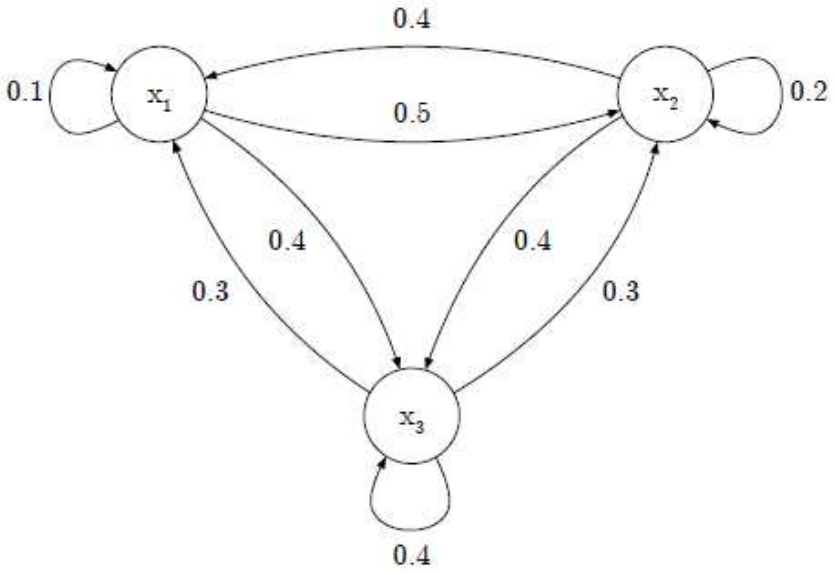
Somit ist $p_1 = p_2 = \frac{1}{3}$.

3) Was schliessen Sie allgemein aus obigen Feststellungen?

Die Entropie ist maximal, wenn alle Zeichen gleichwahrscheinlich auftreten.

Aufgabe 3 Diskrete Quelle mit Gedächtnis

Gegeben sei eine Markov-Quelle erster Ordnung[1] mit drei Symbolen, die alle T Sekunden ein Symbol erzeugt. Die Quelle hat somit drei Zustände, die jeweils durch das zuletzt erzeugte Symbol (x_1, x_2, x_3) gekennzeichnet werden. Diese Zustände mit den Übertragungswahrscheinlichkeiten im stationären Fall[2] sind im nachstehenden Markov-Diagramm wiedergegeben.



1) Berechnen Sie die bedingte Wahrscheinlichkeit P(Y|X), die jeweiligen Wahrscheinlichkeiten der Symbole P(x_i) und die Verbundwahrscheinlichkeiten P(X,Y).

Die bedingten Wahrscheinlichkeiten pro Zeichen können direkt aus dem Markov-Diagramm abgelesen werden.

$P(Y\|X)$	x_1	x_2	x_3
x_1	0.1	0.5	0.4
x_2	0.4	0.2	0.4
x_3	0.3	0.3	0.4

Die Wahrscheinlichkeiten der einzelnen Zeichen müssen mittels des folgenden Gleichungssystems berechnet werden:

$$\left|\begin{array}{c} P(x_1) = P(x_1)\cdot 0.1 + P(x_2)\cdot 0.4 + P(x_3)\cdot 0.3 \\ P(x_2) = P(x_1)\cdot 0.5 + P(x_2)\cdot 0.2 + P(x_3)\cdot 0.3 \\ P(x_3) = P(x_1)\cdot 0.4 + P(x_2)\cdot 0.4 + P(x_3)\cdot 0.4 \\ P(x_1) + P(x_2) + P(x_3) = 1 \end{array}\right|$$

Auflösen des Gleichungssystems führt zu den folgenden Lösungen:

$$P(x_1) = 0.28$$
$$P(x_2) = 0.32$$
$$P(x_3) = 0.4$$

[1] Die Wahrscheinlichkeit eines Zustandes hängt nur von dem unmittelbar vorhergehenden Zustand ab.

[2] Die statistischen Eigenschaften bleiben über die Zeit konstant.

Mit Hilfe der bedingten Wahrscheinlichkeiten und der Wahrscheinlichkeiten der einzelnen Zeichen, kann nun $P(X,Y)$ berechnet werden mit der Formel $P(X,Y) = P(X_i) \cdot P(Y_k|X_i)$:

$P(X,Y)$	x_1	x_2	x_3
x_1	0.028	0.14	0.112
x_2	0.128	0.064	0.128
x_3	0.12	0.12	0.16

2) Berechnen Sie die Entropien H(X), H(X, Y) und H(Y|X).

Die Entropie der Quelle $H(X)$ kann wie üblich berechnet werden:

x	$P(x)$	$I(x)$	$P(x)I(x)$
x_1	0.28	1.84	0.52
x_2	0.32	1.64	0.52
x_3	0.4	1.32	0.53
		$H(X)$	1.57

Weiter kann die Verbundentropie $H(X,Y)$ berechnet werden:

$$
\begin{aligned}
H(X,Y) &= \sum_{i=1}^{N}\sum_{k=1}^{N} P(x_i,y_k) \cdot log_2(\frac{1}{P(x_i,y_k)}) \\
&= 0.028 \cdot log_2(\frac{1}{0.028}) + 0.14 \cdot log_2(\frac{1}{0.14}) + 0.112 \cdot log_2(\frac{1}{0.112}) \\
&\quad + 0.128 \cdot log_2(\frac{1}{0.128}) + 0.064 \cdot log_2(\frac{1}{0.064}) + 0.128 \cdot log_2(\frac{1}{0.128}) \\
&\quad + 0.12 \cdot log_2(\frac{1}{0.12}) + 0.12 \cdot log_2(\frac{1}{0.12}) + 0.16 \cdot log_2(\frac{1}{0.16}) \\
&= 0.144 + 0.397 + 0.354 \\
&\quad + 0.38 + 0.254 + 0.38 \\
&\quad + 0.367 + 0.367 + 0.423 \\
&= 3.066\ bit
\end{aligned}
$$

Zuletzt ist die bedingte Entropie $H(Y|X)$ gefragt. Diese kann nun ganz einfach berechnet werden:

$$H(Y|X) = H(X,Y) - H(X) = 3.066 - 1.57 = 1.496\ bit$$

Übung 6

Quellencodierung – von Entropie zur optimalen Kompression

Lösung

V1.0

Hinweise:

- Übungen sind mit Vorteil alleine zu lösen
- Benutzen Sie die Musterlösung nur zur Korrektur
- Die Übungen sind wichtige Vorbereitungen für die Prüfung. Lösen Sie die Übungen sorgfältig und stellen Sie die Lösungswege übersichtlich dar.
- (Ergänzte) Vorlesungsunterlagen und Fachbücher helfen beim Lösen von Übungen und bringen gleichzeitig eine erweiterte Ansicht auf die Problemstellung.
- Wenn Sie die Übungen nicht verstehen, fragen Sie!

Aufgabe 1 Huffman-Codierung

Gegen sind die folgenden Zeichen (A, B, C, D, E) mit ihren Auftrittswahrscheinlichkeiten:

P(A) = 0.5
P(B) = 0.25
P(C) = 0.1
P(D) = 0.1
P(E) = 0.05

1) Berechnen Die die Redundanz R_Q der Quelle.

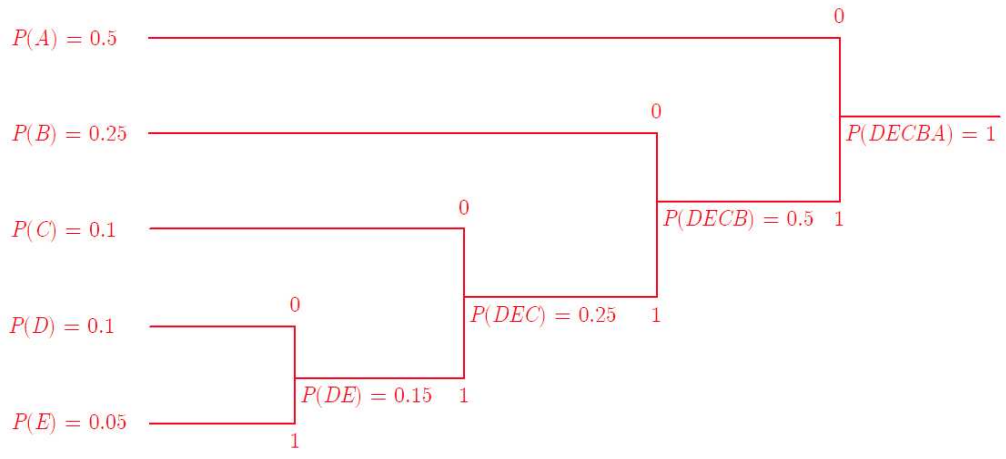
x	$P(x)$	$I(x)$	$P(x)I(x)$
A	0.5	1	0.5
B	0.25	2	0.5
C	0.1	3.322	0.332
D	0.1	3.322	0.332
E	0.05	4.322	0.216
		$H(X)$	1.880

$$R_Q = H_0 - H(X) = log_2(5) - 1.880 = 0.442\ bit$$

2) Nehmen Sie eine Codierung nach Huffman vor. Um wieviel Prozent können Sie die Redundanz des Codes minimieren?

Für eine nicht komprimierte Übertragung müsste die Codewortlänge L gleich 3 gewählt werden. Damit ergäbe sich eine Coderedundanz von $R_C = 3 - 1.88 = 1.12\ bit$.

Nach der Huffman-Codierung gilt folgendes:



x	$P(x)$	$L(x)$	$P(x)L(x)$
A	0.5	1	0.5
B	0.25	2	0.5
C	0.1	3	0.3
D	0.1	4	0.4
E	0.05	4	0.2
		L	1.9

$$R_C = L - H(X) = 1.9 - 1.88 = 0.02\ bit$$

Das entspricht einer Verbesserung von $100\% - 100\% \cdot \frac{0.02}{1.12} = 98.21\%$.

Aufgabe 2 Lauflängencodierung

Bei der Lauflängencodierung werden Sequenzen von identischen Symbolen durch deren Anzahl und (falls notwendig) das Symbol ersetzt.

1) DNA
   a) Codieren Sie die ersten 60 Basenpaare der Nukleotidsequenz des Gens HSP22 der Fruchtfliege:

      gaataaatga agattttaat attaatagct aaaaaaaaac agaaaactta aattatttgt

      $g2at3atg2aga4t2ata2t2atagct8acag4ac2t3a2ta3tgt$

   b) Wie gross ist die resultierende Kompression?

      $$\frac{46}{60} = 76.6\%$$

   c) Geben Sie ein Beispiel für eine Sequenz, welche nur schlecht komprimiert werden kann.

      Je weniger Zeichen wiederholt werden, desto schlechter ist die Komprimierung. Ein Beispiel einer nur schlecht zu komprimierender Sequenz wäre somit:

      $$gatgatat...$$

2) Bit-Folgen
   a) Codieren Sie die folgende Bit-Fole:

0000 1110 1000 1111 0001 1111

Wir beginnen mit einer Null und erhalten den Code "4 3 1 1 3 4 3 5", welche wir mit jeweils drei Bit codieren:

0 100 011 001 001 011 100 011 101

b) Wie gross ist die resultierende Kompression?

Die codierte Nachricht ist länger als die ursprüngliche Nachricht, d.h. wir haben keine Kompression.

Aufgabe 3 LZW-Komprimierung

Gegeben sei eine einfache Lempel-Ziv-Welch Komprimierung (LZW), welche nur Ziffern 0-9 komprimiert. Die LZW Komprimierung startet mit einem Wörterbuch, welches bereits für jedes Zeichen einen Eintrag hat:

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

Die Codierung ist ähnlich wie beim Lempel-Ziv-Algorithmus:

- Suche im Wörterbuch eine möglichst lange Zeichenfolge, die mit den nächsten n zu codierenden Zeichen übereinstimmt.
- Speichere den Index des gefundenen Wörterbucheintrages.
- Bilde einen neuen Eintrag im Wörterbuch mit der gefundenen Zeichenfolge, plus das darauffolgende Zeichen.
- Verschiebe das Fenster um n+1 Zeichen.
- Wiederhole, bis alle Zeichen codiert sind.

Die Zeichenfolge «36363» generiert daher drei weitere Einträge im Wörterbuch und wird mit «3 6 10 3» codiert.

Buffer	Erkannte Zeichenfolge (Index)	Neuer Eintrag
36363	3 (3)	→ 10: 36
3**6**363	6 (6)	→ 11: 63
36**36**3	36 (10)	→ 12: 363
3636**3**	3 (3)	-

Index	Eintrag
10	36
11	63
12	363

1) Codieren Sie die Ziffernfolge «123123123123». Wie sehen das Wörterbuch und die codierte Nachricht aus? Wie gross ist die Kompression?

Buffer	Erkannte Zeichenfolge (Index)	neuer Eintrag
123123123123	1 (1)	→ 10: 12
1**2**3123123123	2 (2)	→ 11: 23
12**3**123123123	3 (3)	→ 12: 31
123**12**3123123	12 (10)	→ 13: 123
12312**31**23123	31 (12)	→ 14: 312
1231231**23**123	23 (11)	→ 15: 231
123123123**123**	123 (13)	-

Wir erhalten die Nachricht "1 2 3 10 12 11 13"(7 Zeichen) mit folgendem Wörterbuch:

Index	Eintrag
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	12
11	23
12	31
13	123
14	312
15	231

Da sowohl die Eingangsziffern als auch das Wörterbuch jeweils weniger als 16 Zeichen enthalten, können wir beide Folgen mit 4 Bit pro Zeichen codieren:

0001 0010 0011 0001 0010 0011 0001 0010 0011 0001 0010 0011
0001 0010 0011 1010 1100 1011 1101

Wir erhalten eine Kompressionsrate von $\frac{28}{48} = 58.3\%$ (ohne Wörterbuch).

2) Wie sehen Wörterbuch und codierte Nachricht bei der Ziffernfolge «123456789987654321» aus?

