Digitaltechnik Aufarbeitung

Kajetan Weiß

24. Februar 2014

Versionstabelle

Jahr-Monat-Tag	Kommentar
2014-01-28	Link zu „FloatConverter“-Website eingefügt. Sektion zu 'Mittlere Codewortlänge von Optimalcodes' eingefügt.
2014-01-24	Korrekturen und Ergänzungen im Kapitel 'Informationstheorie'. Ergänzung Übertragungsmatrix Ü im Kapitel 'Informationstheorie'. Verbesserung der Abbildung zur Symmetrischen Störung im Kapitel 'Informationstheorie'. Erläuterung zur Berechnung der Entropie der Senke hinzugefügt. Verbundmatrix ausführlicher beschrieben.
2014-01-21	Korrektur einer doppelten Bildreferenz mit dem Label `AD-Wandl_unvollst_Sig-Umf_ohne_delta`. Verbesserung der Versionstabelle mittels package `tabularx` und Angabe des Datums zur Versionierung. Abschnitt zum Umgang mit Perioden hinzugefügt. Stellenverschiebung und Zählfunktion als separate `section` deklariert um einfache Referenzierung zu ermöglichen. `chapter` „Darstellung und Arithmetik rationaler Zahlen“ in `section` umgewandelt. Praeambel ausgegliedert. Klickbare Links und PDF-Inhaltsverzeichnis mittels `hyperref` hinzugefügt. Begriffserklärung zu *eineindeutig* hinzugefügt. Abschnitt zu Hamming Codes aktualisiert. Bezeichner im Abschnitt zu R_a und R_r verbessert.
2013-12-24	Verbesserung in Zahlensysteme erster Absatz
2013-10-28	Erste finale Version *YAY*

Inhaltsverzeichnis

Vorwort

Schön, dass Du Dich entschieden hast mit dieser Arbeit zu lernen. Vorweg möchte ich raten, nicht nur die Lektüre zu lesen, sondern zum besseren Verständnis, parallel zu jedem abgeschlossenen Kapitel, die jeweiligen Übungsaufgaben zu erledigen. Wenn Du so vorgehst, merkst du schnell, ob Du alles richtig verstanden hast oder Du Dich noch intensiver mit dem Thema befassen musst. Manchmal kann es sich lohnen, nach einem Abschnitt direkt mit den Übungsaufgaben zu beginnen und erst weiter zu lesen, wenn ein neuer Aufgabentyp eines Kapitels gestellt wird.

Außerdem möchte ich hier vorweg darauf hinweisen, dass ich keinerlei Garantien auf Korrektheit oder Vollständigkeit übernehme. Du kannst mir sehr gerne Fehler berichten oder mich auf Unvollständigkeiten hinweisen. Fehler, seien es inhaltliche, grammatikalische oder Rechtschreibfehler, werde ich umgehend korrigieren. Ergänzungen werde ich vornehmen, sofern ich die Zeit dafür erübrigen kann.

Nach Abschluss der Arbeit sind einige TODOs offen geblieben. Im Quelltext sind die entsprechenden Stellen mit „%TODO“ markiert. Wenn Du Interesse hast, kannst Du Dich gerne um die TODOs kümmern und mir die Ergänzungen zuschicken.

Dieses Dokument ist auf Grundlage der Vorlesung und dem Skript von Professor Gemmar der Hochschule Trier entstanden. Dieses Dokument ist frei von Rechten Dritter. Alle Abbildungen und Texte wurden von mir, Kajetan Weiß, verfasst. Wenn Du das Dokument für gut genug befindest, würde es mich sehr freuen, wenn Du es anderen zugänglich machen würdest. Zwei Bedingungen zur Verbreitung stelle ich: Erstens, Vorwort und Nachwort müssen erhalten bleiben. Zweitens, wenn Du Änderungen oder Ergänzungen vornimmst, bist Du herzlich dazu eingeladen dies zu tun, stelle bitte an entsprechender Stelle oder am Anfang oder am Ende des Dokuments klar, welche Änderungen oder Ergänzungen Du vorgenommen hast.

Bei Fragen oder Unklarheiten kontaktiere mich bitte per E-Mail oder auf GitHub:
`weissk@hochschule-trier.de`
`https://github.com/LittleEntity/Digitaltechnik`

Jetzt wünsche ich Dir viel Erfolg beim Lernen, verstehen und lösen der Herausforderungen in der Veranstaltung.

Kajetan Weiß, Trier den 24. Februar 2014

1 Digitale Nachrichten und Information

Die Begriffe Nachricht, Signal, Information und Daten werden in der Informatik Fachsprache häufig verwendet. Hier die Definitionen der Begriffe:

- Das Signal ist eine messbare zeitliche Änderung eines Zustands. Es wird zwischen analogen Signalen und diskreten Signalen unterschieden. Werte analoger Signale können oft nahezu beliebig viele Zustände und Zwischenzustände abbilden. Werte diskreter Signale können endlich zählbar vielen Zuständen zugeordnet werden. Statt diskreter Signale und Werte wird hier auch von digitalen[1] Signalen und Werten gesprochen.

  Beispiele für analoge Signale: Spannungssignal, Stromsignal, Tonsignal, Wasserstand, Federstand

  Beispiele für diskrete Signale: Morsesignal, Flaggensignal, Erfassung von High/Low Spannungssignalen

- Die Nachricht besteht aus Symbolen zur Beschreibung von Information. Die Bedeutung der Symbole muss erlernt werden. Die Nachricht ist also konkret im Gegensatz zur abstrakten Information.
- Die Daten (Singular Datum) bestehen ebenfalls aus Symbolen zur Beschreibung von Information. Wird im Kontext von der Verarbeitung von Information gesprochen wird der Begriff Daten verwendet.
- Information und Nachricht sind nicht dasselbe. Denn der Informationsgehalt ist vom Empfänger abhängig. Liest ein Empfänger eine ihm bekannte Nachricht lernt er nichts neues. Der Informationsgehalt ist also bei einer bekannten Nachricht null. Der Informationsgehalt einer Nachricht ist demnach messbar am Grad der Neuigkeit.

Eine Information muss aus einer Nachricht ermittelt werden. Dazu dient die Informationsvorschrift.

1.1 Analog-Digital-Wandlung einer Messgröße (Quantisierung)

Werte analoger Signale werden in der digitalen Datenverarbeitung auf diskrete Signalwerte abgebildet. Dies vereinfacht die Verarbeitung und ist vergleichsweise kostengünstiger.

[1] digital von digitus lat. Finger

Wie beschrieben kann der Wertebereich analoger Signalwerte überabzählbar groß und unendlich sein. Diskrete Signalwerte sind allerdings immer einem abzählbaren und in diesem Fall endlichen Wertebereich zu zuordnen. Um eine Abbildung analoger Werte auf diskrete zu ermöglichen steht ein diskreter Wert stets für ein Intervall analoger Werte. Also kann von einem diskreten Wert nicht mehr auf genau einen analogen Ursprungswert geschlossen werden. Es entsteht ein Genauigkeitsverlust. Die Aufteilung des kontinuierlichen Wertebereichs eines analogen Signals wird *Quantisierung* und der Genaugkeitsverlust wird *prinzipieller Quantisierungsfehler* genannt. Die einzelnen Intervalle werden als *Quanten* bezeichnet.

Um den maximalen Quantisierungsfehler möglichst klein zu halten empfiehlt es sich logischer Weise alle Quanten gleich groß zu halten. Außerdem gilt: je größer die Auflösung, das heißt je mehr Quanten für das Abbilden des analogen Wertebereichs verwendet werden, desto kleiner wird der maximale Quantisierungsfehler.

Zudem ist wegen ungenauen technischen Messungen nicht sicher ob ein analoger Signalwert, der in unmittelbarer Nähe einer Intervallgrenze liegt, dem richtigen Intervall zugeordnet wird. Der so entstehende Fehler wird *technischer Quantisierungsfehler* genannt. Die kritischen Werte werden in *Unsicherheitsintervallen* auf der jeweiligen Grenze von einem Quantum zum nächsten zusammengefasst.

1.1.1 A/D-Wandlung mit vollständigem Signalumfang

Es sei ein analoges Signal s_a mit kleinstem annehmbarem Wert s_{min} und größtem annehmbarem Wert Wert s_{max}. Die Wertemenge des analogen Signals ist also:

$$M_{sa} = \{s_a \in \mathbb{R} \mid s_{min} \leq s_a \leq s_{max}\}$$

Das analoge Signal s_a soll in ein digitales Signal s_d gewandelt werden. Dies geschieht über die Quantisierung, wobei jedes Quantum gleich groß sei. Dies wird äquidistante Quantisierung genannt. Bei einer Wandlung mit vollständigem Signalumfang wird der niedrigste Wert des digitalen Signals auf den niedrigsten Wert des analogen Signals gesetzt und der höchste Wert des digitalen Signals auf den höchsten Wert des analogen Signals. Die Anzahl der digitalen Werte sei N_d.