Da keine bekannten Muster erkannt werden, entspricht die codierte Nachricht der ursprünglichen Nachricht: "123456789987654321" mit folgendem Wörterbuch:

Index	Eintrag	Index	Eintrag	Index	Eintrag
0	0	10	12	20	87
1	1	11	23	21	76
2	2	12	34	22	65
3	3	13	45	23	54
4	4	14	56	24	43
5	5	15	67	25	32
6	6	16	78	26	21
7	7	17	89		
8	8	18	99		
9	9	19	98		

3) Dekomprimieren Sie die Nachricht «7 5 10 11 12».

Da wir die ersten zehn Einträge des Wörterbuches kennen, können wir direkt die ersten zwei Zeichen decodieren und daraus den ersten zusätzlichen Eintrag konstruieren. Das machen wir so lange, bis wir den gesamten Text decodiert haben:

Nachricht	Decodiert	Eintrag
7 5 10 11 12	7	
7 5 10 11 12	75	→ 10: 75
7 5 10 11 12	7575	→ 11: 57
7 5 10 11 12	757557	→ 12: 755
7 5 10 11 12	75755775 5	→ 13: 577

Wir erhalten die Nachricht «757557755»

OST Ostschweizer Fachhochschule

Aufgabe 4 Kombination von Komprimierungen

Kann der Lempel-Ziv-Algorithmus und die Huffman-Codierung sinnvoll kombiniert werden?

Ja. Zuerst werden die Daten Lempel-Ziv-komprimiert, anschliessend das Wörterbuch Huffman-codiert, d.h. die gefundenen Phrasen werden entsprechend ihrer Häufigkeit codiert.

Aufgabe 5 Huffman-Codierung

Eine Quelle enthält einen Zeichenvorrat von 4 Zeichen (A, B, C, D). Die Auftrittswahrscheinlichkeiten der Zeichen seien wie folgt ermittelt worden:

P(A) = 0.8
P(B) = 0.1
P(C) = 0.08
P(D) = 0.02

1) Wie gross sind die Informationsgehalte I der einzelnen Zeichen?

$$I_A = log_2(\frac{1}{0.8}) = 0.322\ bit$$
$$I_B = log_2(\frac{1}{0.1}) = 3.322\ bit$$
$$I_C = log_2(\frac{1}{0.08}) = 3.644\ bit$$
$$I_D = log_2(\frac{1}{0.02}) = 5.644\ bit$$

2) Wie gross ist die Entropie H(X) der Quelle) Wie gross wäre die Entropie H'(X) der Quelle bei gleicher Auftrittswahrscheinlichkeit der vier Zeichen?

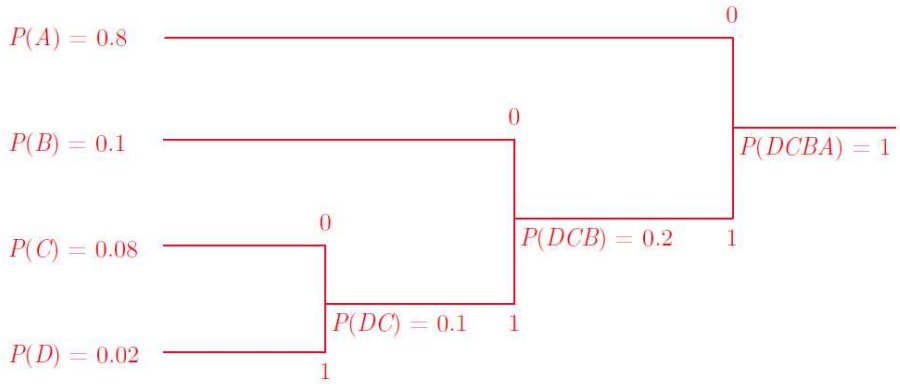
Mit den gegebenen Auftrittswahrscheinlichkeiten berechnet sich die Entropie wie folgt:

$$\begin{aligned} H(X) &= \sum_{k=1}^{4} P(x_k) \cdot I_k \\ &= 0.8 \cdot 0.322 + 0.1 \cdot 3.322 + 0.08 \cdot 3.644 + 0.02 \cdot 5.644 \\ &= 0.994\ bit \end{aligned}$$

Sind aber alle Zeichen gleich häufig, so wird die Entropie folgendermassen berechnet:

$$\begin{aligned} H'(X) &= \sum_{k=1}^{4} P(x_k) \cdot I_k \\ &= 4 \cdot (0.25 \cdot 2) \\ &= 2\ bit \end{aligned}$$

3) Entwerfen Die für die Übertragung einen binären Code nach Huffman.



Die resultierenden Codeworte sind:

$$A : 0$$
$$B : 10$$
$$C : 110$$
$$D : 111$$

4) Welche mittlere Codewortlänge L ergibt sich auf dem erstellten Code?

$$L = 0.8 \cdot 1 + 0.1 \cdot 2 + 0.08 \cdot 3 + 0.02 \cdot 3 = 1.3\ bit$$

OST Ostschweizer Fachhochschule

Aufgabe 6 Huffman-Codierung

Zur Übertragung von Nachrichten werden 8 verschiedene Zeichen (A, B, …, H) verwendet. Bisher waren die Nachrichtenzeichen alle mit der gleichen Wortlänge von drei Bit codiert. Bei einer mittleren Übertragungsrate von $6000\,\frac{Zeichen}{s}$ entspricht das einer Datenrate von $18000\,\frac{bit}{s}$. Die Auswertung von 18000 übertragenen Zeichen ergab die Häufigkeitsverteilung:

Zeichen	A	B	C	D	E	F	G	H
Anzahl	1200	4800	900	3990	1445	2900	2005	760

1) Wie gross ist die Redundanz R_Q der Quelle?

Die Redundanz der Quelle ist definiert als Entscheidungsgehalt minus der Entropie.
Im ersten Ansatz ist der Entscheidungsgehalt der Quelle gleich der mittleren Wortlänge L von 3Bit
Zur übersichtlichen Berechnung der Entropie siehe die untere Tabelle:

x	$P(x)$	$I(x)$	$P(x)I(x)$
A	0.0667	3.9	0.26
B	0.2667	1.9	0.51
C	0.05	4.32	0.22
D	0.221	2.17	0.48
E	0.08	3.64	0.29
F	0.161	2.63	0.42
G	0.111	3.17	0.35
H	0.042	4.57	0.19
		$H(X)$	2.72

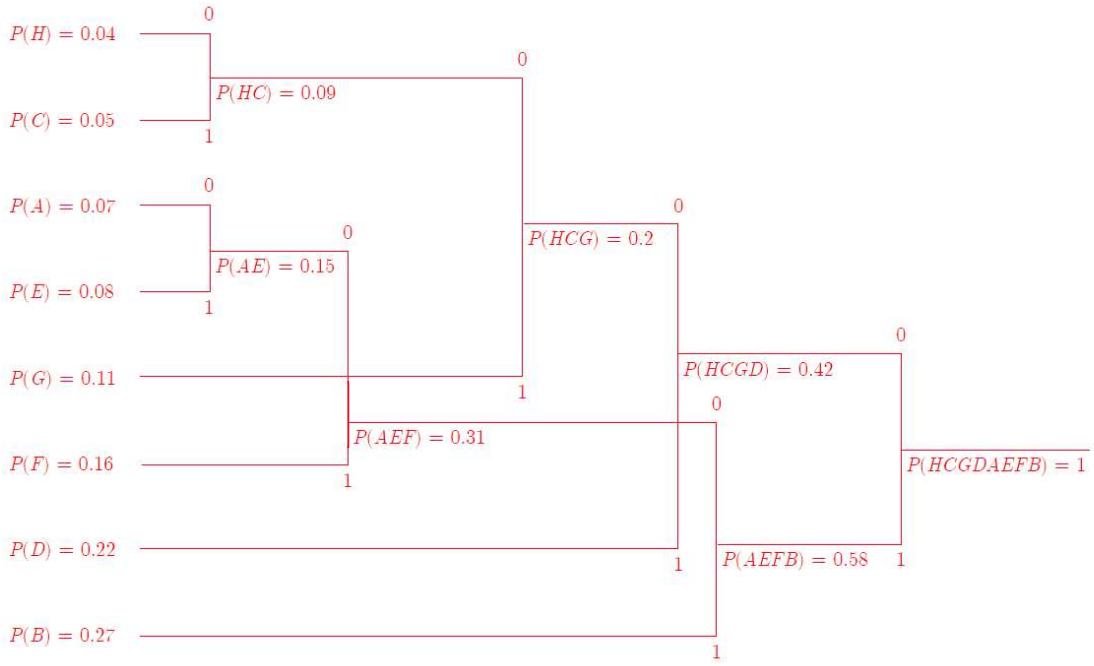
Damit ergibt sich eine Redundanz der Quelle von:

$$R_Q = H_0 - H(X) = log_2(8) - 2.72 = 0.28\ bit$$

Das entspricht in diesem Fall auch der Redundanz des Codes bei einer konstanten Codewortlänge von drei Bit.
Anm. Das muss nicht so sein!

2) Entwickeln Sie nach Huffman eine redundanzärmere Codierung der Codeworte. Was ist die resultierende Redundanzminderung?

Mit Hilfe des Huffman-Verfahrens erhalten wir einen optimierten Code für jedes Zeichen. Im Ideal wäre die Redundanz jetzt null, und zwar, wenn der Informationsgehalt des Zeichens genau soviel Bit hätte, wie zur Codierung benötigt werden, das trifft leider nur in konstruierten Fällen zu, wie beispielsweise in der Vorlesung!



Gemäss dieser Huffman-Codierung werden die Zeichen wie folgt codiert:

Zeichen	H	C	A	E	G	F	D	B
CW	0000	0001	1000	1001	001	101	01	11

Um daraus die Redundanz R_C zu berechnen, braucht man zuerst die mittlere Codewortlänge L:

x	$P(x)$	$L(x)$	$P(x) \cdot L(x)$
A	0.0667	4	0.2668
B	0.2667	2	0.5334
C	0.05	4	0.2
D	0.221	2	0.442
E	0.08	4	0.32
F	0.161	3	0.483
G	0.111	3	0.333
H	0.042	4	0.168
		L	2.7462

Der einzige Wert, der sich geändert hat, ist die mittlere Codewortlänge L, die jetzt ja nicht mehr konstant auf drei Bit eingestellt ist. Damit ergibt sich:

$$R_C = L - H(X) = 2.75 - 2.72 = 0.03\ bit$$

Da der Informationsgehalt erwartungsgemäss nicht genau der gewählten bzw. erforderlichen Codewortlänge entspricht bleibt eine Redundanz von 0.03 bit bestehen.

Hatten wir vorher bei konstanter Codewortlänge von 3 bit eine Redundanz von:

$$R'_C = 3 - H(X) = 3 - 2.72 = 0.28\ bit$$

So ergibt sich jetzt eine Redundanzminderung von $R'_C - R_C = 0.28 - 0.03 = 0.25\ bit$.

Zeichen	H	C	A	E	G	F	D	B
CW	0000	0001	1000	1001	001	101	01	11

Um daraus die Redundanz R_C zu berechnen, braucht man zuerst die mittlere Codewortlänge L:

x	$P(x)$	$L(x)$	$P(x) \cdot L(x)$
A	0.0667	4	0.2668
B	0.2667	2	0.5334
C	0.05	4	0.2
D	0.221	2	0.442
E	0.08	4	0.32
F	0.161	3	0.483
G	0.111	3	0.333
H	0.042	4	0.168
		L	2.7462

Der einzige Wert, der sich geändert hat, ist die mittlere Codewortlänge L, die jetzt ja nicht mehr konstant auf drei Bit eingestellt ist. Damit ergibt sich:

$$R_C = L - H(X) = 2.75 - 2.72 = 0.03\ bit$$

Da der Informationsgehalt erwartungsgemäss nicht genau der gewählten bzw. erforderlichen Codewortlänge entspricht bleibt eine Redundanz von 0.03 bit bestehen.

Hatten wir vorher bei konstanter Codewortlänge von 3 bit eine Redundanz von:

$$R'_C = 3 - H(X) = 3 - 2.72 = 0.28\ bit$$

So ergibt sich jetzt eine Redundanzminderung von $R'_C - R_C = 0.28 - 0.03 = 0.25\ bit$.

3) Um wieviel Prozent kann die mittlere Datenrate gesenkt werden, damit die gleiche Information für die Übertragung gleich lange braucht?

Um die Rate vom 6000 Zeichen/s bei einer mittleren Wortlänge von 3 bit zu übertragen, war bis her eine Datenrate von 18000bit/s erforderlich. Durch die Komprimierung reduziert sich mittlere Codewortlänge von 3 auf 2.75 bit und bei Datenmengen von 6000 Zeichen jeweils auf eine mittlere Datenrate von 16320 bit/s.

$3 \cdot 6000/s$ entspreche 100%, dann entsprechen $2.75 \cdot 6000/s$ gleich $\frac{100\%}{3 \cdot 6000} \cdot 2.75 \cdot 6000 = 91.67\%$.

Aufgabe 7 Implementierung LZ77 (Zusatzaufgabe)

Im Unterricht wurde im Rahmen der Lempel-Ziv-Komprimierung konkret der LZ77-Algorithmus besprochen. Implementieren Sie diesen in einer beliebigen Programmiersprache. Die Implementierung führt häufig zu einem tieferen Verständnis des Algorithmus (gilt für beliebige Algorithmen).

Übung 7

Quellencodierung – Kryptographie
Struktur statt Geheimnis

Lösung

V1.0

Hinweise:

- Übungen sind mit Vorteil alleine zu lösen
- Benutzen Sie die Musterlösung nur zur Korrektur
- Die Übungen sind wichtige Vorbereitungen für die Prüfung. Lösen Sie die Übungen sorgfältig und stellen Sie die Lösungswege übersichtlich dar.
- (Ergänzte) Vorlesungsunterlagen und Fachbücher helfen beim Lösen von Übungen und bringen gleichzeitig eine erweiterte Ansicht auf die Problemstellung.
- Wenn Sie die Übungen nicht verstehen, fragen Sie!