Für die Anzahl der Quanten N_Q gilt bei Wandlung mit vollständigem Signalumfang[2]:

$$N_Q = N_d - 1$$

Die konstante Quantum-Größe[3] Δs ist demnach:

$$\Delta s = \frac{s_{max} - s_{min}}{N_Q} = \frac{s_{max} - s_{min}}{N_d - 1}$$

[2]Prof. Gemmar benennt die Variable N_Q mit N_I für die Anzahl Intervalle.
[3]Prof. Gemmar nennt die Quantum-Größe Δs Intervallbreite Δs_i.

Die Wertemenge des digitalen Signals ist also:

$$M_{sd} = \{s_{di} \mid (s_{di} = s_{min} + i \cdot \Delta s) \wedge (i \in \mathbb{N}) \wedge (0 \leq i \leq N_d - 1)\}$$

Abbildung 1.1 zeigt schematisch die Abbildung der analogen Werte auf die entsprechenden digitalen Werte mittels der jeweiligen Schwellwerte sw_i. Beispielsweise werden alle analogen Werte zwischen s_{min} und sw_1 auf den digitalen Wert s_{d1} abgebildet und alle analogen Werte zwischen sw_1 und sw_2 auf den digitalen Wert s_{d2}.

Der maximale Quantisierungsfehler lässt sich über den Schwellwert und die Quantum-Größe bestimmen. Um den maximalen Quantisierungsfehler zu minimieren wird der jeweilige Schwellwert eines Quantums auf den Mittelwert des jeweiligen Quantums gesetzt. Formal ergibt sich für die Schwellwertemenge also:

$$M_{sw} = \left\{ sw_i \mid \left(sw_i = s_{min} + \frac{1}{2}\Delta s + i \cdot \Delta s \right) \wedge (i \in \mathbb{N}) \wedge (0 \leq i \leq N_Q - 1) \right\}$$

Der maximale Quantisierungsfehler F_{Qmax} beträgt also wegen der äquidistanten Quantisierung und der, wie angegeben, gesetzten Schwellwerte:

$$F_{Qmax} = \pm\frac{1}{2}\Delta s$$

1.1.2 A/D-Wandlung mit unvollständigem Signalumfang

Als nächstes wird betrachtet wie eine äquidistante Quantisierung vorgenommen werden kann, ohne dass der minimale und maximale Wert vom analogen Signal Teil der Wertemenge vom digitalen Signal sind. Dies ergibt im Vergleich zur A/D-Wandlung mit vollständigem Signalumfang bei gleich bleibender Anzahl digitaler Werte einen Genauigkeitsvorteil, da der Quantisierungsfehler bei dieser Methode kleiner ist. Die jeweiligen digitalen Werte werden auf die jeweiligen Mittelwerte der jeweiligen Quanten gesetzt. So ergibt sich pro Quantum genau ein digitaler Wert auf den alle analogen Werte des Quantums abgebildet werden. Es gilt also für die Anzahl digitaler Werte N_d und die Anzahl Quanten N_Q:

$$N_Q = N_d$$

Die konstante Quantum-Größe Δs ist demnach:

$$\Delta s = \frac{s_{max} - s_{min}}{N_Q} = \frac{s_{max} - s_{min}}{N_d}$$

Die Wertemenge des digitalen Signals ist also:

$$M_{sd} = \left\{ sd_i \mid \left(sd_i = s_{min} + \frac{1}{2}\Delta s + i \cdot \Delta s \right) \wedge (i \in \mathbb{N}) \wedge (0 \leq i \leq N_d - 1) \right\}$$

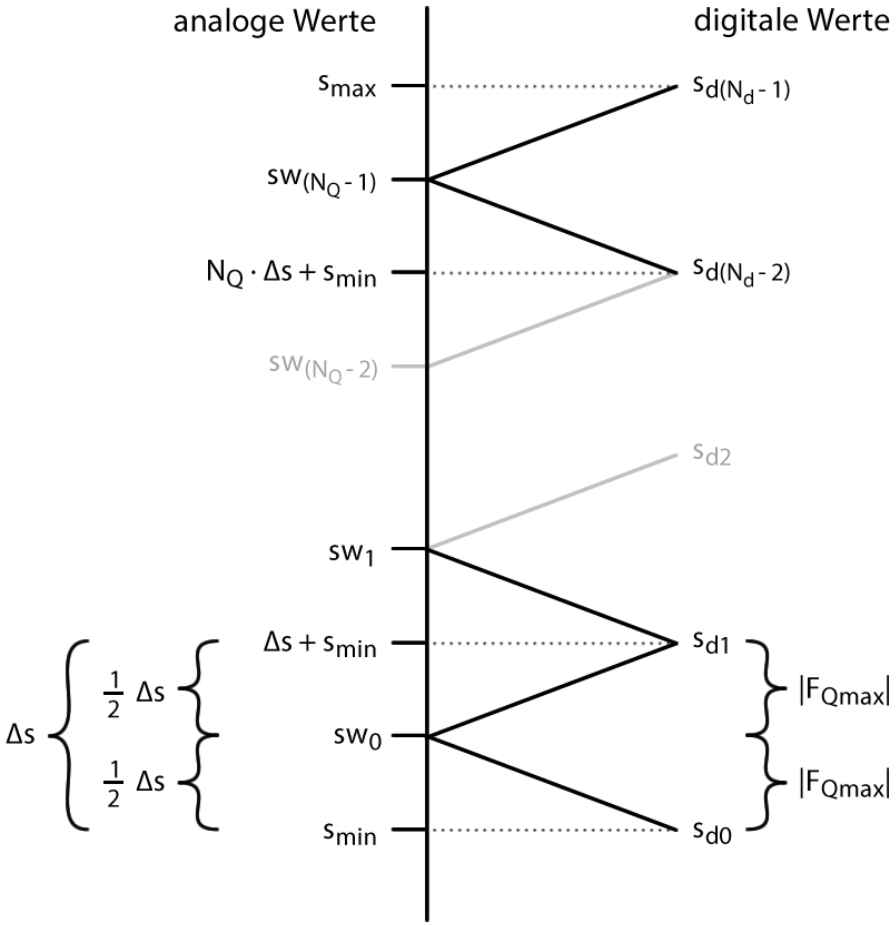


Abbildung 1.1: Analog-Digital-Wandlung mit vollständigem Signalumfang

Für die Menge der Schwellwerte ergibt sich:

$$M_{sw} = \{sw_i \mid (sw_i = s_{min} + i \cdot \Delta s) \wedge (i \in \mathbb{N}) \wedge (1 \leq i \leq N_Q - 1)\}$$

Der maximale Quantisierungsfehler F_{Qmax} beträgt wiederum wegen der äquidistanten Quantisierung und der, wie angegeben, gesetzten Schwellwerte:

$$F_{Qmax} = \pm\frac{1}{2}\Delta s$$

Abbildung 1.2 zeigt schematisch die Abbildung der analogen Werte auf die entsprechenden digitalen Werte mittels der jeweiligen Schwellwerte sw_i. Beispielsweise werden alle analogen Werte zwischen s_{min} und sw_1 auf den digitalen Wert s_{d0} abgebildet. Alle analogen Werte zwischen sw_1 und sw_2 werden auf den digitalen Wert s_{d1} abgebildet.

1.1.3 Der technische Unsicherheitsbereich

Wegen technisch ungenauer Messungen kann ein Wert eines analogen Signals zu hoch oder zu niedrig eingestuft werden. Dies führt zu einer weiteren Fehlerquelle, wenn ein Wert in unmittelbarer Nähe zu einem Schwellwert gemessen wird. Dementsprechend wird dieser Wert entweder einem zu hohen oder zu niedrigen digitalen Wert zugeordnet. Der Bereich um einen Schwellwert herum, in dem solche kritischen Werte gemessen werden können, wird Unsicherheitsbereich δ genannt. Abbildung 1.3 veranschaulicht diesen Sachverhalt.

Für den tatsächlichen maximalen Fehler[4] F_{max} muss also die Hälfte des Unsicherheitsbereiches zum Quantisierungsfehler addiert werden. Es folgt:

$$F_{max} = F_{Qmax} \pm \frac{1}{2}\delta$$

[4]Prof.Gemmar benennt den maximalen Fehler mit F'_{Qmax}.

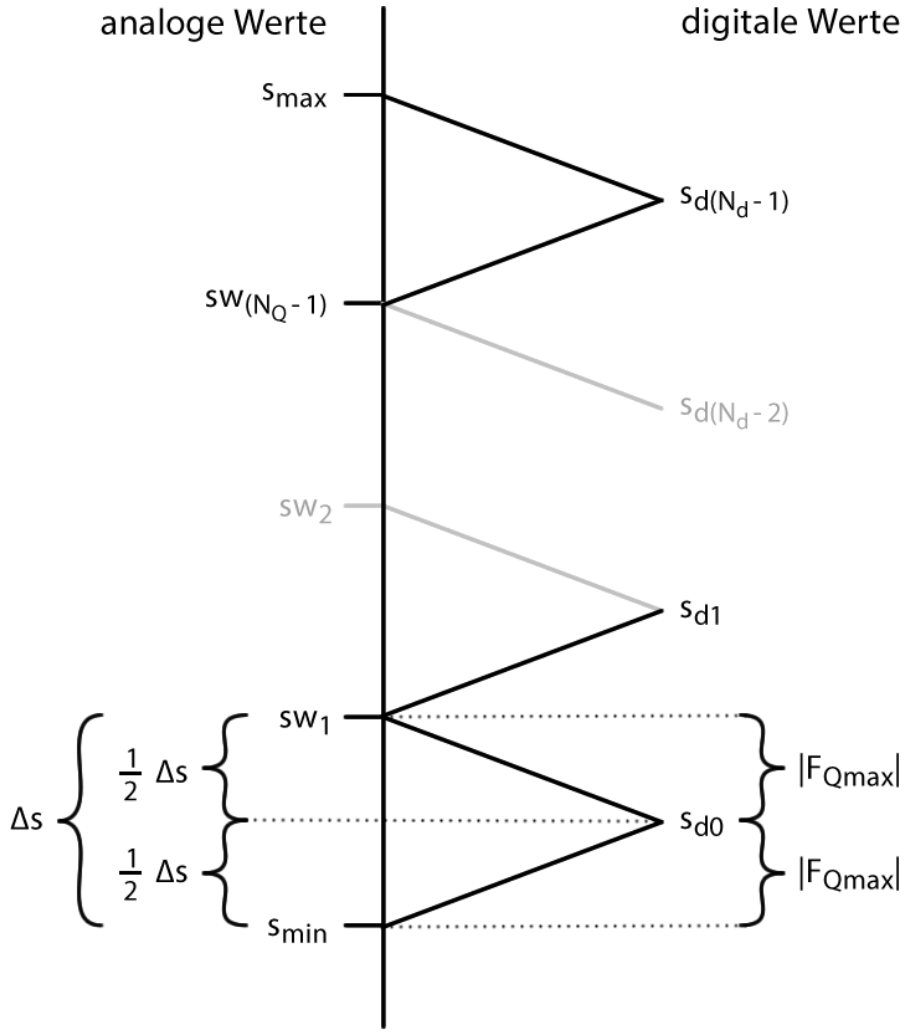


Abbildung 1.2: Analog-Digital-Wandlung mit unvollständigem Signalumfang

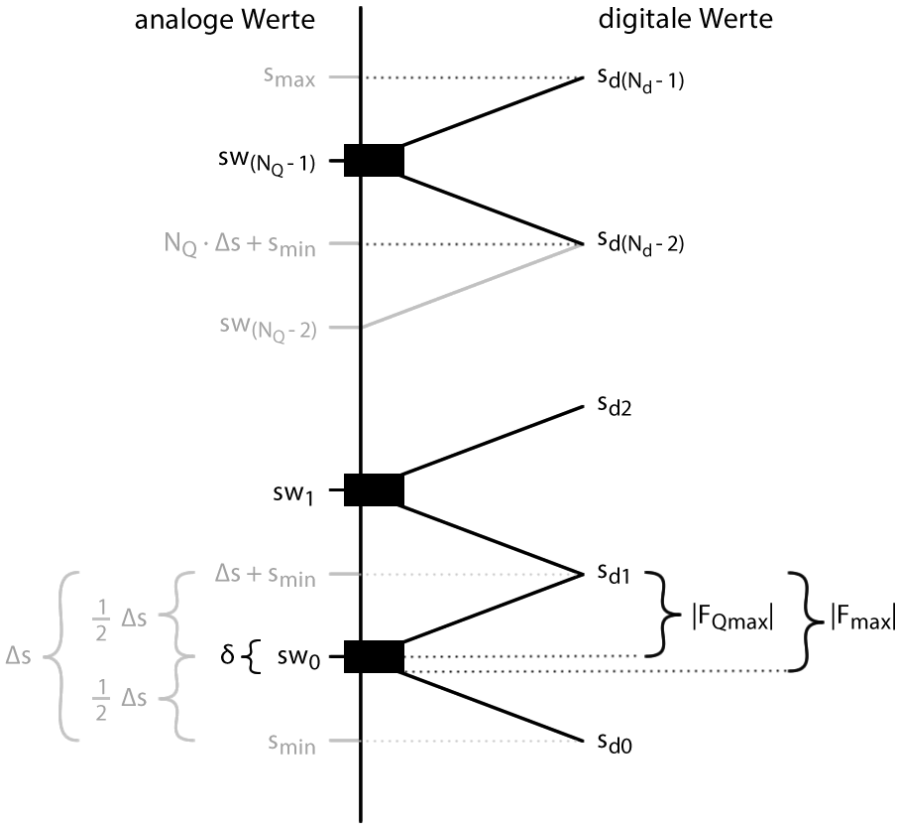


Abbildung 1.3: technischer Unsicherheitsbereich δ um die Schwellwerte

2 Zahlensysteme

Im folgenden werden Polyadische Zahlensysteme besprochen. Solche Systeme haben die Eigenschaft Zahlen anhand einer Basis in Vielfachen von Potenzen dieser Basis aufzuteilen. Dies erleichtert Operationen auf Zahlen und lässt sich mittels der Potenzschreibweise auch leicht und relativ verständlich aufschreiben. Im Alltag benutzen wir das 10ner Polyadische Zahlensystem mit der Stellenschreibweise.

Ein Beispiel: Die Zahl 7645 ist in Stellenschreibweise im 10ner System geschrieben. Die Zahl hat die Wertigkeit 5 Einer, 4 Zehner, 6 Hunderter und 7 Tausender. Das selbe dargestellt als Addition von Vielfachen von Potenzen über die Basis 10:

$$7645 = 7 \cdot 10^3 + 6 \cdot 10^2 + 4 \cdot 10^1 + 5 \cdot 10^0$$

Gebrochene Zahlen können mittels Nachkommastellen dargestellt werden. Jede Nachkommastelle steht dann für die Potenz eines Anteils einer Einheit.

Ein Beispiel: Die Zahl $10,23$ ist in Stellenschreibweise im 10ner System geschrieben. Die Zahl hat die Wertigkeit ein Zehner, null einer, zwei zehntel und drei hundertstel. Das selbe dargestellt als Addition von Vielfachen von Potenzen über die Basis 10:

$$10,23 = 1 \cdot 10^1 + 0 \cdot 10^0 + 2 \cdot 10^{-1} + 3 \cdot 10^{-2} = 1 \cdot 10^1 + 0 \cdot 10^0 + 2 \cdot \frac{1}{10^1} + 3 \cdot \frac{1}{10^2}$$

Eine Ziffer steht also je nach Stelle, an der sie steht, für ein Vielfaches einer Potenz der Basis. Die Potenz wird durch die Stelle bestimmt an der die Ziffer steht. Das Vielfache der Potenz wird mit dem Symbol der Ziffer bestimmt. Da die Wertigkeit einer Potenz grundsätzlich in die jeweils nächste Stelle übertragen wird, gilt für den Wertebereich einer Ziffer z in Basis B:

$$M_z = \{z \in \mathbb{N} \mid 0 \leq z \leq B - 1\}$$

Im Allgemeinen kann die Wertigkeit einer ganzen Zahl Z mit n Vorkomma-, m Nachkommastellen und den Ziffern z_i in der Basis B folgendermaßen bestimmt werden:

$$\begin{aligned} Z_B &= \sum_{i=-m}^{n} B^i \\ &= z_{n-1} \cdot B^{n-1} + z_{n-2} \cdot B^{n-2} + ... + z_1 \cdot B^1 + z_0 \cdot B^0 + z_{-1} \cdot B^{-1} + ... + z_{-m} \cdot B^{-m} \end{aligned}$$

2.1 Zählfunktion

Über die Zählfunktion werden Arithmetische Operationen vereinfacht. Wird der Wertebereich einer Ziffer um eins überschritten wird die Ziffer auf 0 gesetzt und ein Übertrag[1] auf die nächste Stelle übernommen. Bei Unterschreiten des Wertebereichs um eins erfolgt eine Anleihe[2] auf die nächst höhere Stelle. Die Ausführung von arithmetischen Operationen lässt sich auf die Zählfunktion zurück führen. Die Berechnung für jede Stelle ist auf einen simplen Algorithmus zurückführbar und somit gut für eine automatische Ausführung geeignet.

2.2 Stellenverschiebung

Eine weitere Besonderheit ist die Operation der Stellenverschiebung. Eine Stellenverschiebung nach links in einem B-System bedeutet eine Division durch B der Zahl. Dies kann bei ganzen Zahlen entweder durch streichen der niedrigsten Stelle geschehen oder bei gebrochenen Zahlen durch verschieben des Kommas nach links. Eine Stellenverschiebung nach rechts bedeutet eine Multiplikation mit B der Zahl. Eine k-fache Verschiebung bedeutet eine Division bzw. Multiplikation um die $k$te Potenz der Basis. Beispiele:

- $1337_{10} \div 10^1 = 133_{10}$
- $32,1_{10} \cdot 10^2 = 3210_{10}$
- $10110_2 \div 2^1 = 1011_2$
- $111_2 \cdot 2^3 = 111000_2$

2.3 Stellen- und Ziffernaufwand

Der Stellen- und Ziffernaufwand einer Zahl ist von der gewählten Basis und der Größe der Zahl abhängig. Wieviele unterschiedliche Zahlen N bei angegebener Stellenzahl n und Basis B dargestellt werden können lässt sich folgendermaßen berechnen:

$$N = B^n$$

Ist gefragt wieviele Stellen n für N unterschiedliche Zahlen unter Verwendung der Basis B aufzuwenden sind, lässt sich die obige Gleichung folgendermaßen umstellen:

$$n = \lceil log_B(N) \rceil$$

Der Ziffernaufwand gibt an wieviele Zeichen zum Darstellen von einer bestimmten Anzahl unterschiedlicher Zahlen nötig sind. Für jede Stelle sind Anzahl Ziffern in Höhe der

[1] engl. carry
[2] engl. borrow

Basis notwendig. Der Ziffernaufwand zn_B für N unterschiedliche Zahlen in der Basis B wird also bestimmt mit:

$$zn_B = B \cdot n = B \cdot \lceil log_B(N) \rceil$$

2.4 Umwandlung von Zahlensystemen

Mit Hilfe des Hornerschemas lässt sich ein allgemeingültiger Allgorithmus zur Umwandlung eines beliebigen Zahlensystems in ein anderes herleiten. Das Hornerschema beschreibt das sukzessive Ausklammern eines Faktors. Dieses Verfahren lässt sich gut auf die Potenzschreibweise einer Zahl anwenden.