Aufgabe 1 Kryptographie

In dieser Aufgabe sollen Sie sich mit verschiedenen Verfahren zum Thema Verschlüsselung vertraut machen, um ein Gefühl dafür zu entwickeln. Dazu sollen Sie die bereitgestellte Infrastruktur von https://www.cryptoprograms.com verwenden.

Keine Angst, Sie brauchen keine Software zu installieren. Sie können alles bequem im Browser erledigen.

1) Caesar-Chiffre:
   a. Navigieren Sie via «Substitution (Create) » zu «Caesar». Wenn Sie sich vor Beginn der Aufgabe noch einmal mit dem Verfahren vertraut machen wollen, erhalten Sie über die Checkbox «Show description» eine Beschreibung.
   Nehmen Sie nun einen beliebigen Text und geben diesen im Feld «Enter plaintext tob e enciphered» ein. Wählen Sie einen Schlüssel und codieren Sie den Text. Das Ergebnis erscheint wahrlich kryptisch.
   b. Aber ist es wirklich so schwierig den Code zu knacken? Finden Sie die passende Analyse und versuchen Sie, Ihren Chiffre-Text zu entschlüsseln.
   c. Tauschen Sie mit einem Kollegen einen unbekannten verschlüsselten Text (Caesar Chiffre) aus. Analysieren Sie den Text und finden Sie den passenden Schlüssel.
   d. Hat die Textlänge oder die Sprache des Urtextes einen Einfluss auf den Erfolg der Analyse? Und wenn ja, welchen?

      Ja, sowohl die Textlänge als auch die Sprache des Urtextes können einen erheblichen Einfluss auf den Erfolg der Analyse einer mit der Cäsar-Verschlüsselung verschlüsselten Nachricht haben.

      Textlänge:
      1. Längere Texte: Bei längeren Texten ist es einfacher, die Verschlüsselung zu brechen, da mehr Daten zur Verfügung stehen, um die Häufigkeit einzelner Buchstaben oder Buchstabengruppen zu analysieren. In längeren Texten gleichen sich zufällige Abweichungen in der Häufigkeitsverteilung eher aus, und die charakteristischen Muster der Sprache (wie häufige Buchstaben oder Wortendungen) werden deutlicher.
      2. **Kürzere Texte: Bei kürzeren Texten ist es schwieriger, statistische Muster zu erkennen, da weniger Daten zur Analyse zur Verfügung stehen. Ein sehr kurzer Text kann unter Umständen zu wenig Information bieten, um die Verschlüsselung sicher zu brechen.

      Sprache:
      1. Sprachspezifische Buchstabenhäufigkeiten: Verschiedene Sprachen haben unterschiedliche Buchstabenhäufigkeiten. Zum Beispiel ist der häufigste Buchstabe im Englischen "e", während im Deutschen "e" ebenfalls sehr häufig ist, aber auch andere Buchstaben wie "n" und "i" sehr häufig vorkommen. Die Kenntnis der spezifischen Buchstabenhäufigkeit einer Sprache kann bei der Entschlüsselung einer Cäsar-Verschlüsselung entscheidend sein.
      2. Struktur und Syntax: Sprachen unterscheiden sich auch in ihrer Syntax und Struktur, was Auswirkungen auf die Häufigkeit bestimmter Buchstabenkombinationen oder Wortformen hat. Diese Unterschiede können die Analyse sowohl erleichtern als auch erschweren, je nachdem, wie charakteristisch diese Muster für eine Sprache sind.

      Zusammengefasst kann gesagt werden, dass sowohl die Länge des Textes als auch die spezifischen Eigenheiten der Sprache des Urtextes wichtige Faktoren sind, die den Erfolg der Analyse einer Cäsar-Verschlüsselung beeinflussen können. Je mehr man über diese Aspekte weiss, desto effektiver kann die Analyse durchgeführt werden.

2) Transpositionsverfahren: Wiederholen Sie die Schritte (a) - (d) aus Aufgabe 1.1. Benutzen Sie dafür den «Complete Columnar» Chiffre, den Sie unter «Transposition (Create)» finden.

Aufgabe 2 RSA

Gegeben seien die beiden Primzahlen p = 11 und q = 7. Berechnen Sie von Hand:

1) Die Zahl n und die Zahl Φ(n).

Die Zahl n ergibt sich aus dem Produkt der beiden gewählten Primzahlen p und q:
$n = p \cdot q = 77$
Weil p und q Primzahlen sind, kann Φ(n) wie folgt berechnet werden:
$\Phi(n) = \Phi(p \cdot q) = (p - 1) \cdot (q - 1) = 60$

2) Die zu $e_1 = 7$ und $e_2 = 13$ inversen Schlüssel d_1 und d_2 zur De-Chiffrierung.

Um den zu $e_1 = 7$ passenden inversen Schlüssel d_1 zu berechnen, kann der erweiterte euklidische Algorithmus angewendet werden. Das funktioniert, weil die Bedingung
$e_1 \cdot d_1 \equiv 1 \bmod \Phi(n)$ erfüllt sein muss.

Es gilt also $7 \cdot d_1 \bmod 60 = 1$. Wir starten mit der Faktorisierung:
$60 = 8 \cdot 7 + 4$
$7 = 1 \cdot 4 + 3$
$4 = 1 \cdot 3 + 1$

Nun kann wieder expandiert werden:
$1 = 4 - 1 \cdot 3$
$1 = 4 - 1 \cdot (7 - 1 \cdot 4) = 2 \cdot 4 - 1 \cdot 7$
$1 = 2 \cdot (60 - 8 \cdot 7) - 1 \cdot 7 = 2 \cdot 60 - 17 \cdot 7$

$\Rightarrow d_1 = -17 \bmod 60 = 43$

Das gleiche Verfahren kann nun für $e_2 = 13$ durchgeführt werden. Es gilt dann
$13 \cdot d_2 \bmod 60 = 1$. Wir starten wieder mit der Faktorisierung:
$60 = 4 \cdot 13 + 8$
$13 = 1 \cdot 8 + 5$
$8 = 1 \cdot 5 + 3$
$5 = 1 \cdot 3 + 2$
$3 = 1 \cdot 2 + 1$

Nun kann wieder expandiert werden:
$1 = 3 - 1 \cdot 2$
$1 = 3 - 1 \cdot (5 - 1 \cdot 3) = 2 \cdot 3 - 1 \cdot 5$
$1 = 2 \cdot (8 - 1 \cdot 5) - 1 \cdot 5 = 2 \cdot 8 - 3 \cdot 5$
$1 = 2 \cdot (60 - 4 \cdot 13) - 3 \cdot 5 = 2 \cdot 60 - 8 \cdot 13 - 3 \cdot 5$
$1 = 2 \cdot 60 - 8 \cdot 13 - 3 \cdot (13 - 1 \cdot 8) = 2 \cdot 60 - 11 \cdot 13 + 3 \cdot 8$
$1 = 2 \cdot 60 - 11 \cdot 13 + 3 \cdot (60 - 4 \cdot 13) = 5 \cdot 60 - 23 \cdot 13$

$\Rightarrow d2 = -23 \bmod 60 = 37$

3) Zeigen Sie an einem Schlüsselpaar mit dem Klartext m = 2, dass das Verfahren funktioniert.

Der Vollständigkeit halber werden in dieser Lösung beide Schlüsselpaare gezeigt. Es reicht aber, wenn Sie nur eines davon oder sogar ein anderes Schlüsselpaar zeigen.

Schlüsselpaar $(e_1, d_1) = (7, 43)$: Das Codewort c zu m = 2 berechnet sich wie folgt:

$c = m^{e1} \bmod n = 2^7 \bmod 77 = 51$

Um das Verfahren zu überprüfen, können Sie aus dem Codewort wie folgt wieder die Nachricht berechnen:

$m' = c^{d1} \bmod n = 51^{43} \bmod 77 = 2$

Schlüsselpaar $(e_2, d_2) = (13, 37)$: Das Codewort c zu m = 2 berechnet sich wie folgt:

$c = m^{e2} \bmod n = 2^{13} \bmod 77 = 30$

Um das Verfahren zu überprüfen, können Sie aus dem Codewort wie folgt wieder die Nachricht berechnen:

$m' = c^{d2} \bmod n = 30^{37} \bmod 77 = 2$

4) Worin sehen Sie in der Handhabung eines asymmetrischen Verfahrens Probleme?

Performance: Wie Sie sich nach den bisher gelösten Aufgaben vorstellen können, sind asymmetrische Algorithmen sehr langsam (ca. 10'000 Mal langsamer als symmetrische Verfahren)[1].

Infrastruktur: Der öffentliche Schlüssel muss über eine vertrauenswürdige Stelle abrufbar sein. Die Sicherstellung der Vertrauenswürdigkeit und die Bestimmung der Herkunft des öffentlichen Schlüssels sind mit grossem Aufwand verbunden.

Mehrere Empfänger: Bei mehreren Empfängern muss die Nachricht mit dem öffentlichen Schlüssel jedes einzelnen Empfängers verschlüsselt werden.

Man-In-The-Middle: Wenn es ein Angreifer schafft, seinen öffentlichen Schlüssel als den des eigentlichen Empfängers anzupreisen, kann er mit seinem privaten Schlüssel die Nachricht entschlüsseln. Um nicht aufzufallen, kann er zudem die Nachricht weiter mit dem öffentlichen Schlüssel des eigentlichen Empfängers wieder verschlüsseln und dann die Nachricht weiterschicken. Um das zu verhindern, muss die Authentizität des öffentlichen Schlüssels gewährleistet sein[2].

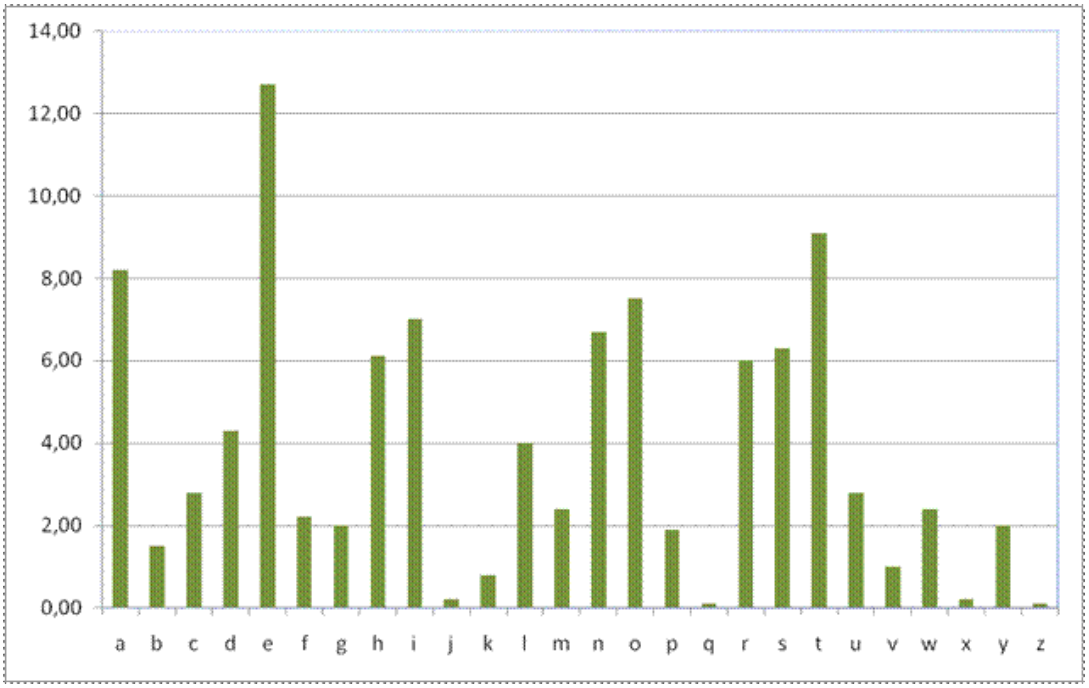
[1] https://www.kryptowissen.de/asymmetrische-verschluesselung.html
[2] https://www.philipphauer.de/info/info/asymmetrische-verschluesselung/#nachteile_asymmetrisch

Aufgabe 3 Entschlüsselung

Versuchen Sie die folgende Nachricht zu entschlüsseln!

V SENGTGGTGPVHWA

Sie vermuten eine englische Nachricht und haben daher die entsprechende Häufigkeitsverteilung Herausgesucht[3]:



Die Nachricht enthält eine vierfache Anzahl an G sowie eine doppelte Anzahl T und V. Im Englischen ist der mit Abstand häufigste Buchstabe das E. Wir versuchen daher einen Caesar Chiffre von −2 und erhalten die Nachricht:

T QCLEREEERENTFUY

Die Nachricht sieht nun bereits in Bezug auf die Häufigkeitsverteilung der Buchstaben besser aus, schein jedoch noch verwürfelt zu sein. Wir versuchen ein Transpositionsverfahren anzuhängen und erstellen dazu eine 4x4-Matrix:

T	<space>	Q	C
L	E	R	E
E	R	E	N
T	F	U	Y

Mit etwas ausprobieren können wir die Verwürfelung ausfindig machen:

L	E	R	E

[3] http://www.arne-lueker.de/Objects/projects/Krypto.html

E	R	E	N
T	<space>	Q	C
T	F	U	Y

Wir erhalten die entschlüsselte Nachricht:

LETTER FREQUCENCY

Aufgabe 4 Kombination Komprimierung und Verschlüsselung

Sie wollen für einen Datentransfer Ihre Daten komprimieren und verschlüsseln. In welcher Reihenfolge machen Sie das und wieso?

Da eine gute Verschlüsselung ihre Daten pseudozufällig macht, hat eine anschliessende Datenkompression keine Wirkung mehr - die Zeichen werden gleich wahrscheinlich. Sie müssen daher die Daten zuerst komprimieren und erst anschliessend verschlüsseln. Zudem maximiert die Komprimierung die Entropie der Nachricht und reduziert so etwas die Angriffsfläche für statistische Angriffe auf die Verschlüsselung.

Aufgabe 5 Euklidischer Algorithmus

Berechnen Sie den grössten gemeinsamen Teiler mit Hilfe des euklidischen Algorithmus:

1) ggT(48, 18)

48 = 2 · 18 + 12
18 = 1 · 12 + 6
12 = 2 · 6 + 0

Der grösste gemeinsame Teiler von 48 und 18 ist 6.

2) ggT(19, 13)

19 = 1 · 13 + 6
13 = 2 · 6 + 1
1 = 1 · 1 + 0

Der grösste gemeinsame Teiler von 19 und 13 ist 1.

Aufgabe 6 Erweiterter euklidischer Algorithmus

Berechnen Sie d mit Hilfe des erweiterten euklidischen Algorithmus, falls $47 \cdot d \equiv 1 \; mod \; 60$.

a	b	q	r	x	y
47	60	0	47	23	-18
60	47	1	13	-18	23
47	13	3	8	5	-18
13	8	1	5	-3	5
8	5	1	3	2	-3
5	3	1	2	-1	2
3	2	1	1	1	-1
2	1	2	0	0	1

$d = 23, Probe: 47 \times 23 = 1081 \; mod \; 60 = 1$

Aufgabe 7 Verschlüsseln und Entschlüsseln mit RSA

Verschlüsseln und Entschlüsseln Sie nun den Klartext «2» mit folgenden Parametern:

p=7, q=11, e=47

n=77, Φ(77)=60, e=47 ➔ öffentlicher Schlüssel: (47,77)

d=23 (siehe Aufgabe 6) ➔ privater Schlüssel: (23,77)

Verschlüsseln:

$$2^{47}\ mod\ 77 \equiv 2^5(2^7)^6 \equiv 32 \cdot 51^6 \equiv 32 \cdot (51^2)^3 \equiv 32 \cdot 60^3 \equiv 32 \cdot (-17)^3 \equiv 32 \cdot -17 \cdot 17^2 \equiv -5 \cdot 58$$
$$\equiv -5 \cdot -19 \equiv 95 \equiv 18\ mod\ 77$$

Geheimtext = 18

Entschlüsseln:

$$18^{23}\ mod\ 77 = 2$$

Klartext = 2

Übung 09

Kanalmodell
Wie Information durch den Kanal verzerrt wird
Lösung

V1.0

Hinweise:

- Übungen sind mit Vorteil alleine zu lösen
- Benutzen Sie die Musterlösung nur zur Korrektur
- Die Übungen sind wichtige Vorbereitungen für die Prüfung. Lösen Sie die Übungen sorgfältig und stellen Sie die Lösungswege übersichtlich dar.
- (Ergänzte) Vorlesungsunterlagen und Fachbücher helfen beim Lösen von Übungen und bringen gleichzeitig eine erweiterte Ansicht auf die Problemstellung.
- Wenn Sie die Übungen nicht verstehen, fragen Sie!

Aufgabe 1 Kanalmatrix

1) Geben Sie eine Kanalmatrix eines nicht gestörten Kanals an.

Ein nicht gestörter Kanal hat die Einheitsmatrix als Kanalmatrix, also z.B. für einen Binärkanal:

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

2) Geben Sie eine Kanalmatrix eines vollständig gestörten Kanals an.

Bei einem vollständig gestörten Kanal sind alle Wahrscheinlichkeiten gleichverteilt, also z.B. für einen Binärkanal:

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

Aufgabe 2 Maximale Symbolrate

Gegeben ist der Binärkanal mit der folgenden Kanalmatrix:

$$P(Y|X) = \begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix}$$

Über den oben definierten Kanal sollen die Symbole x_1 und x_2 einer binären Quelle übertragen werden. Die Übertragungsrate sei 1 kbit/s. Die Auftrittswahrscheinlichkeiten der Zeichen der Quelle sind $p(x_1)=0.3$ und $p(x_2)=0.7$.

1) Wie gross ist die Entropie H(X) am Kanaleingang?

$$\begin{aligned} H(X) &= -\sum_{k=1}^{2} p(x_k) \cdot log_2(p(x_k)) \\ &= -0.3 \cdot log_2(0.3) - 0.7 \cdot log_2(0.7) \\ &= 0.8813 \; bit/Zeichen \end{aligned}$$

2) Wie gross ist die Entropie H(Y) am Kanalausgang?

$$p(y_1) = p(x_1)p(y_1|x_1) + p(x_2)p(y_1|x_2) = 0.3 \cdot 0.9 + 0.7 \cdot 0.1 = 0.34$$
$$p(y_2) = p(x_1)p(y_2|x_1) + p(x_2)p(y_2|x_2) = 0.3 \cdot 0.1 + 0.7 \cdot 0.9 = 0.66$$

$$\begin{aligned} H(Y) &= -\sum_{k=1}^{2} p(y_k) \cdot log_2(p(y_k)) \\ &= -0.34 \cdot log_2(0.34) - 0.66 \cdot log_2(0.66) \\ &= 0.9248\ bit/Zeichen \end{aligned}$$

3) Welche maximale Symbolrate $R_{max}\left[\frac{Zeichen}{s}\right]$ kann der Kanal im Prinzip fehlerfrei übertragen?

Gesucht ist die Transinformation.

Irrelevanz:

$$\begin{aligned} H(Y|X) &= -\sum_{i=1}^{2}\sum_{k=1}^{2} p(x_i, y_k) \cdot log_2(p(y_k|x_i)) \\ &= -\sum_{i=1}^{2}\sum_{k=1}^{2} p(x_i) \cdot p(y_k|x_i) \cdot log_2(p(y_k|x_i)) \\ &= -0.3 \cdot 0.9 \cdot log_2(0.9) - 0.3 \cdot 0.1 \cdot log_2(0.1) \\ &\quad -0.7 \cdot 0.9 \cdot log_2(0.9) - 0.7 \cdot 0.1 \cdot log_2(0.1) \\ &= 0.469\ bit/Zeichen \end{aligned}$$

Transinformation:

$$T = H(Y) - H(Y|X) = 0.455\ bit/Zeichen$$

Maximale Rate:

$$R_{max} = 0.455\ bit/Zeichen \cdot 1000\ Zeichen/s = 455\ bit/s$$

Aufgabe 3 Entscheider

Ein Kanal sei durch die folgende Kanalmatrix beschrieben:

$$P(Y|X) = \begin{bmatrix} 0.2 & 0.5 & 0.3 \\ 0.7 & 0.2 & 0.1 \\ 0.4 & 0 & 0.6 \end{bmatrix}$$

1) Bestimmen Sie nach dem Maximum-Likelihood-Verfahren den Entscheider.

In der Kanalmatrix kann man den Entscheider ablesen, indem nacheinander die Zuordnungen mit der jeweils grössten verbleibenden Wahrscheinlichkeit auswählt.

$$P(Y|X) = \begin{bmatrix} 0.2 & \textcircled{0.5} & 0.3 \\ \textcircled{0.7} & 0.2 & 0.1 \\ 0.4 & 0 & \textcircled{0.6} \end{bmatrix}$$

Der Entscheider sieht also wie folgt aus:

$$y_1 \rightarrow x_2$$
$$y_2 \rightarrow x_1$$
$$y_3 \rightarrow x_3$$

2) Berechnen Sie die Fehlerwahrscheinlichkeit, wenn die Eingangszeichen gemäss folgender Häufigkeitsanalyse auftreten:

x1: 550
x2: 1200
x3: 3000

Aus den gegebenen Häufigkeiten müssen zuerst die zugehörigen Auftrittswahrscheinlichkeiten berechnet werden:

$$p(x_1) = \frac{550}{4750} = 0.116$$
$$p(x_2) = \frac{1200}{4750} = 0.253$$
$$p(x_3) = \frac{3000}{4750} = 0.631$$

Dann kann man wie folgt weiterfahren:

$$\begin{aligned} P(\text{Keine Fehler}) &= 0.7 \cdot p(x_2) + 0.5 \cdot p(x_1) + 0.6 \cdot p(x_3) \\ &= 0.7 \cdot 0.253 + 0.5 \cdot 0.116 + 0.6 \cdot 0.631 \\ &= 0.614 \end{aligned}$$

Und schliesslich:

$$P(\text{Fehler}) = 1 - P(\text{Keine Fehler}) = 1 - 0.614 = 0.386$$

3) Kann ein anderer Entscheider zu einem besseren Ergebnis, d.h. zu einer geringeren Restfehlerwahrscheinlichkeit führen?

Ein besseres Resultat liefert zum Beispiel der folgende Entscheider:

$$y_1 \rightarrow x_3$$
$$y_2 \rightarrow x_1$$
$$y_3 \rightarrow x_3$$

Um sich das besser vorstellen zu können, ist diese Auswahl an der Kanalmatrix noch einmal verdeutlicht.

$$P(Y|X) = \begin{bmatrix} 0.2 & \textcircled{0.5} & 0.3 \\ 0.7 & 0.2 & 0.1 \\ \textcircled{0.4} & 0 & \textcircled{0.6} \end{bmatrix}$$

Analog zur Berechnung aus der vorherigen Aufgabe resultiert die Fehlerwahrscheinlichkeit $P(\text{Fehler}) = 0.311$. Die Fehlerwahrscheinlichkeit ist also geringer als zuvor, jedoch wird das Codewort x_2 nicht mehr ausgewertet.

Aufgabe 4 Bestimmung der Kanalmatrix

Wie können Sie die Kanalmatrix eines Kanals für ein Eingabealphabet praktisch ermitteln (kurze Beschreibung in Worten)?

Sie führen viele Messungen (senden eines bekannten Zeichens, festhalten des empfangenen Zeichens) durch und errechnen aus diesen Messungen die Häufigkeiten. Daraus kann dann die Kanalmatrix erstellt werden.

Aufgabe 5 Kanalmatrix

Geben Sie eine Kanalmatrix eines verlustfreien, rauschbehafteten Kanals an.
Hinweis: die Matrix muss nicht zwingend symmetrisch sein!

$$P(Y|X) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.4 & 0.6 \end{bmatrix}$$

Dies ist nur eines von vielen Beispielen. Beachten Sie, dass die Summe der Zeilen gleich 1 sein muss, da ein Ereignis unbedingt eintreten muss. Eine eindeutige Zuordnung bzw. Erkennung der gesendeten Signale ist dadurch möglich, dass ...

1. ... das Signal x_1 eindeutig auf y_1 abgebildet wird und ...
2. ... das Signal x_2 entweder auf y_2 oder y_3 abgebildet wird und x_3 nicht existiert (asymmetrischer Kanal).

Aufgabe 6 Transinformation

Messungen an einem symmetrischen Kanal ergeben die unten angegebenen Auftrittswahrscheinlichkeiten der Zeichen am Kanaleingang x_i respektive am Kanalausgang y_i mit:
$x_1 = 0.3$ und $x_2 = 0.7$
$y_1 = 0.34$ und $y_2 = 0.66$.

1) Bestimmen Sie die Kanalmatrix.

Aus

$$P(Y) = P(X)^T \cdot P(Y|X)$$

$$\begin{bmatrix}0.34 & 0.66\end{bmatrix} = \begin{bmatrix}0.3 & 0.7\end{bmatrix} \cdot \begin{bmatrix}a & b\\ b & a\end{bmatrix} = \begin{bmatrix}(0.3a + 0.7b) & (.07a + 0.3b)\end{bmatrix}$$

folgt das lineare Gleichungssystem

$$0.34 = 0.3a + 0.7b$$
$$0.66 = 0.3b + 0.7a$$

mit $a = 0.9$ und $b = 0.1$ und damit

$$P(Y|X) = \begin{bmatrix}0.9 & 0.1\\ 0.1 & 0.9\end{bmatrix}$$

2) Wie gross ist die Transinformation des Kanals?

Ausgangsentropie:

$$H(Y) = -\sum_i^N p(y_i) \cdot log_2(p(y_i))$$
$$H(Y) = 0.34 \cdot log_2(0.34) + 0.66 \cdot log_2(0.66) = 0.92$$

Irrelevanz:

$$H(Y|X) = -\sum_i^N \sum_j^N p(x_i) \cdot p(y_j|x_i) \cdot log_2(p(y_j|x_i))$$
$$\begin{aligned} H(Y|X) = {} & 0.3 \cdot 0.9 \cdot log_2(0.9) \\ & + 0.3 \cdot 0.1 \cdot log_2(0.1) \\ & + 0.7 \cdot 0.9 \cdot log_2(0.9) \\ & + 0.7 \cdot 0.1 \cdot log_2(0.1) \end{aligned}$$
$$H(Y|X) = 0.47$$

Transinformation:

$$T = H(Y) - H(Y|X) = 0.92 - 0.47 = 0.45$$

3) Wie lange benötigen Sie mindestens zur korrekten Übertragung eines 500Mbit Blocks, wenn der Kanal eine Übertragungsrate von 140 kbit/s besitzt?

$$R_{max} = T \cdot R = 0.45 \cdot 140 \cdot 10^3 \ = 63000 bit/s$$

$$t = \frac{L}{R_{max}} = \frac{500 \cdot 10^6}{63 \cdot 10^3} = 7940s = 2.2h$$

Aufgabe 7 Transinformation

Zeichnen Sie die Transinformation für einen symmetrischen Binärkanal in Funktion der Zeichenwahrscheinlichkeit p und der Fehlerwahrscheinlichkeit ϵ auf.

$$P(Y) = P(X)^T \cdot P(Y|X)$$

$$P(Y) = [p \quad 1-p] \cdot \begin{bmatrix} 1-\epsilon & \epsilon \\ \epsilon & 1-\epsilon \end{bmatrix} = [(\epsilon + p - 2\epsilon p) \quad (1-(\epsilon + p - 2\epsilon p))]$$

Wir berechnen hier nur einige Punkte: $p = (0.0, 0.1, 0.2, 0.5)$ und $\epsilon = (0.0, 0.1, 0.2, 0.5)$.