Beispielsweise lässt sich die Zahl 2.938 wie folgt zerlegen:

$$\begin{aligned} 2.938 &= 8 + 10 \cdot 3 + 10^2 \cdot 9 + 10^3 \cdot 2 \\ &= 8 + 10 \cdot (3 + 10 \cdot (9 + 10 \cdot 2)) \end{aligned}$$

Anschließend können die einzelnen Stellen durch Division mit der Basis aquiriert werden indem das jeweilige Ergebnis wiederum mit der Basis dividiert wird.

$$\begin{aligned} 8 + 10 \cdot (3 + 10 \cdot (9 + 10 \cdot 2)) \div 10 &= 3 + 10 \cdot (9 + 10 \cdot 2) \quad \text{Rest } 8 \\ 3 + 10 \cdot (9 + 10 \cdot 2) \div 10 &= 9 + 10 \cdot 2 \quad \text{Rest } 3 \\ 9 + 10 \cdot 2 \div 10 &= 2 \quad \text{Rest } 9 \\ 2 \div 10 &= 0 \quad \text{Rest } 2 \end{aligned}$$

Die Zahl 25_{10} lässt sich in Potenzschreibweise zur Basis 2 folgendermaßen aufschreiben:

$$\begin{aligned} 25_{10} &= 1 + 2 \cdot 0 + 2^2 \cdot 0 + 2^3 \cdot 1 + 2^4 \cdot 1 \\ &= 1 + 2 \cdot (0 + 2 \cdot (0 + 2 \cdot (1 + 2 \cdot 1))) \end{aligned}$$

Anschließend können wiederum die einzelnen Stellen für die Stellenschreibweise zur Basis 2 durch wiederholte Division mit der Basis 2 aquiriert werden.

$$\begin{aligned} 1 + 2 \cdot (0 + 2 \cdot (0 + 2 \cdot (1 + 2 \cdot 1+))) \div 2 &= 0 + 2 \cdot (0 + 2 \cdot (1 + 2 \cdot 1)) \quad \text{Rest } 1 \\ 0 + 2 \cdot (0 + 2 \cdot (1 + 2 \cdot 1)) \div 2 &= 0 + 2 \cdot (1 + 2 \cdot 1) \quad \text{Rest } 0 \\ 0 + 2 \cdot (1 + 2 \cdot 1) \div 2 &= 1 + 2 \cdot 1 \quad \text{Rest } 0 \\ 1 + 2 \cdot 1 \div 2 &= 1 \quad \text{Rest } 1 \\ 1 \div 2 &= 0 \quad \text{Rest } 1 \end{aligned}$$

Da durch das Dividieren mit der Basis immer die niedrigste Stelle des aktuellen Ergebnisses aquiriert wird, ist die letzte Stelle die höchste und muss somit als erstes im Ergebnis notiert werden usw. Das Ergebnis der Umwandlung von 25_{10} in das Dualsystem ist also 11001_2.

Allgemein kann folgender Algorithmus angewandt werden um eine ganze Zahl Z_B zur Basis B in die Basis B^* zu wandeln:

a) Teile Z_B durch B^*. Notiere das Ergebnis Z'_B und den Rest.

b) Wiederhole Schritt a mit Z'_B statt Z_B bis das Ergebnis aus Schritt a 0 ist.

c) Notiere die Zahl im B^*-System. Der Rest der letzten Division ist die höchste Stelle der Zahl im B^*-System. Der Rest der vorletzten Division ist die zweit-höchste Stelle der Zahl im B^*-System usw.

Für die Umwandlung echt-gebrochener Zahlen lässt sich ein ähnliches Verfahren anwenden. Hierbei muss die Zahl wiederholt mit der neuen Basis multipliziert werden. Nach jeder Multiplikation wird der Vorkommaanteil notiert. Mit dem Nachkommaanteil wird die Multiplikation fortgesetzt solange bis der Nachkommaanteil 0 ergibt oder der gleiche Vorkommaanteil, wie zuvor, erreicht wird. Im letzten Fall ergibts sich daraus die Periode. Die Periode lässt sich ab der Stelle mit der Ziffer, die später erneut gefunden wurde, bis zu der Stelle mit der gleichen Ziffer genau feststellen, da sich ab dieser Stelle die vorhergehenden Rechnungen wiederholen würden.

Beispielsweise lässt sich die Zahl $0,375_{10}$ in Potenzschreibweise zur Basis 2 wie folgt zerlegen:

$$\begin{aligned} 0,375_{10} &= 2^{-1} \cdot 0 + 2^{-2} \cdot 1 + 2^{-3} \cdot 1 \\ &= 2^{-1} \cdot (0 + 2^{-1} \cdot (1 + 2^{-1} \cdot 1)) \end{aligned}$$

Die einzelnen Stellen zur Basis 2 erhält man, indem mit 2 multipliziert wird. Die Resultierende Vorkommastelle ist die erste Stelle. Der Nachkommaanteil wird erneut mit 2 multipliziert um die nächste Stelle zu erhalten.

$$\begin{aligned} 2^{-1} \cdot (0 + 2^{-1} \cdot (1 + 2^{-1} \cdot 1)) \cdot 2 &= \underbrace{0}_{Vorkommaanteil} + \underbrace{2^{-1} \cdot (1 + 2^{-1} \cdot 1)}_{Nachkommaanteil} \\ 2^{-1} \cdot (1 + 2^{-1} \cdot 1) \cdot 2 &= \underbrace{1}_{Vorkommaanteil} + \underbrace{2^{-1} \cdot 1}_{Nachkommaanteil} \\ 2^{-1} \cdot 1 \cdot 2 &= \underbrace{1}_{Vorkommaanteil} \end{aligned}$$

Da durch das Multiplizieren mit der Basis immer die höchste Stelle des aktuellen Ergebnisses aquiriert wird, ist die erste Stelle die höchste und muss somit als erstes im Ergebnis notiert werden usw. Das Ergebnis der Umwandlung von $0,375_{10}$ in das Dualsystem ist also $0,011_2$.

Ein Beispiel mit Periode: die Zahl $0,1_{10}$ soll in das Binärsystem gewandelt werden.

$$\left| \begin{array}{rcl} 0,1 \cdot 2 &=& 0,2 \\ 0,2 \cdot 2 &=& 0,4 \\ 0,4 \cdot 2 &=& 0,8 \\ 0,8 \cdot 2 &=& 1,6 \\ 0,6 \cdot 2 &=& 1,2 \end{array} \right|$$

$$0,2 \cdot 2 = 0,4$$

Das Ergebnis ist $0,1_{10} \mapsto 0,\overline{0011}_2$
Ein weiteres Beispiel: $0,2_{10} \mapsto 0,\overline{00111}_2$

Zahlen mit ganzem und gebrochenem Teil werden gewandelt in dem der ganze Teil mit dem Divisionsverfahren und der gebrochene Teil mit dem Multiplikationsverfahren gewandelt wird. Die Ergebnisse der beiden Teilschritte ergeben durch Komma getrennt das Gesamtergebnis.

Ein Beispiel: $19,625_{10}$ soll in das Dualzahlsystem gewandelt werden.

- Wandlung des ganzen Teils:

$$\begin{aligned} 19 \div 2 &= 9 \quad \text{Rest } 1 \\ 9 \div 2 &= 4 \quad \text{Rest } 1 \\ 4 \div 2 &= 2 \quad \text{Rest } 0 \\ 2 \div 2 &= 1 \quad \text{Rest } 0 \\ 1 \div 2 &= 0 \quad \text{Rest } 1 \end{aligned}$$

  also ist der ganze Teil $19_{10} \mapsto 10011_2$

- Wandlung des gebrochenen Teils:

$$\begin{aligned} 0,625 \cdot 2 &= 1,25 \\ 0,25 \cdot 2 &= 0,5 \\ 0,5 \cdot 2 &= 1,0 \end{aligned}$$

  also ist der gebrochene Teil $0,625_{10} \mapsto 0,101_2$

- Das Ergebnis ist demnach $19,625_{10} \mapsto 10011,101_2$

2.4.1 Zahlenwandlungen mit Basen der Form 2^i

Die fortgesetzte Division ist bei der Wandlung vom Dualsystem in ein anderes 2^i-System durch einfache Stellenverschiebung um i möglich. Der Rest sind dabei die Stellen, welche durch die Verschiebung wegfallen.