Ausgangswahrscheinlichkeiten $P(Y) = P(Y|X) \cdot P(X)$

$P(Y)$		p			
		0.0	0.1	0.2	0.5
ϵ	0.0	[0.0, 1.0]	[0.1, 0.9]	[0.2, 0.8]	[0.5, 0.5]
	0.1	[0.1, 0.9]	[0.18, 0.82]	[0.26, 0.74]	[0.5, 0.5]
	0.2	[0.2, 0.8]	[0.26, 0.74]	[0.32, 0.68]	[0.5, 0.5]
	0.5	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]

Ausgangsentropie $H(Y) = -\sum_i^N p(y_i) \cdot log_2(p(y_i))$

$H(Y)$		p			
		0.0	0.1	0.2	0.5
ϵ	0.0	-0.0	0.47	0.72	1.0
	0.1	0.47	0.68	0.83	1.0
	0.2	0.72	0.83	0.9	1.0
	0.5	1.0	1.0	1.0	1.0

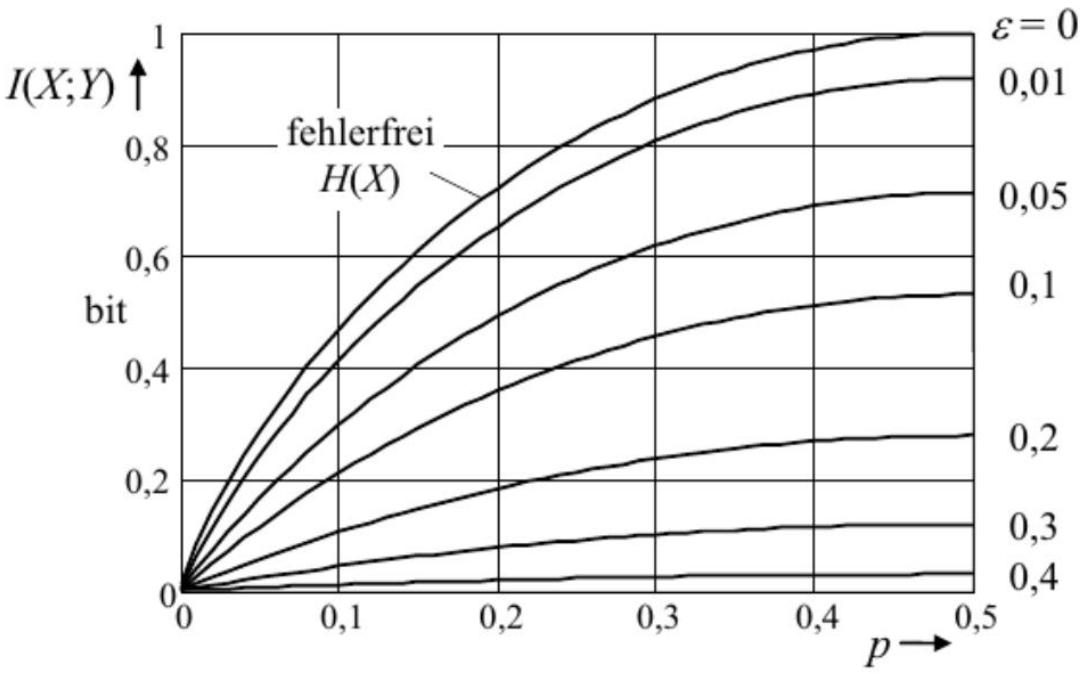
Irrelevanz $H(Y|X) = -\sum_i^N \sum_j^N p(x_i) \cdot p(y_j|x_i) \cdot log_2(p(y_j|x_i))$

$H(Y\|X)$		p			
		0.0	0.1	0.2	0.5
ϵ	0.0	-0.0	0.47	0.72	1.0
	0.1	-0.0	0.47	0.72	1.0
	0.2	-0.0	0.47	0.72	1.0
	0.5	-0.0	0.47	0.72	1.0

Transinformation $T = H(Y) - H(Y|X)$

T		p			
		0.0	0.1	0.2	0.5
ϵ	0.0	0.0	0.0	0.0	0.0
	0.1	0.47	0.21	0.1	0.0
	0.2	0.72	0.36	0.18	0.0
	0.5	1.0	0.53	0.28	0.0

Wir erhalten Kurven analog Werner (2008)[1]. Die Kurven entsprechen im Prinzip der bereits bekannten Kurve der maximalen Entropie: Die Entropie wird maximal bei Gleichverteilung der Zeichenwahrscheinlichkeit. Diese Kurve wird nun mit zunehmender Fehlerwahrscheinlichkeit gedämpft.



[1] Werner, M. (2008). Information und Codierung: Grundlagen und Anwendungen. Vieweg + Teubner, S.94.

Übung 10

Kanalcodierung und Blockcodes
Theorie, Grenzen und Konstruktion
Lösung

V1.0

Hinweise:

- Übungen sind mit Vorteil alleine zu lösen
- Benutzen Sie die Musterlösung nur zur Korrektur
- Die Übungen sind wichtige Vorbereitungen für die Prüfung. Lösen Sie die Übungen sorgfältig und stellen Sie die Lösungswege übersichtlich dar.
- (Ergänzte) Vorlesungsunterlagen und Fachbücher helfen beim Lösen von Übungen und bringen gleichzeitig eine erweiterte Ansicht auf die Problemstellung.
- Wenn Sie die Übungen nicht verstehen, fragen Sie!

OST Ostschweizer Fachhochschule

Aufgabe 1 Hammingdistanz

Wie gross muss die Hammingdistanz eines Codes mindestens sein, wenn 5 Fehler sicher erkannt werden sollen?

Die Anzahl der sichere erkennbaren Fehler ist $e* = h - 1$. Daraus kann abgeleitet werden, dass

$$h = e^* + 1 = 5 + 1 = 6$$

Aufgabe 2 Hammingdistanz

Gegeben sind die Codeworte des folgenden Codes:

Codewort	Zeichen
00000000	A
11000000	B
10001100	C
01010000	D
01010101	E
10000110	F
11111111	G

Wie gross ist die Hammingdistanz dieses Codes?

Codewort	Zeichen
00000000	A
11000000	B
01010**0**0**0**	D
01010**1**0**1**	E
1000**1**1**0**0	C
1000**0**1**1**0	F
11111111	G

Die kleinste Distanz zwischen zwei Codeworten ist h = 2.

Aufgabe 3 Blockcode

Gegeben ist ein Code mit m = 10 Nachrichtenstellen und k = 5 Kontrollstellen und einer Hammingdistanz von h = 4.

1) Geben Sie die Anzahl der gültigen sowie die Anzahl der möglichen Codeworte an.

Die Anzahl der gültigen Codeworte ist $2^m = 2^{10} = 1024$. Mögliche Codeworte gibt es $2^{m+k} = 2^{15} = 32768$.

2) Ermitteln Sie die sicher erkennbare Fehlerzahl sowie die sicher korrigierbare Fehlerzahl.

Gemäss bekannter Formel ist die Anzahl der sicher erkennbaren Fehler $e^* = h - 1 = 4 - 1 = 3$. Die Anzahl der sicher korrigierbaren Fehler ist $e = \frac{h-2}{2} = \frac{4-2}{2} = 1$, weil h in diesem Fall gerade ist.

3) Was geschieht, wenn mehr als die Anzahl der sicher korrigierbaren Fehler e in einem Codewort auftreten?

In diesem Fall wird entweder falsch korrigiert oder der Fehler wird nicht erkannt.

4) Ist der Code dichtgepackt?

Der Code ist nicht dichtgepackt, weil die Hammingdistanz $h = 4$ ist. Es befinden sich deshalb nicht alle Codeworte in einer Korrigierkugel.

Alternativ mit Rechnung:

$$2^m \cdot \sum_{w=0}^{e} \binom{n}{w} = 2^{10} \cdot \sum_{w=0}^{1} \binom{15}{w} = 2^{10} \cdot \left(\binom{15}{0} + \binom{15}{1} \right)$$

$$= 2^{10}(1 + 15) = 16 \cdot 2^{10} = 2^{14} < 2^{15}$$

OST Ostschweizer Fachhochschule

Aufgabe 4 Hamming Blockcode

Gegeben sei die folgende Prüfmatrix eines Hamming Blockcodes:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

1) Wie viele Kontrollstellen k besitzt dieser Code?

Bei Hamming Blockcodes markiert die Einheitsmatrix in der Prüfmatrix die Anzahl der Kontrollstellen.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Die Anzahl der Kontrollstellen ist also $k = 4$.

2) Wie viele gültige Codeworte können mit diesem Blockcode gebildet werden?

Die Anzahl der gültigen Codeworte ist $2^m = 2^{11} = 2048$, wobei m sich aus der Anzahl Spalten der Matrix minus der Anzahl Kontrollstellen ergibt.

3) Ermitteln Sie die Kontrollstellen für das Codewort 10110000010.

Die Prüfgleichungen für die Kontrollstellen können aus der Prüfmatrix gelesen werden. In diesem Fall sind das:

$$x_{12} = (x_1 + x_2 + x_3 + x_4 + x_6 + x_7 + x_8) \ mod\ 2$$
$$x_{13} = (x_1 + x_2 + x_3 + x_5 + x_6 + x_9 + x_{10}) \ mod\ 2$$
$$x_{14} = (x_1 + x_2 + x_4 + x_5 + x_7 + x_9 + x_{11}) \ mod\ 2$$
$$x_{15} = (x_1 + x_3 + x_4 + x_5 + x_8 + x_{10} + x_{11}) \ mod\ 2$$

Das Einsetzen der Nachrichtenbits in diese Gleichungen ergibt die 4 Kontrollstellen und somit auch das komplette Codewort:

101100000101100

4) Geben Sie die Fehlersyndrome für die Fälle an, dass …

a) … nur x_2 gestört ist.

$$\begin{bmatrix}1\\1\\1\\0\end{bmatrix}$$

b) … nur x_{11} gestört ist.

$$\begin{bmatrix}0\\0\\1\\1\end{bmatrix}$$

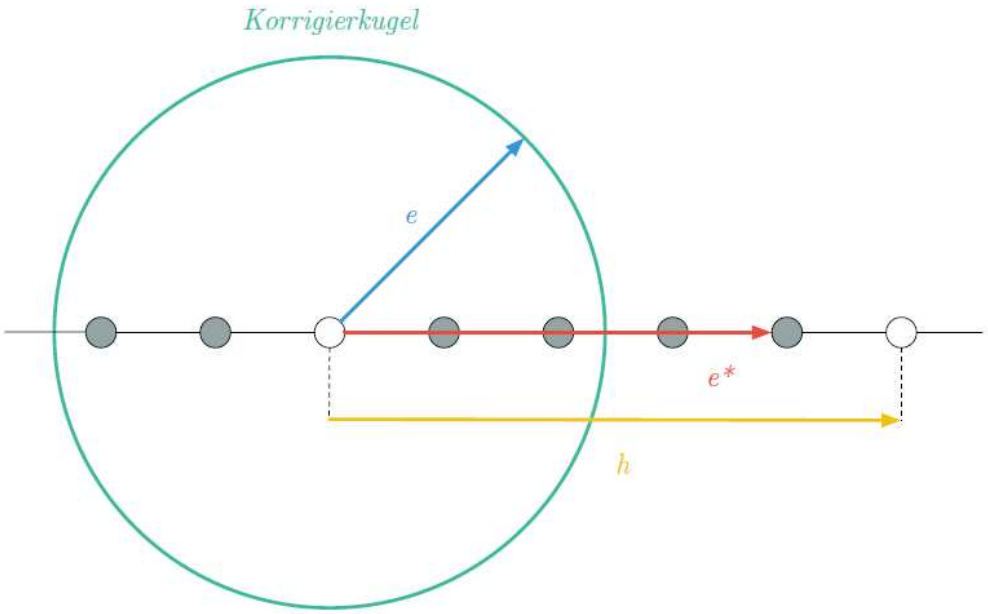
c) … x_2 und x_{11} gestört sind.

$$\begin{bmatrix}1\\1\\0\\1\end{bmatrix}$$

Wie leicht zu sehen ist, entspricht auch dieses Syndrom einer Spalte der Prüfmatrix, nämlich der dritten. Geht man nun davon aus, dass nur ein Fehler passiert ist, so wird man hier eine Falschkorrektur vornehmen (durch flippen des dritten Bits).

Aufgabe 5 Korrigierkugel

Verdeutlichen Sie anhand einer Skizze den Begriff der Korrigierkugel (natürlich ohne die Vorlesungsunterlagen zu Hilfe zu ziehen).



Das Zentrum der Korrigierkugel liegt auf jedem gültigen Codewort (weiss) und schliesst alle umliegenden, ungültigen Codeworte (grau), die sicher korrigiert werden können, ein. Die Anzahl der sicher korrigierbaren Fehler e kann deshalb als Radius der Korrigierkugel betrachtet werden. Die Hammingdistanz h ist die kürzeste Distanz zwischen zwei gültigen Codeworten unter Betrachtung aller Paare von gültigen Codeworten im gesamten Coderaum. Die Anzahl der sicher erkennbaren Fehler e^* ist deshalb h − 1.

Aufgabe 6 Parity Blockcode

Gegeben ist ein Blockcode, der für die Nachrichtenstellen x_{ij} ein Längs- und Quersummen-Paritybit p einfügt. Dies ist in der untenstehenden Abbildung visualisiert.

x_{11}	x_{12}	x_{13}	$\cdots$	x_{1n}	$p_{1(n+1)}$
x_{21}	x_{22}	x_{23}	$\cdots$	x_{2n}	$p_{2(n+1)}$
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
x_{k1}	x_{k2}	x_{k3}	$\cdots$	x_{kn}	$p_{k(n+1)}$
$p_{(k+1)1}$	$p_{(k+1)2}$	$p_{(k+1)3}$	$\cdots$	$p_{(k+1)n}$	$p_{(k+1)(n+1)}$

1) Wie gross ist die Zahl der sicher erkennbaren und der sicher korrigierbaren Fehler?

Obwohl k und n beliebig gross gewählt werden können, ist die Zahl der sicher erkennbaren und sicher korrigierbaren Fehler konstant. Mit der obigen Wahl an Paritybits können 3 Fehler sicher erkannt werden. Ein einzelner Fehler kann sicher korrigiert werden. Um das besser verstehen zu können, ist der Fall, bei dem 4 Fehler eine Erkennung verunmöglichen, in der nachfolgenden Tabelle dargestellt.

x_{11}	x_{12}	x_{13}	$\cdots$	x_{1n}	$p_{1(n+1)}$
x_{21}	x_{22}	x_{23}	$\cdots$	x_{2n}	$p_{2(n+1)}$
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
x_{k1}	x_{k2}	x_{k3}	$\cdots$	x_{kn}	$p_{k(n+1)}$
$p_{(k+1)1}$	$p_{(k+1)2}$	$p_{(k+1)3}$	$\cdots$	$p_{(k+1)n}$	$p_{(k+1)(n+1)}$

2) Wie gross ist die Hammingdistanz?

Die Hammingdistanz ist um 1 grösser als die Anzahl der sicher erkennbaren Fehler. Also $h = 4$.

3) Ermitteln Sie die Anzahl der Prüfstellen als Funktion der Nutzdatenmenge f(m). Wie wächst die Anzahl der Prüfstellen? Zur Vereinfachung der Aufgabe dürfen Sie annehmen, dass der Nachrichtenblock quadratisch ist und somit k ≈ n gilt. Das mindert zwar die Genauigkeit der Berechnung, ist für unsere Zwecke aber völlig ausreichend.

Wie gehabt hat der Blockcode k Zeilen und n Spalten. Die Anzahl der Nachrichtenwerte m kann also wie folgt ausgedrückt werden:

$$m = k \cdot n$$

Geht man davon aus, dass die Anzahl der Zeilen ungefähr gleich der Anzahl der Spalten ist, kann man folgendes annehmen:

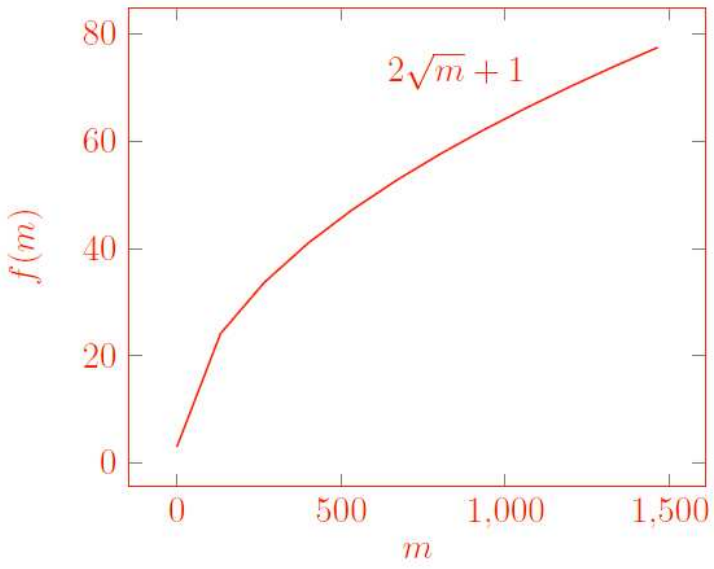
$$k \approx n \approx \sqrt{m}$$

Die Zahl der Prüfstellen (Paritybits) berechnet sich dann wie folgt:

$$k + n + 1 \approx 2\sqrt{m} + 1$$

Die Funktion ist also definiert als:

$$f(m) = 2\sqrt{m} + 1$$



Aufgabe 7 Hamming Blockcode

Sie haben die Aufgabe, die Übertragung von Messwerten abzusichern. Es werden insgesamt 32 Messwerte unterschieden. Bestimmen Sie, was für einen Hamming Blockcode sie dazu benötigen. Zeigen Sie, dass Sie eine Fehlerstelle sicher erkennen können. Prüfen Sie, ob der Code «dichtgepackt» ist und interpretieren Sie das Ergebnis.

Um 32 Messwerte unterscheiden zu können, werden 5 Bit benötigt ($2^5 = 32$). Die Anzahl der Nachrichtenbits m muss also mindestens gleich 5 sein. Einen Hamming Blockcode mit dieser Anzahl Nachrichtenstellen gibt es aber nicht. Den nächstgrösseren Hamming Blockcode findet man, indem man die Formel $m = 2^k - k - 1$ anwendet. Wir suchen also den kleinsten Wert für k, bei dem m grösser oder gleich 5 wird:

$$\begin{aligned} k = 2: &\qquad m = 2^2 - 2 - 1 = 1 &\qquad (< 5)\\ k = 3: &\qquad m = 2^3 - 3 - 1 = 4 &\qquad (< 5)\\ k = 4: &\qquad m = 2^4 - 4 - 1 = 11 &\qquad (> 5) \end{aligned}$$

Es werden also 4 Kontrollstellen benötigt, um den gewünschten Hamming Blockcode zu konstruieren. Die Blocklänge kann dann einfach berechnet werden aus der Formel $n = m + k$. Für $k = 4$ ergibt das eine Blocklänge von $n = 11 + 4 = 15$. Sie benötigen also einen *(11, 4)-Hamming-Blockcode*. Ein Beispiel eines solchen Hamming Blockcodes wurde bereits in der letzten Aufgabe bearbeitet.

Die Hammingdistanz für diesen Code ist $h = 3$, wie das für alle Hamming Blockcodes der Fall ist. Die Anzahl der sicher erkennbaren Fehler ist damit $e = \frac{h-1}{2} = \frac{3-1}{2} = 1$.

Ausserdem gilt:

$$2^m \cdot \sum_{w=0}^{e} \binom{n}{w} = 2^{11} \cdot \sum_{w=0}^{1} \binom{15}{w} = 2^{11} \cdot \left(\binom{15}{0} + \binom{15}{1} \right) = 32768 = 2^{15} = 2^n$$

Der Code ist also dichtgepackt.

Merke: Mit diesem Code können mehr als nur 32 verschiedene Nachrichten übertragen werden. Um genau zu sein, sind es $2^{11} = 2048$ Nachrichten. Das ist aber nicht weiter schlimm. Man muss einfach ein Mapping der gewünschten Nachrichten auf gültige Codeworte vorgeben.

Aufgabe 8 Parity Blockcode

Vergleichen Sie die Effizienz des Parity Blockcodes aus Aufgabe 8 mit dem Hamming Blockcode. Auch hier dürfen Sie zur Vereinfachung eine Näherungsrechnung machen. Überlegen Sie, wie Sie das am besten machen können.

Wie schon in Aufgabe 8 macht es Sinn, die Anzahl der Prüfstellen als eine Funktion der Nachrichtenstellen $f(m)$ anzunähern. So kann schliesslich der Plot der beiden Ansätze direkt verglichen werden. Es gilt:

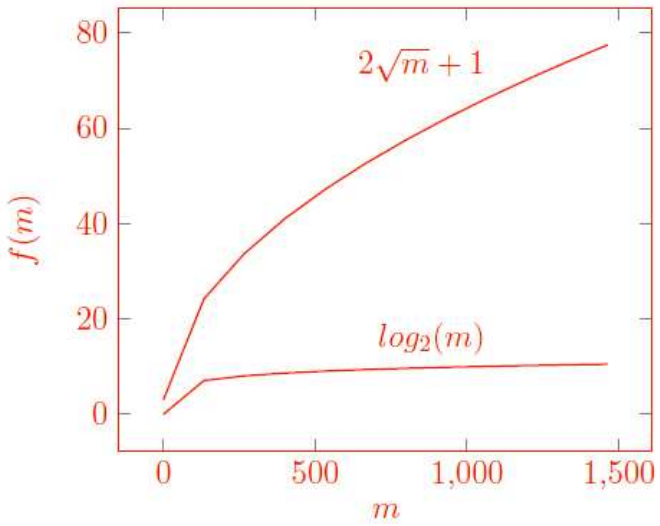
$$n = 2^k - 1$$

und

$$n = k + m$$

Somit kann folgende Näherungsrechnung durchgeführt werden:

$$\begin{aligned} 2^k - 1 &= k + m \\ 2^k - (1 + k) &= m \\ 2^k &\approx m \\ f(m) = k &\approx log_2(m) \end{aligned}$$



Übung 11

Zyklische Codes
Algebraische Konstruktion von Fehlererkennung
Lösung

V1.0

Hinweise:

- Übungen sind mit Vorteil alleine zu lösen
- Benutzen Sie die Musterlösung nur zur Korrektur
- Die Übungen sind wichtige Vorbereitungen für die Prüfung. Lösen Sie die Übungen sorgfältig und stellen Sie die Lösungswege übersichtlich dar.
- (Ergänzte) Vorlesungsunterlagen und Fachbücher helfen beim Lösen von Übungen und bringen gleichzeitig eine erweiterte Ansicht auf die Problemstellung.
- Wenn Sie die Übungen nicht verstehen, fragen Sie!

OST Ostschweizer Fachhochschule

Aufgabe 1 Polynomrechnungen in $\mathbb{Z}_2$

a) Gegeben sind folgende zwei Polynome
$A(x)=u^4+u^2+u+1$
$B(x)=u^3+u^1+u^0$

1) Addieren Sie die zwei Polynome A(x) und B(x). Interpretieren Sie das Resultat als Codewort.

$(u^4+u^2+u+1) + (u^3+u^1+u^0) = (u^4+u^2+u^1+u^0) + (u^3+u^1+u^0) = u^4+u^3+u^2+2u^1+2u^0 = u^4+u^3+u^2$
Codewort: 11100

2) Multiplizieren Sie die zwei Polynome A(x) und B(x)

$(u^4+u^2+u+1) * (u^3+u^1+u^0) = (u^4+u^2+u^1+u^0) * (u^3+u^1+u^0) =$
$u^7+u^5+u^4+u^5+u^3+u^2+u^4+u^2+u^1+u^3+u^1+u^0=u^7+2u^5+2u^4+2u^3+2u^2+2u^1+u^0=u^7+u^0$
Codewort: 10000001

3) Notieren Sie die zwei Polynome als Vektoren und führen Sie die Addition nochmals in dieser Darstellung durch.

$$\begin{pmatrix}1\\0\\1\\1\\1\end{pmatrix} + \begin{pmatrix}0\\1\\0\\1\\1\end{pmatrix} = \begin{pmatrix}1\\1\\1\\0\\0\end{pmatrix}$$

b) Berechnen Sie $\frac{x^6}{x^3+x+1}$

$$\begin{array}{rrrrrrr} x^6 & & & & & & : x^3+x+1 = x^3+x+1 \\ -\ x^6 & +x^4 & +x^3 & & & & \\ \hline & x^4 & +x^3 & & & & \\ -\ & x^4 & & +x^2 & +x & & \\ \hline & & x^3 & +x^2 & +x & & \\ -\ & & x^3 & & +x & +1 & \\ \hline & & & & x^2 & +1 & \end{array}$$

$$\frac{x^6}{x^3+x+1} = (x^3+x+1)\ Rest\ (x^2+1)$$

c) Als Resultat einer Polynomdivision haben Sie (x^4+1) Rest (x+1) erhalten. Was war vermutlich die ursprüngliche Division?

$$(x^4+1)+\frac{x+1}{x^4+1} = \frac{(x^4+1)^2+x+1}{x^4} = \frac{x^8+x}{x^4+1}$$

Aufgabe 2 Erweiterungskörper

Bestimmen Sie die Elemente, die durch das primitive Polynom x^4+x+1 erzeugt werden.

Mit $a^4 + a + 1 = 0$ folgt:

$$
\begin{aligned}
a &= a \\
a^2 &= a^2 \\
a^3 &= a^3 \\
a^4 &= a + 1 \\
a^5 &= a^2 + a \\
a^6 &= a^3 + a^2 \\
a^7 &= a^3 + a + 1 \\
a^8 &= a^2 + 1 \\
a^9 &= a^3 + a \\
a^{10} &= a^2 + a + 1 \\
a^{11} &= a^3 + a^2 + a \\
a^{12} &= a^3 + a^2 + a + 1 \\
a^{13} &= a^3 + a^2 + 1 \\
a^{14} &= a^3 + 1 \\
a^{15} &= 1 \\
a^{16} &= a \\
a^{17} &= a^2 \\
&\ldots
\end{aligned}
$$

Die Zyklusl̈änge ist 15, wir erhalten:

{0010, 0100, 1000, 0011, 0110, 1100, 1011, 0101, 1010, 0111, 1110, 1111, 1101, 1001, 0001}

Aufgabe 3 Reduzible Polynome

Ein Polynom bezeichnet man als reduzibel, wenn es als Produkt zweier Polynome mit jeweils niedrigerem Grad dargestellt werden kann. Sind die folgenden Polynome reduzibel?

a) x^2+x

Ja. $x^2 + x = x(x + 1)$

b) x^2+x+1

Nein. Alle in Frage kommenden Polynomdivision ergeben einen Rest:

$x^2 + x + 1 : x^2 \quad = 1 \text{ Rest } x + 1$
$x^2 + x + 1 : x^2 + x \quad = 1 \text{ Rest } 1$
$x^2 + x + 1 : x^2 + 1 \quad = 1 \text{ Rest } x$

$x^2 + x + 1 : x$ = $x + 1$ Rest 1

$x^2 + x + 1 : x + 1$ = x Rest 1

Aufgabe 4 Reduzible Polynome – mathematische Betrachtungen

Wir betrachten ein Polynom $p(x) = x^3 + a_2x^2 + a_1x + a_0 \; mit \; a_1 \in GF(2)$.

a) Welche Aussage können Sie über a_0 treffen, wenn das Polynom p(x) irreduzibel ist? Begründen Sie Ihre Antwort.

Wenn das Polynom p(x) über dem Körper GF(2) irreduzibel ist, bedeutet das, dass es keine Polynome niedrigeren Grades gibt, die in GF(2) Faktoren von p(x) sind. Das heisst, es kann nicht als Produkt von Polynomen ersten Grades (Linearfaktoren) oder als Produkt eines linearen und eines quadratischen Polynoms in GF(2) ausgedrückt werden.