Beispielsweise lässt sich die Zahl 1010100110_2 durch Tetradenanordnung leicht in das Hexadezimalsystem[3] wandeln:

$$\begin{aligned} 10\,1010\,0110_2 \div 16 &= 10\,1010 \quad \text{Rest } 0110_2 = 6_{16} \\ 10\,1010_2 \div 16 &= 10 \quad \text{Rest } 1010_2 = A_{16} \\ 10_2 \div 16 &= 0 \quad \text{Rest } 10_2 = 2_{16} \end{aligned}$$

$$\underbrace{10}_{2}\,\underbrace{1010}_{A}\,\underbrace{0110}_{6}$$

$$10\,1010\,0110_2 \mapsto 2A6_{16}$$

Mit der Umkehrfunktion lässt sich die Hexadezimaldarstellung einer Zahl auch in die Dualdarstellung wandeln, indem jede Stelle in vier bits codiert wird. Allgemein kann

[3] Hexa = 6; dezi = 10; 16ner System; $16 = 2^4$

von vom 2^i-System in das Binärsystem gewandelt werden, indem jeweils eine Stelle der Zahl im 2^i-System in i Stellen des Binärsystems gewandelt werden. Ein Beispiel:

$$\underbrace{A}_{1010}\underbrace{F}_{1111}\underbrace{F}_{1111}\underbrace{E}_{1110}$$

$$AFFE_{16} \mapsto 1010\ 1111\ 1111\ 1110_2$$

Weitere Beispiele:

$$1\ 00\ 10\ 00_2 \mapsto 1020_4$$
$$231_4 \mapsto 10\ 11\ 01_2$$
$$1\ 001\ 000_2 \mapsto 110_8$$
$$55_8 \mapsto 101\ 101_2$$

2.4.2 Umgang mit Perioden

Je nach Zahlensystem können Zahlen nur mit Periode in Stellenschreibweise geschrieben werden. Die Zahlen können in einen Bruch gewandelt werden. Im Folgenden werden die mathematischen Zusammenhänge erklärt.

Sei p eine periodische Zahl ohne Vorperiode in einem beliebigen Zahlensystem zur Basis B und n sei deren Periodenlänge. Sei o eine ganze Zahl ohne Periode und wie folgt definiert:

$$o = \underbrace{\underbrace{p \cdot B^n}_{\text{Periode als ganzzahliger Anteil}} - p}_{\text{Periode subtrahiert}}$$

o ist also die Darstellung der Periode als Ganzzahl. Durch Umformung gilt:

$$\begin{aligned} o &= p \cdot B^n - p \\ o &= (B^n - 1) \cdot p \\ p &= \frac{o}{B^n - 1} \end{aligned}$$

Beispiel: $0, \overline{1011}_2$ soll in einen Bruch zur Basis 10 gewandelt werden.

$$o = 1011_2; n = 4; B = 2$$

$$\begin{aligned} 0, \overline{1011}_2 &\mapsto \frac{1011_2}{2^4 - 1} \\ 0, \overline{1011}_2 &\mapsto \frac{11_{10}}{15_{10}} \end{aligned}$$

Bei einer Zahl mit Vorperiode der Länge k muss zuerst die Vorperiode durch Multiplizieren mit B^k eliminiert werden. Danach lässt sich der Periodische Anteil berechnen, wie oben angegeben. Das Resultat muss um die Operation auszugleichen mit $\frac{1}{B^k}$ multipliziert und die Vorperiode als Bruch addiert werden. Ein Beispiel: Wandlung der Zahl $0,01\overline{01001011}_2$ in einen Bruch im Dezimalsystem.

1. Eliminiere Vorperiode
$$0,01\overline{01001011}_2 = 0,01 + \frac{1}{2^2} \cdot 0,\overline{01001011}_2$$
2. Wandle periodischen Anteil in Bruch um
$$0,01\overline{01001011}_2 = 0,01_2 + \frac{1}{2^2} \cdot \frac{01001011_2}{2^8 - 1}$$
3. Berechne Ergebnis
$$0,01\overline{01001011}_2 = \frac{1}{4} + \frac{1}{4} \cdot \frac{75}{255} = \frac{11}{34}$$

Mit der Formel $p = \frac{o}{B^n - 1}$ lassen sich auch manche Brüche in Stellenschreibweise umformen. Zum Beispiel kann $\frac{1}{3}$ auch anders geschrieben werden als $\frac{1}{2^2-1}$. Mit Hilfe der Formel kann die Periodenlänge $n = 2$, die Periode als Ganzzahl $o = 1$ und die neue Basis $B = 2$ bestimmt werden. Also ist $\frac{1}{3} \mapsto 0,\overline{01}_2$. Ist eine Zahl mit Periode auf diese Weise bestimmt, lassen sich Vielfache zur Basis B der Zahl ebenfalls bestimmen indem die Periodischen Stellen geschoben werden. Beispielsweise ist $\frac{1}{6} \mapsto 0,0\overline{01}_2$, da $\frac{1}{3} \mapsto 0,\overline{01}$ und $\frac{1}{3} \cdot 2^{-1} = \frac{1}{6}$ ist. Die Multiplikation mit 2^{-1} kann durch eine Stellenverschiebung der Stellen einer Dualzahl um eine Stelle nach rechts vollzogen werden.[4]

2.5 Addition und Subtraktion im Dualsystem

Nach der Zählfunktion[5] gelten folgende Regeln:

$$\begin{aligned}
0+0 &= 0 \\
0+1 &= 1 \\
1+0 &= 1 \\
1+1 &= 0+ \text{ carry in die nächsthöhere Stelle}
\end{aligned}$$

$$\begin{aligned}
0-0 &= 0 \\
0-1 &= 1- \text{ borrow von der nächsthöheren Stelle} \\
1-0 &= 1 \\
1-1 &= 0
\end{aligned}$$

[4] siehe Abschnitt 2.2 zum Thema Stellenverschiebung
[5] Siehe Abschnitt 2.1 unter Zählfunktion

Beim schriftlichen Addieren von Dualzahlen kann folgendes Verfahren angewandt werden:

1. Zähle die 1en in der ersten Spalte.
2. die erste Stelle der Anzahl 1en als Dualzahl wird in dieselbe Spalte geschrieben. Die nachfolgenden Stellen werden in die jeweils nachfolgenden Spalten als carry übernommen.
3. Wiederhole Schritt 1. mit den nachfolgenden Spalten.

Beispiel:

	1	0	0	1	1
+	0	1	0	1	1
				1	
			1		
	1	1	1	1	0

2.5.1 Subtraktion durch Komplementaddition

Das Basis-Komplement $K_B(Z)$ einer Zahl Z mit Anzahl Stellen n ist definiert mit:

$$K_B(Z) = B^n - Z$$

Das Basis-Komplement ist die Umkehrfunktion von sich selbst.

$$\begin{aligned} K_B(K_B(Z)) &= K(B^n - Z) \\ &= B^n - (B^n - Z) \\ K_B(K_B(Z)) &= Z \end{aligned}$$

Das Stellenkomplement[6] $K'_B(Z)$ ist definiert über die Differenz jeder Ziffer an jeder Stelle zur höchstwertigen Ziffer. Es gilt also für jede Ziffer z_i an Stelle i:

$$K'_B(Z) : z_i \mapsto ((B-1) - z_i)$$

Das Basis-Komplement lässt sich leicht berechnen indem das Stellenkomplement um eins inkrementiert wird. Beispiel:

$$K'_{10}(123) = 876$$
$$K_{10}(123) = 1000 - 123 = 876 + 1 = 877$$

Werden negative Zahlen mit dem Basis-Komplement dargestellt, kann die Subtraktion zweier Zahlen über die Addition der beiden Zahlen abgearbeitet werden. Dies gilt weil:

$$a - b = a + K(b) - B^n = a - b + B^n - B^n$$

Das Subtrahieren von B^n kann methodisch je nach Fall behandelt werden. Ist der Minuend größer oder gleich dem Subtrahenden wird die führende eins nach der Berechnung $M \geq S$

[6] Das Stellenkomplement wird auch Einerkomplement genannt.

gestrichen. Wichtig bei dieser Methode ist, dass Minuend und Subtrahend die gleiche Stellenanzahl besitzen. Die höheren Stellen des Subtrahenden müssen gegebenenfalls mit Nullen aufgefüllt werden und über diese Zahl das Komplement gebildet werden. Beispiel:

$$\begin{aligned} 8127_{10} - 342_{10} &= 8127 - 0342 \\ &= 8127 + K_{10}(0342) - 10000 \\ &= 8127 + 10000 - 0342 - 10000 \\ &= 8127 + 9658 - 10000 \end{aligned}$$

Addition von 8127 und 9658. Die Subtraktion von 10000 erfolgt über das Streichen der führenden 10000der-Stelle.

	8	1	2	7
+	9	6	5	8
			1	
1				
	7	7	8	5

Ist der Minuend kleiner als der Subtrahend steht das Ergebnis im negativen Basis-Komplement. Das Ergebnis kann also über das Komplement des Zwischenergebnisses berechnet werden. Beispiel: $M < S$

$$\begin{aligned} 342_{10} - 8127_{10} &= 342 + K_{10}(8127) - 10000 \\ &= 342 + 10000 - 8127 - 10000 \\ &= 342 + 1873 - 10000 \\ &= 2215 - 10000 \\ &= -(10000 - 2215) \\ &= -K_{10}(2215) \\ &= -7785 \end{aligned}$$

Die angegebenen Methoden lassen sich auf jedes Basis-System anwenden.
Beispiel: $65_{10} - 23_{10} = 42_{10} \mapsto 1000001_2 - 10111_2 = 101010_2$

dezimal		dual	
	65		100 0001
$K_{10}(23)$	+77	$K_2(0010111)$	+110 1001
	*1*42		*1*010 1010
	+42		**+10 1010**

Beispiel: $12_{10} - 74_{10} = -62_{10} \mapsto 1100_2 - 1001010_2 = -111110_2$

dezimal		dual	
	12		000 1100
$K_{10}(74)$	$\underline{+26}$	$K_2(1001010)$	$\underline{+011\ 0110}$
	$\underline{0\,38}$		$\underline{0100\ 0010}$
$-K_{10}(38)$	**-62**	$-K_2(1000010)$	**-11 1110**

2.6 Darstellung und Arithmetik rationaler Zahlen

In der Praxis gibt es in digitalen automatischen Rechenanlagen zwei Darstellungsarten für rationale Zahlen: das Festkommaformat und das Fließkommaformat. Den folgenden zwei Abschnitten wird näher auf die einzelnen Formate eingegangen.

2.6.1 Festkommma-Arithmetik

Die Festkommadarstellung wird verwendet um Zahlen Z im Wertebereich $W_F = \{Z \in \mathbb{Q} \mid -1 < Z < 1\}$ abzubilden. Das höchstwertige Bit wird als Vorzeichenbit VZ verwendet und gibt das Vorzeichen der Zahl an. Die übrigen Bits werden als Mantissenbits verwendet. Die Mantisse gibt alle Nachkommastellen an. Die Vorkommastelle ist immer null und muss deshalb nicht codiert werden. Abbildung 2.1 zeigt Schematisch die Festkommadarstellung.

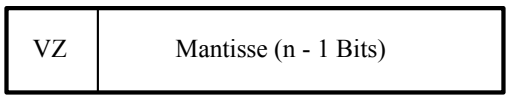


Abbildung 2.1: Festkommadarstellung einer Zahl

Der numerisch kleinste Abstand zwischen zweier Zahlen ist konstant $\Delta Z = 2^{n-1}$. Dies ist ein äquidistantes Zahlenformat.

Der größte Zahlenwert Z_{max} beträgt:

$$\begin{aligned} Z_{max} &= 0,111...1 \\ &= \sum_{i=1}^{n-1} 2^{-i} \\ &= 1 - 2^{-(n-1)} \end{aligned}$$

Der kleinste Zahlenwert Z_{min} beträgt:

$$\begin{aligned} Z_{min} &= -0,111...1 \\ &= -\sum_{i=1}^{n-1} 2^{-i} \\ &= -1 + 2^{-(n-1)} \end{aligned}$$

2.6.2 Fließkomma-Arithmetik

Bei der Fließkommadarstellung wird zusätzlich ein Exponent zur Mantisse gespeichert. Dem Exponenten wird ein konstanter Wert implizit hinzu addiert um auf ein Vorzeichenbit für den Exponenten verzichten zu können. Die so entstandene Zahl wird Charakteristik einer Fließkommazahl genannt. Abbildung 2.2 zeigt schematisch die Fließkommadarstellung.

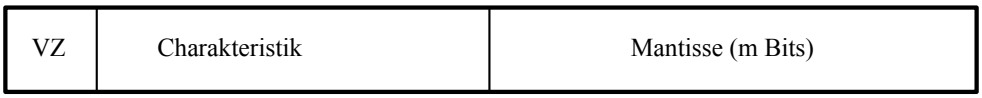


Abbildung 2.2: Fließkommadarstellung einer Zahl

Der Wert einer dualen Fließkommazahl Z mit Mantisse M, Charakteristik C mit konstantem Teil k und Vorzeichen VZ lässt sich also berechnen mit:

$$Z = VZ \cdot M \cdot 2^{C-k}$$

Fließkommazahlen werden normalisiert. Das heißt der Exponent wird so gewählt, dass die Mantisse genau eine Vorkommastelle mit einer Wertigkeit größer null hat. Bei Dualzahlen im Fließkommaformat ist die Vorkommastelle deshalb immer eins und muss nicht gespeichert werden. Eine Ausnahme ist hier die Null. Null hat auch als Vorkommastelle die Ziffer 0 und die Nachkommastellen sind bekanntlicher Weise ebenfalls Null. Deshalb wird die Konvention eingeführt, dass wenn alle Ziffern der dualen Fließkommazahl null sind, der Wert Null gemeint ist.

Beispiel: $+0,010100111_2$ im IEEE 754 32bit float Format:

VZ	Charakteristik (8bit; $k = 2^{8-1} - 1$)	Mantisse (23bit)
+	E + k = -2 + 127 = 125	1,0100 111
0	0111 1101	0100 1110 0000 0000 0000 000

Vor der Addition oder Subtraktion zweier Fließkommazahlen müssen diese erst denormalisiert werden. Der niedrigere Exponent wird immer auf den höherwertigen aufgewertet und die Mantisse um eben so viele Stellen nach rechts verschoben. Durch dieses Verfahren können zwangsläufig Rechenfehler entstehen, da im Computer bei der Denormalisierung bzw. Normalisierung signifikante Stellen der Mantisse verloren gehen können. Beispiele:

$$\begin{aligned} 1,0101E3 + 1,1110111E0 &= 1,0101E3 + 0,0011110111E3 \\ &= (1,0101 + 0,0011110111)E3 \\ &= 1,100011011E3 \end{aligned}$$

$$
\begin{aligned}
1,1101E2 - 1,01E1 &= 1,1101E2 - 0,101E2 \\
&= (1,1101 - 0,101)E2 \\
&= 1,0011E2
\end{aligned}
$$

Nach einer Multiplikation oder einer Division muss das Ergebnis häufig normalisiert werden. Dabei können wiederum zwangsläufig Rechenfehlehler entstehen.

Zum besseren Verständnis empfiehlt es sich mit dieser Standardschreibweise zu experimentieren und auszuprobieren. Dafür eignet sich die Website `http://www.h-schmidt.net/FloatConverter/IEEE754de.html` besonders. Jedes Bit kann per Mausklick oder Tastatur eingabe belegt werden. Es ist auch möglich eine Zahl in Dezimaldarstellung einzugeben und die resultierenden Bits der Float Zahl zu analysieren.

3 Codes

Nachrichten werden in Form von Zeichenketten dargestellt. Zeichenketten wiederum bestehen aus einzelnen Zeichen. Die Zeichen, die für bestimmte Nachrichten verwendet werden, werden im jeweiligen Alphabet[1] festgelegt. Welche Zeichenketten ein Wort einer Nachricht bilden und welche nicht wird in den Produktionsregeln festgelegt.

Ein Wort kann über Konkatenation[2] von Zeichen gebildet werden. Das Symbol der Konkatenation ist „$\circ$“. Wörter einer bestimmten Länge über ein Alphabet werden in Wortmengen zusammen gefasst. Hierzu wird das Alphabetssymbol mit der im Exponenten angegebenen Wortlänge angegeben. Beispiel:
Es sei das Alphabet $A = \{0, 1\}$.
Die Zeichen 0 und 1 sind Elemente von Alphabet A. Kurz: $0, 1 \in A$
Ein Wort über A ist $w = 0 \circ 1 \circ 1 \circ 0 = 0110$
w ist Element von A^4. Kurz: $w \in A^4$

In der Praxis ist das zu verarbeitende Alphabet oft länger als das technisch verfügbare Alphabet. Der Computer kennt auf unterster Ebene nur zwei unterschiedliche Zeichen. Dem Computer steht also ein binäres Alphabet zur Verfügung. Um dennoch umfangreichere Alphabete und Nachrichten, die über dieses gebildet werden, verarbeiten zu können, müssen die Zeichen des Ursprungsalphabets auf Wörter des kleineren Alphabets abgebildet werden. Dieser Umsetzungsprozess wird Codierung genannt. Ein Code c ist also aus dieser Sicht eine eindeutige Zuordnung zwischen Zeichen aus einem Urspungsalphabet U auf Wörter eines anderen Alphabets A.

$$c : U \mapsto A^m$$

$$c : u_i \mapsto a_1 a_2 ... a_m \text{ mit } u_i \in U \wedge a_i \in A$$

Mit einem Alphabet A und m vielen Stellen können maximal $|A|^m$ viele Zeichen codiert werden. Um den minimalen Stellenaufwand S_{min} für n unterschiedliche Zeichen zu bestimmen gilt demnach:

$$S_{min} = \lceil log_{|A|}(n) \rceil$$

Im Folgenden wird beispielhaft berechnet wieviele Bits mindestens nötig sind um die Dezimalziffern $\{0, 1, 2, ..., 9\}$ zu codieren.

$$S_{min} = \lceil ld(10) \rceil = 4$$

[1] Ein Alphabet wird auch Zeichenvorrat genannt.
[2] Konkatenation ist die Aneinanderreihung von Zeichen

Demnach werden mindestens vier bit benötigt um zehn Zeichen über das Binäralphabet abzubilden. Mit vier bit wären allerdings $2^4 = 16$ unterschiedliche Konkatenationen von Zeichen möglich. Um irgendeine Zuordnung von Binärworten auf die zehn Ziffern zu definieren gibt es

$$\frac{16!}{(16-10)!} \approx 2,9 \cdot 10^{10} \text{ Möglichkeiten}$$

Tatsächlich Verwendung finden nur wenige. Welche Vorteile welche Abbildung bringt, wird in den nachfolgenden Abschnitten besprochen.