In GF(2) kann a_0 entweder 0 oder 1 sein. Wenn a_0=0, dann hätte das Polynom p(x) eine Nullstelle bei x=0, da $p(0) = 0^3 + a_2 0^2 + a_1 0 + a_0 = a_0 = 0$. Das würde bedeuten, dass x ein Faktor von p(x) wäre und somit das Polynom reduzierbar wäre.

Damit das Polynom irreduzibel ist, darf es keine Nullstellen in GF(2) haben. Also muss a_0=1 sein, damit p(0)≠0, was keine Nullstelle bei x=0 impliziert. Somit kann man schliessen, dass für ein irreduzibles Polynom dritten Grades über GF(2) der Koeffizient a_0 immer 1 sein muss.

b) Nennen Sie einen Grund, warum das Polynom p(x) reduzierbar (d.h. nicht irreduzibel) ist, wenn

$$\sum_{i=0}^{2} a_i \equiv 1 \; (mod \; 2)$$

$\sum_{i=0}^{2} a_i \equiv 1 \; (mod \; 2)$ bedeutet, dass wenn die Koeffizienten a_0, a_1 und a_2 addiert werden, das Ergebnis modulo 2 gleich 1 sein wird. In GF(2) bedeutet das, dass eine ungerade Anzahl von diesen Koeffizienten gleich 1 sein muss, weil die Summe von zwei Einsen in GF(2) wieder 0 ergibt (1 + 1 = 0 mod 2) und die Summe von einer Eins bleibt eine Eins (0 + 1 = 1 mod 2).

Wenn wir nun die Werte dieser Koeffizienten in das Polynom einsetzen und x=1 ist, dann vereinfacht sich p(1) zu: $p(1)=1^3+a_2 1^2+a_1 1^1+a_0=1+a_2+a_1+a_0$. Wenn die Summe der Koeffizienten a_0, a_1 und a_2 gleich 1 ist, dann bedeutet das für p(1)=1+1=0 (in GF(2)).

Da p(1)=0, wissen wir, dass 1 eine Wurzel des Polynoms ist. Dies wiederum bedeutet, dass p(x) durch x−1 teilbar ist (Faktorsatz, da wenn x=1 somit x-1=0), aber da in GF(2) das Konzept von negativen Zahlen nicht existiert und −1 = 1 ist, können wir sagen, dass p(x) durch x+1 teilbar ist und entsprechend reduzierbar ist.

Wenn wir zur Prüfung die Division $(x^3+x^2+x+1) : (x+1)$ durchführen, erhalten wir x^2+1 ohne Rest.

Aufgabe 5 Zyklischer Hammingcode

Gegeben ist ein zyklischer Hammingcode mit dem Generatorpolynom $g(x) = 1 + x + x^3$.

1) Ermitteln Sie alle gültigen Codeworte durch die schnelle Mehrfachaddition.

Der Grad des Generatorpolynoms entspricht der Anzahl der Kontrollstellen $k = 3$. Ausserdem gilt, dass $n = 2^k - 1$ und $n = m + k$. Somit ist die Länge der Codeworte $n = 7$ und die Anzahl Nachrichtenstellen $m = 4$. Das Generatorpolynom entspricht dem Vektor $[1 \quad 0 \quad 1 \quad 1]$.

Wir erhalten zum Beispiel die folgenden Codeworte:

$$\begin{array}{cccc|ccc} 0 & 0 & 0 & 1 & & & \\ & & & 1 & 0 & 1 & 1 \\ \hline & & & 0 & 0 & 1 & 1 \end{array}$$

$$\begin{array}{cccc|ccc} 0 & 0 & 1 & 0 & & & \\ & & 1 & 0 & 1 & 1 & \\ \hline & & 0 & 0 & 1 & 1 & 0 \end{array}$$

$$\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & & & \\ 1 & 0 & 1 & 1 & & & \\ \hline & & 1 & 1 & & & \\ & & 1 & 0 & 1 & 1 & \\ \hline & & 0 & 1 & 1 & 1 & \\ & & & 1 & 0 & 1 & 1 \\ \hline & & & 0 & 1 & 0 & 1 \end{array}$$

$$\begin{array}{cccc|ccc} 1 & 0 & 1 & 0 & & & \\ 1 & 0 & 1 & 1 & & & \\ \hline & & & 1 & & & \\ & & & 1 & 0 & 1 & 1 \\ \hline & & & 0 & 0 & 1 & 1 \end{array}$$

Alle gültigen Codeworte sind die folgenden:

0000000	0100111	1000101	1100010
0001011	0101100	1001110	1101001
0010110	0110001	1010011	1110100
0011101	0111010	1011000	1111111

2) Ermitteln Sie die Syndrome für jeweils eine Fehlerstelle.

Vorgehen: Bauen Sie bei einem gültigen Codewort an der 1. Stelle einen Fehler ein und berechnen Sie den Rest, der bei der Division durch das Generatorpolynom entsteht. Wiederholen Sie diesen Vorgang mit jeder Stelle des Codeworts. So erhalten Sie die 7 gesuchten Syndrome.

Wir erhalten zum Beispiel für das Codewort 0000000 für den Fehler an erster Stelle x^6 den folgenden Rest:

$$
\begin{array}{rlll}
 & x^6 & & : x^3+x+1 = x^3+x+1 \text{ Rest } \boldsymbol{x^2+1} \\
- & x^6 & +x^4 & +x^3 \\
\hline
 & & x^4 & +x^3 \\
 & - & x^4 & +x^2 \quad +x \\
\hline
 & & x^3 & +x^2 \quad +x \\
 & - & x^3 & +x \quad +1 \\
\hline
 & & & x^2 \quad +1
\end{array}
$$

Und damit das Syndrom x^2+1 respektive $\begin{bmatrix}1 & 0 & 1\end{bmatrix}$.

Für die anderen Fehlerstellen erhalten wir

Fehler	Resultat	
x^5	x^2+1 Rest x^2+x+1	$\begin{bmatrix}1 & 1 & 1\end{bmatrix}$
x^4	x Rest x^2+x	$\begin{bmatrix}1 & 1 & 0\end{bmatrix}$
x^3	1 Rest $x+1$	$\begin{bmatrix}0 & 1 & 1\end{bmatrix}$
x^2	Rest x^2	$\begin{bmatrix}1 & 0 & 0\end{bmatrix}$
x	Rest x	$\begin{bmatrix}0 & 1 & 0\end{bmatrix}$
1	Rest 1	$\begin{bmatrix}0 & 0 & 1\end{bmatrix}$

3) Entwickeln Sie die zu diesem Code passende Prüfmatrix des Hamming Blockcodes.

Jedes Fehlersyndrom stellt den Vektor an dessen Stelle für die Prüfmatrix. Diese ist somit:

$$
\begin{bmatrix}
1 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 & 1
\end{bmatrix}
$$

Aufgabe 6 CRC-Code

Gegeben ist das primitive Polynom $1 + x + x^4$.

1) Konstruieren Sie mit diesem Polynom einen CRC-Code.

Um einen CRC-Code zu erhalten, muss das primitive Polynom mit dem Term $(1 + x)$ multipliziert $(mod\ 2)$ werden:

$$(1 + x + x^4) \cdot (1 + x) = 1 + x^2 + x^4 + x^5$$

2) Ermitteln Sie das CRC Generatorpolynom.

Das Generatorpolynom entspricht dem obigen Ergebnis, also $g(x) = 1 + x^2 + x^4 + x^5$. In Bits kann dieses auch als 110101 dargestellt werden.

3) Wie viele Kontrollstellen besitzt ein Codewort dieses CRC?

Der Grad des Generatorpolynoms entspricht der Anzahl der Kontrollstellen $k = 5$.

4) Wie gross ist die Hammingdistanz dieses Codes?

Abramson/CRC-Codes haben eine Hammingdistanz von $h = 4$ (durch die Multiplikation 1 mehr als die zyklischen Hamming-Codes).

5) Ermitteln Sie einige gültige Codeworte mit der schnellen Mehrfachaddition. Wie viele gültige Codeworte können insgesamt gebildet werden?

Beispiele von gültigen Codeworten sind:

000000000000000

101101011110000

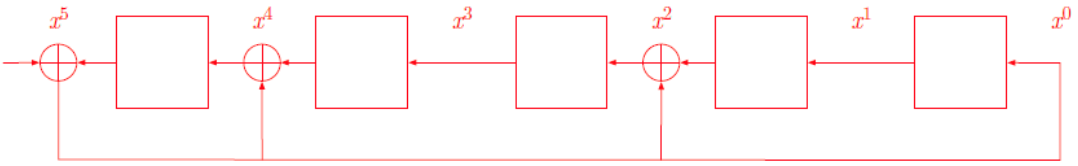
100010111011111

Gültige Codeworte insgesamt: 2^{10}

Mögliche Codeworte insgesamt: 2^{15}

Abramson Code: $n = 2^{k-1} - 1 = 2^4 - 1 = 15$

6) Zeichnen Sie das rückgekoppelte Schieberegister, mit dem Sie die Kontrollstellen ermitteln können.



Übung 12

Faltungscodes - Fortlaufende Fehlererkennung und Korrektur

Lösung

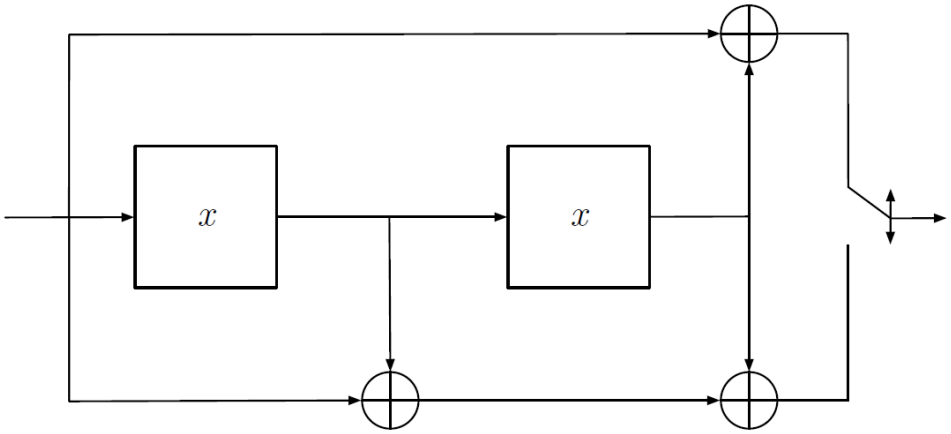
V1.0

Hinweise:

- Übungen sind mit Vorteil alleine zu lösen
- Benutzen Sie die Musterlösung nur zur Korrektur
- Die Übungen sind wichtige Vorbereitungen für die Prüfung. Lösen Sie die Übungen sorgfältig und stellen Sie die Lösungswege übersichtlich dar.
- (Ergänzte) Vorlesungsunterlagen und Fachbücher helfen beim Lösen von Übungen und bringen gleichzeitig eine erweiterte Ansicht auf die Problemstellung.
- Wenn Sie die Übungen nicht verstehen, fragen Sie!

Aufgabe 1 Faltungscode

Gegeben ist die folgende Encoder-Schaltung eines Faltungscoders:



1) Wie lauten die Generatorpolynome?

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

2) Skizzieren Sie das Zustandsdiagramm

Am einfachsten ist es, wenn wir zuerst eine Tabelle mit den Eingangsbits und den Zuständen erstellen.

Eingang	Schieberegister		Zustand
0	0	0	S_0
1	0	0	
0	1	0	S_1
1	1	0	
0	0	1	S_2
1	0	1	
0	1	1	S_3
1	1	1	

Nun können wir die Ausgangsbits bestimmen:

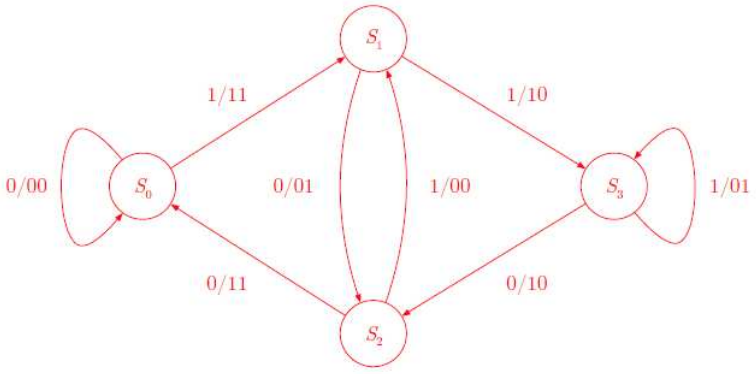
- Für das erste Ausgangsbit (g_1) addieren wir (mod 2), das Eingangsbit und das zweite Schieberegisterbit (erste und dritte Spalte).
- Für das zweie Ausgangsbit (g_2) addieren wir (mod 2) das Eingangsbit und beide Schieberegisterbits (erste drei Spalten).

Eingang	Schieberegister		Zustand	Ausgang	
0	0	0	S_0	0	0
1	0	0		1	1
0	1	0	S_1	0	1
1	1	0		1	0
0	0	1	S_2	1	1
1	0	1		0	0
0	1	1	S_3	1	0
1	1	1		0	1

Nun bestimmen wir den nächsten Zustand durch Schieben der Schieberegister: Es wird das Eingangsbit und das erste Schieberegisterbit übernommen (zweite und dritte Spalte).

Eingang	Zustand	Schieberegister		Ausgang		Nächster Zustand
0	S_0	0	0	0	0	S_0
1		0	0	1	1	S_1
0	S_1	1	0	0	1	S_2
1		1	0	1	0	S_3
0	S_2	0	1	1	1	S_0
1		0	1	0	0	S_1
0	S_3	1	1	1	0	S_2
1		1	1	0	1	S_3

Nun zeichnen wir das Zustandsdiagramm.