3.1 Beschreibungsmerkmale für Codes

Grundsätzlich wird zwischen Alphanumerischen Codes und Numerischen Codes unterschieden. Alphanumerische Codes dienen der Darstellung von Ziffern und anderen Zeichen, wie Buchstaben, Trenn und Sonderzeichen und weitere. Numerische Codes dienen der Darstellung von Zahlen. Numerische Codes lassen sich in Wortcodes und Zifferncodes unterteilen. Bei Zifferncodes wird, wie bei Alphanumerischen Codes, jede Ziffer einzeln codiert. In Wortcodes wird dagegen eine Zahl als Ganzes codiert. Ein Beispiel dafür ist der Dualzahlencode. Der Nachteil dabei ist der Kodieraufwand bei der Wandlung bei der Ein-, Ausgabe. Der Vorteil dabei ist, dass arithmetische Operationen oft direkt möglich sind.

Allgemeine Beschreibungsmerkmale für Codes sind:

- Die **Bewertbarkeit** W**:** Für jede Stelle i in einem Codewort wird eine Stellenwertigkeit w_i definiert. Der Dualzahlencode ist zum Beispiel bewertet.
- Das **Gewicht** G ist die Anzahl der mit 1en belegten Stellen in einem Codewort. Bei einem gleichgewichteten Code haben alle Codewörter das gleiche Gewicht.
- Die **Distanz** D zweier Codewörter ist die Anzahl stellen in denen sich die Codewörter unterscheiden.
- Die **Hamming Distanz** HD ist die minimale Stellendistanz zwischen beliebigen Codewörtern eines Codes.

$$HD(C) = min(D(cw_a, cw_b)) \text{ mit } cw_a, cw_b \in C \wedge cw_a \neq cw_b$$

- Die **Stetigkeit**: ein Code ist stetig, wenn die Distanz zwischen benachbarten Codewörtern im gesamten Code gleich ist.

$$C \text{ ist stetig} \Leftrightarrow D(cw_i, cw_{i+1}) = D(cw_j, cw_{j+1})$$

- Die **Redundanz**: ein Code ist redundant, wenn mögliche Kombinationen von Zeichen nicht als Codewörter genutzt werden.
- Die **absolute Redundanz**[3] R_a ist eine Kenngröße eines Codes. Die absolute Redundanz wird über die Anzahl der Kombinationsmöglichkeiten n_k und die Anzahl

[3] Differenz zwischen Anzahl benötigten Bits für alle Kombinationen und Anzahl verwendeten Bits

Codewörter n_{cw} berechnet. Die Anzahl Kombinationsmöglichkeiten hängt von der Wortlänge len_{cw} des Codes ab. Es gilt:

$$n_k = 2^{len_{cw}}$$

$$R_a = ld(n_k) - ld(n_{cw}) \text{ [bit]}$$

Beispiel: Redundanz des BCD-Codes für die Ziffern 0 bis 9

$$n_k = 2^4$$

$$n_{cw} = 10$$

$$R_a = ld(2^4) - ld(10) \approx 4 - 3,32 = 0,68Bit$$

- Die **relative Redundanz**[4] R_r ist eine Kenngröße eines Codes. Die relative Redundanz wird über die Anzahl der Kombinationsmöglichkeiten n_k und die absolute Redundanz berechnet. Es gilt:

$$R_r = \frac{R_a}{ld(n_k)}$$

- Die **Vollständigkeit:** Ein Code wird vollständig genannt wenn die absolute Redundanz null ist. Ist die absolute Redundanz größer null, wird der Code als unvollständig bezeichnet.

$$C \text{ ist vollständig} \Leftrightarrow R_a = 0$$

$$C \text{ ist unvollständig} \Leftrightarrow R_a > 0$$

- **binär-reflektierend:** Ein Code ist binär-reflektierend, wenn die Codewörter aus der Hälfte des Codes gleich der Codewörter der anderen Hälfte des Codes sind mit Ausnahme der vordersten Stelle, die mit jeweils 0 bzw. 1 codiert ist.
- **selbstkomplementierend / selbstinvertierend:** Ein Code ist selbstkomplementierend, wenn das Neuenerkomplement des Dezimalwertes identisch mit dem Stellenkomplement des jeweiligen Codewortes des Dezimalwertes ist.
- **prüfbar bzw. gesichert:** Ein Code ist prüfbar, wenn eine Fehlererkennung möglich ist. Dies ist nur mit redundanten Codes möglich. Ausführlich wird dies im Abschnitt 3.3 besprochen.
- **fehlerkorrigierbar:** Ein Code ist fehlerkorrigierbar, wenn eine Fehlererkennung möglich ist und Fehler korrigiert werden können. Dies ist nur mit redundanten Codes möglich. Ausführlich wird dies im Abschnitt 3.3 besprochen.
- **eineindeutig:** Ein Code ist eineindeutig, wenn für jedes Codewort genau ein Urbild existiert.

[4] Verhältnis zwischen nicht benutzten bits zu insgesamt verwendeten bits

3.2 Beispiele für bekannte Ziffern-Codes

Bewertete Ziffern-Codes

8-4-2-1 Code mit den Eigenschaften:

- stetig / einschrittig
- bewertet 8 - 4 - 2 - 1

Dezimalziffer	8-4-2-1 Code
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Aiken-Code mit den Eigenschaften:

- negationssymmetrisch
- bewertet 2 - 4 - 2 - 1
- selbstkomplementierend

Dezimalziffer	Aiken-Code
0	0000
1	0001
2	0010
3	0011
4	0100
5	1011
6	1100
7	1101
8	1110
9	1111

Excess-3-Code / Stibitz-Code mit den Eigenschaften:

- negationssymmetrisch
- vermeidet 0000 oder 1111
- aus Dualcode mit Addition von 3
- bewertet 8 - (-4) - $\overline{(-2)}$ - $\overline{(-1)}$

Dezimalziffer	Stibitz-Code
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

2 aus 5 mit den Eigenschaften:

- mit gerader Parität
- gleichgewichtig mit $G = 2$
- bewertet 7 - 4 - 2 - 1 - 0

Dezimalziffer	2 aus 5
0	11000 $(= 11_{10})$
1	00011
2	00101
3	00110
4	01001
5	01010
6	01100
7	10001
8	10010
9	10100

Nicht bewertbare Zifferncodes

Gray-Code mit den Eigenschaften:

- binär-reflektierend[a]
- einschrittig
- vollständig

Dezimalziffer	Gray-Code
0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101
10	1111
11	1110
12	1010
13	1011
14	1001
15	1000

[a] Achsen-Spiegelung der Codewörter

O'Brien mit den Eigenschaften:

- binär-reflektierend[a]
- einschrittig

Dezimalziffer	Gray-Code
0	0000
1	0001
2	0011
3	0010
4	0110
5	1110
6	1010
7	1011
8	1001
9	1000

[a] Achsen-Spiegelung der Codewörter

3.3 Gesicherte Codes

Gesicherte Codes haben die Eigenschaft Fehler bei der Übertragung von Codewörtern zu erkennen. Der Empfänger kann in bestimmten Fällen den Fehler erkennen indem eine Kombination aus $\{0,1\}^n$ empfangen wird, welche kein gültiges Codewort darstellt. Mit bestimmten Verfahren können Fehler teilweise korrigiert werden. Codes können somit in die Klassen nicht fehlererkennbarer, fehlererkennbarer oder fehlerkorrigierbarer Codes eingeordnet werden.

Damit ein Code fehlererkennbar sein kann, muss mindestens ein Bit mehr verwendet werden als nötig. Also ist die notwendige Bedingung für fehlererkennbare Codes, dass die absolute Redundanz größer als eins ist.

$$\text{Code ist fehlererkennbar } \Rightarrow R_a \geq 1 \text{ Bit}$$

Damit ein Code tatsächlich fehlererkennbar ist, muss die Hamming-Distanz größer als

eins sein. Die Hinreichende Bedingung ist also $HD(C) > 1$. Es können $HD(C) - 1$ viele Fehler erkannt werden. Um n_F viele Fehler zu erkennen muss der Code also folgende Hamming-Distanz aufweisen:

$$HD_{Fe}(C) = n_F + 1$$

Für die Fehlerkennung haben sich die Quersummenprüfung und gleichgewichtige Codes etabliert. Bei der Quersummenprüfung wird jedes Codewort um ein Parity Bit ergänzt. Das Parity Bit sorgt bei gerader Parität dafür, dass eine gerade Anzahl 1en im jeweiligen Codewort vorkommt. Bei ungerader Parität sorgt das Parity Bit dafür, dass eine ungerade Anzahl 1en im jeweiligen Codewort vorkommt. Damit sind Einzelfehler erkennbar. Doppelfehler werden nicht erkannt, da sie sich gegenseitig aufheben.