3) Codieren Sie die Nachrichtenfolge {u[n]} = {1, 0, 0, 1, 1, 0, 1}. Geben Sie zunächst die Abfolge der Zustände an. Ergänzen Sie die Nachricht so, dass die Übertragung im Nullzustand endet.

Zustandsfolge:

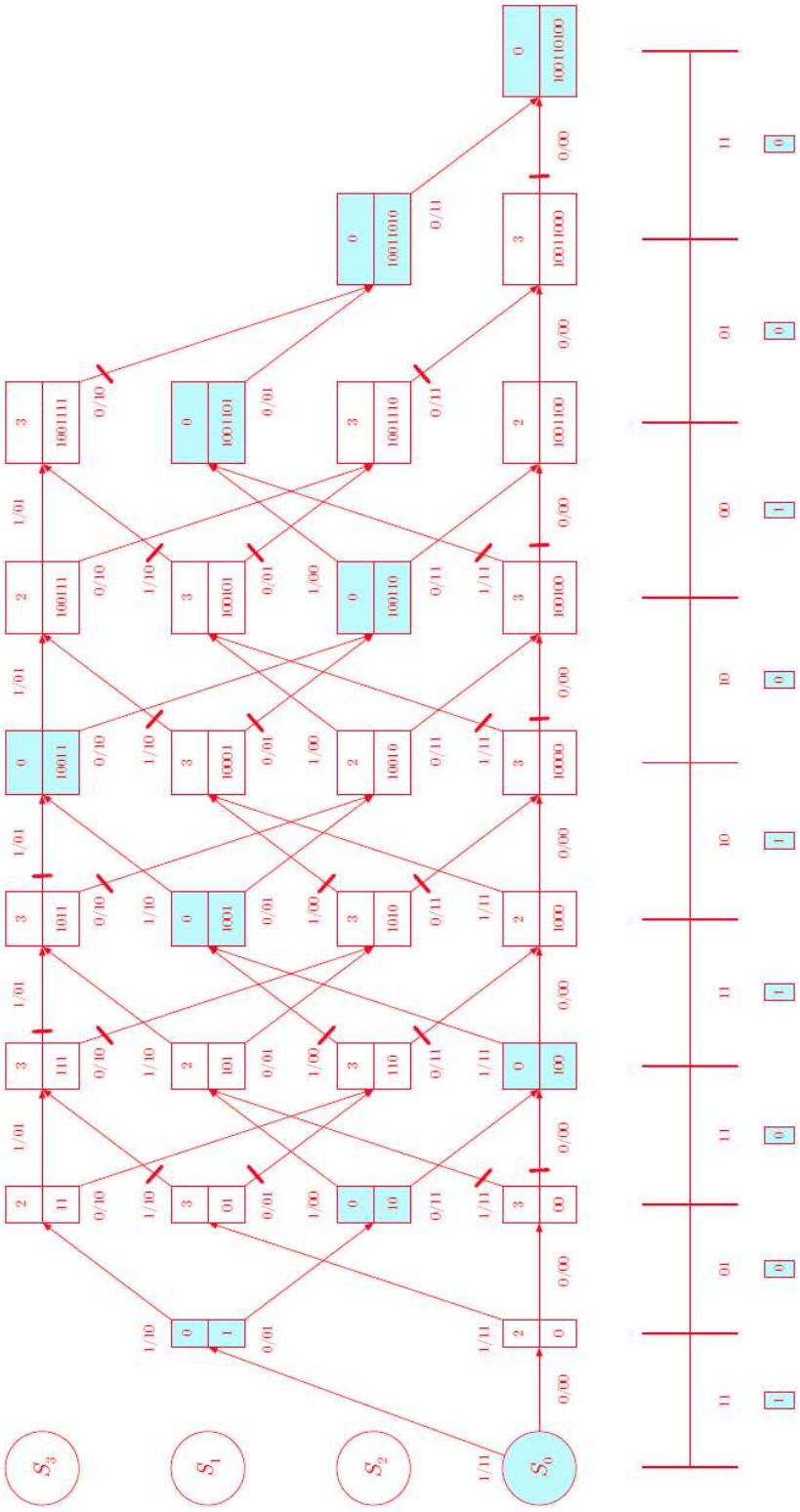
$$(\text{Start } S_0) \rightarrow S_1 \rightarrow S_2 \rightarrow S_0 \rightarrow S_1 \rightarrow S_3 \rightarrow S_2 \rightarrow S_1 (\rightarrow S_2 \rightarrow S_0 \text{ Tail})$$

Codewort:

$$\{v[n]\} = \{11, 01, 11, 11, 10, 10, 00, 01, 11\}$$

4) Es wird die Folge {r[n]} = {11, 01, 11, 11, 10, 10, 00, 01, 11} empfangen. Decodieren Sie die Folge mit Hilfe des Netzdiagramms. Die Codierung beginnt und endet im Nullzustand.

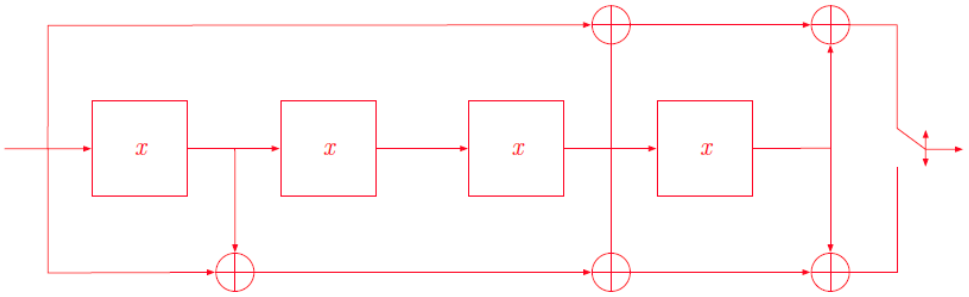
Die ursprüngliche Nachricht ist $\{u'[n]\} = \{1, 0, 0, 1, 1, 0, 1\}$. Merke: Hier sind die Tailbits bereits wieder entfernt worden.



Aufgabe 2 GSM Faltungscode

Bei der GSM Sprachübertragung im Fullrate Modus wird ein speziell optimierter Faltungscode mit den Generatorpolynomen $g_1(x) = 1 + x^3 + x^4$ und $g_2(x) = 1 + x + x^3 + x^4$ benutzt.

1) Skizzieren Sie die Encoder-Schaltung.



2) Wie viele Tailbits müssen einer Nachrichtenfolge angefügt werden?

Es müssen 4 Tailbits angefügt werden.

3) Geben Sie das Encodergedächtnis m und die Block-Coderate R an. Die Block-Coderate ist das Verhältnis der Anzahl codierter Bits zu den gesendeten Bits. Es werden 185 Bits codiert.

Encodergedächtnis:

$$m = 4$$

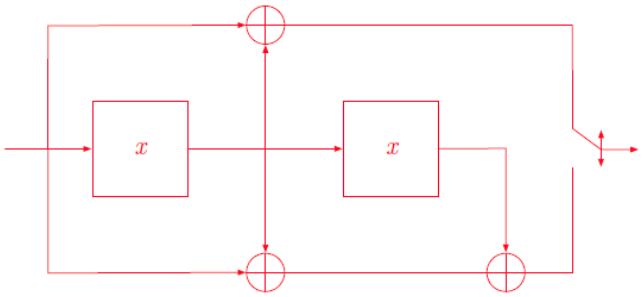
Block-Coderate:

$$R = \frac{185}{2 \cdot (185 + 4)} = 0.489$$

Aufgabe 3 Faltungscode

Gegeben Seien die folgenden Impulsantworten eines Faltungscodierers mit einem Eingang:
{g1} = {1, 1, 0}; {g2}={1, 1, 1}

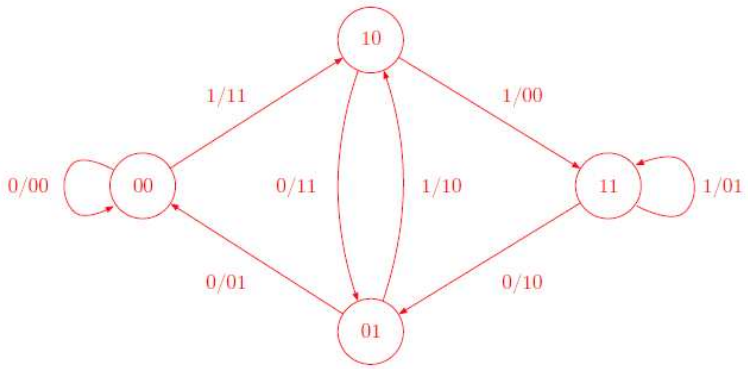
1) Zeichnen Sie die Encoder-Schaltung.



2) Wie viele Tailbits müssen einer Nachrichtenfolge angefügt werden?

Es müssen 2 Tailbits angefügt werden, damit die 2 Speicherstellen sicher wieder mit Nullen belegt sind.

3) Geben Sie das Encoder-Zustandsdiagramm an.

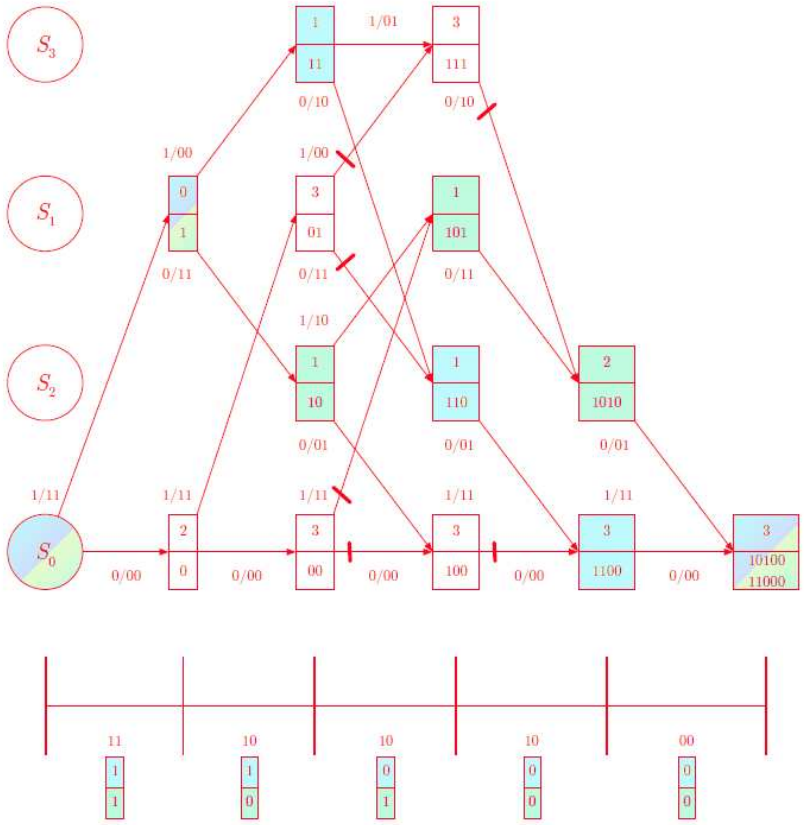


4) Handelt es sich um einen «guten Code»? Begründen Sie.

Ja, es handelt sich um einen guten Code, weil der Unterschied der Ausgabe bei einem Zustandsübergang immer maximal ist.

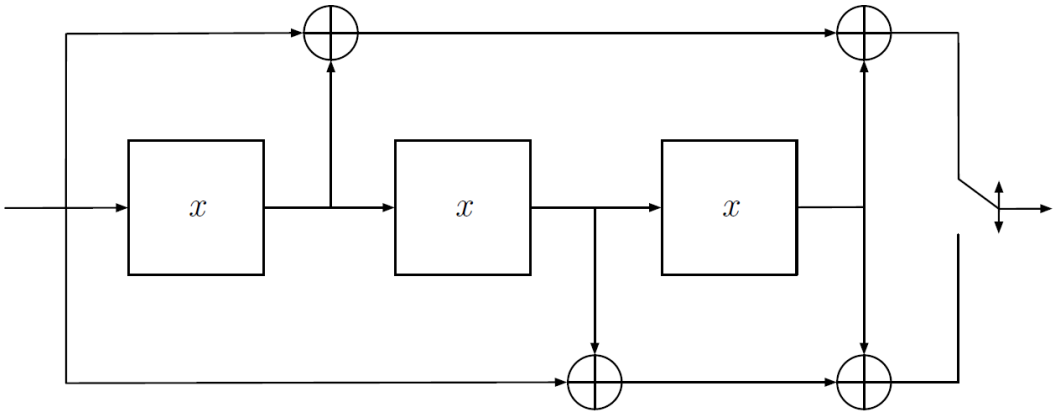
5) Dekodieren Sie die Zeichenfolge Z = {11, 10, 10, 10, 00}

Es gibt zwei Wege, die zum gleichen Gewicht führen. Man kann deshalb nicht sagen, ob $\{u'[n]\} = \{1, 1, 0\}$ oder $\{u'[n]\} = \{1, 0, 1\}$ war. Merke: Die Tailbits wurden hier bereits entfernt.



Aufgabe 4 Faltungscode

Gegeben ist der folgende Faltungscodierer:



1) Wie viele Bits werden zur Berechnung der Ausgangsbits herangezogen?

Insgesamt werden 4 Bits herangezogen, nämlich das aktuelle Bit und 3 weitere Bits aus dem Speicher.

2) Bestimmen Sie die Impulsantwort der Decoderschaltung.

Um die Impulsantwort für die Decoderschaltung zu bestimmen, wird die folgende Nachrichtenfolge als Input für den Faltungscodierer verwendet:

$$\{u[n]\} = \{1, 0, 0, 0\}$$

Die Impulsantwort ist dann:

$$\{v[n]\} = \{11, 10, 01, 11\}$$

3) Geben Sie die Codes in Polynomdarstellung an.

$$g_1(x) = 1 + x + x^3$$
$$g_2(x) = 1 + x^2 + x^3$$

4) Wie viele Zustände hat der Coder?

Der Coder hat insgesamt $2^3 = 8$ Zustände.