Um n_F viele Fehler korrigieren zu können wird mindestens folgende Hamming-Distanz benötigt:

$$HD_{Fk}(C) = 2 \cdot n_F + 1$$

3.3.1 Blocksicherungsverfahren

Mit Blocksicherungsverfahren ist die blockweise Korrektur von Einfachfehlern pro Block möglich. Doppelfehler werden erkannt, können aber nicht korrigiert werden. Ein Block besteht bei Codewortlänge n_{cw} aus $n_{cw} + 1$ zusätzlichen Parity Bits und einem zusätzlichen Prüfwort der Länge $n_{cw} + 1$. Es kann auf gerade oder ungerade Parität geprüft werden. Bei einem Einfachfehler im Block kann über die fehlerhafte Zeile und fehlerhafte Spalte der Fehler genau lokalisiert und damit korrigiert werden. Beispiele mit Codewörtern der Länge drei und Prüfung auf gerader Parität:

	P	$\mathbf{c_2}$	$\mathbf{c_1}$	$\mathbf{c_0}$	
cw_0	0	1	0	1	✓
cw_1	1	0	1	0	✓
cw_2	1	1	0	0	✓
pw	0	0	1	1	✓
	✓	✓	✓	✓	

Prüfung erfolgreich abgeschlossen; kein Fehler

	P	$\mathbf{c_2}$	$\mathbf{c_1}$	$\mathbf{c_0}$	
cw_0	0	1	0	1	✓
cw_1	1	0	1	0	✓
cw_2	1	1	**1**	0	f
pw	0	0	1	1	✓
	✓	✓	f	✓	

Prüfung erkennt einen Fehler; Fehler ist korrigierbar

	P	$\mathbf{c_2}$	$\mathbf{c_1}$	$\mathbf{c_0}$	
cw_0	0	1	0	1	✓
cw_1	1	0	1	0	✓
cw_2	1	1	1	1	✓
pw	0	0	1	1	✓
	✓	✓	f	f	

Prüfung erkennt zwei Fehler; Fehler sind nicht korrigierbar

	P	$\mathbf{c_2}$	$\mathbf{c_1}$	$\mathbf{c_0}$	
cw_0	0	1	0	1	✓
cw_1	1	0	0	0	f
cw_2	1	1	1	0	f
pw	0	0	1	1	✓
	✓	✓	✓	✓	

Prüfung erkennt zwei Fehler; Fehler sind nicht korrigierbar

	P	$\mathbf{c_2}$	$\mathbf{c_1}$	$\mathbf{c_0}$	
cw_0	0	1	0	1	✓
cw_1	1	0	1	1	f
cw_2	1	1	1	0	f
pw	0	0	1	1	✓
	✓	✓	f	f	

Prüfung erkennt zwei Fehler; Fehler sind nicht korrigierbar

	P	$\mathbf{c_2}$	$\mathbf{c_1}$	$\mathbf{c_0}$	
cw_0	0	0	0	1	f
cw_1	1	0	1	0	✓
cw_2	1	1	0	1	f
pw	0	0	1	1	✓
	✓	f	✓	f	

Prüfung erkennt zwei Fehler; Fehler sind nicht korrigierbar

3.3.2 Systematische Hamming-Codes

Das Verfahren um Hamming-Codes zu erstellen ist vom Blocksicherungsverfahren abgeleitet. Ein einzelnes Codewort ist dabei als Block zu betrachten und zu prüfen. Ein Codewort mit n Stellen setzt sich aus n_p vielen Paritätsbits und n_i vielen Informationsbits zusammen (vgl. Abbildung 3.1).

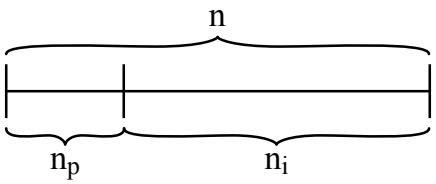


Abbildung 3.1: Schematische Darstellung eines Hamming-Codeworts

Um einen Fehler korrigieren zu können, muss jede Stelle auf einen Fehler überprüft werden und es muss ein gültiges Codewort existieren. Da jedes Bit, auch Paritätsbits, fehlerhaft sein können, müssen n Fehlerfälle abgedeckt werden. Zusätzlich muss der Fall erfasst werden, dass das Codewort korrekt übertragen wurde. Somit müssen $n+1$ Fälle abgedeckt werden. Mit n_p vielen Paritätsbits können 2^{n_p} viele Fälle codiert werden. Damit alle Fälle abgedeckt werden können muss die folgende Ungleichung erfüllt sein:

$$\begin{aligned} 2^{n_p} &\geq n+1 \\ 2^{n_p} &\geq n_p + n_i + 1 \end{aligned}$$

Beispiel: um vier Informationsbits zu sichern werden mindestens drei Paritätsbits benötigt.

$$\begin{aligned} 2^3 &\geq 3+4+1 \\ 8 &\geq 8 \end{aligned}$$

Für die Einfehlerkorrektur bei Blockcodes muss jedes Informationsbit durch mindestens zwei Paritätsbits gesichert werden. Für ein Informationsbit i, dessen zwei Paritätsbits p_a und p_b und deren Paritäten $P(i,p_a)$ und $P(i,p_b)$ kann dann folgende Fallunterscheidung vorgenommen werden:

- $P(i, p_a)$ und $P(i, p_b)$ sind korrekt. Demnach sind p_a, p_b und i korrekt.
- $P(i, p_a)$ und $P(i, p_b)$ sind falsch. Demnach ist i falsch.
- $P(i, p_a)$ ist falsch. $P(i, p_b)$ ist korrekt. Demnach ist p_a falsch.
- $P(i, p_a)$ ist korrekt. $P(i, p_b)$ ist falsch. Demnach ist p_b falsch.

Diese Regel schließt nicht aus, dass ein Paritätsbit für mehrere Informationsbits zur Prüfung der Parität verwendet werden kann. Die Parität wird in diesem Fall nicht über zwei Bits gebildet, sondern über das jeweilige Paritätsbit inklusive der ihm zugeordneten Informationsbits. Damit eine eindeutige Fehlerlokalisierung möglich ist muss auf eine entsprechend günstige Verteilung der Informationsbits auf die Paritätsbits geachtet werden. Wenn über die Fallunterscheidung der Paritäten einem Fall genau ein Fehler zugeordnet werden kann, kann dieser Fehler entsprechend korrigiert werden. Ein Praktisches Schema zum Finden einer optimalen Verteilung wird im folgenden Abschnitt mit Hilfe eines Beispiels erläutert.

Ein Code mit vier Informationsbits soll gesichert werden. Dazu sind drei Paritätsbits notwendig[5]. Jedes Codewort umfasst also sieben Stellen.

1.	die Stellen werden von 1 bis n durchnummeriert.	1	2	3	4	5	6	7
2.	die Nummerierung wird dual codiert.	001	010	011	100	101	110	111
3.	das j-te Paritätsbit wird an der Stelle eingetragen, die im Dualcode an der j-ten Stelle eine 1 trägt. Die Informationsbits füllen der Reihe nach die übrigen Stellen.	p_1	p_2	i_1	p_3	i_2	i_3	i_4
4.	Dem j-ten Paritätsbit werden diejenigen Informationsbits zugeordnet deren Stelle im Dualcode an der j-ten Stelle eine 1 trägt.	p_1 prüft $\{i_1, i_2, i_4\}$	p_2 prüft $\{i_1, i_3, i_4\}$		p_3 prüft $\{i_2, i_3, i_4\}$			
5.	Die Codewörter werden erstellt. Im Beispiel wird das Codewort für 1010 angegeben. Es wird auf gerade Parität geprüft.	1	0	1	1	0	1	0

[5] siehe oben

Dieses Schema bietet mehrere Vorteile:

- Jedes Informationsbit wird durch mindestens zwei Paritäten geprüft.
- Jedes Paritätsbit wird durch eine Parität geprüft.
- Es entsteht eine optimale Zuordnung der Paritätsbits auf die Informationsbits.
- Fehlercode: Wegen der Anordnung der Paritätsbits wird beim Prüfen der Paritäten die Fehlerstelle direkt über die fehlerhaften Paritäten dual codiert angezeigt. Wenn zum Beispiel die Parität über dem Paritätsbit p_2 fehlerhaft ist, alle anderen jedoch richtig, dann ist das fehlerhafte Bit das zweite. Wenn die Paritäten über p_1 und p_2 fehlerhaft ist, dann ist das dritte Bit fehlerhaft usw.

4 Boolesche Algebra

In diesem Artikel wird die Boolesche Algebra stark in Verbindung mit der Schaltalgebra behandelt. Deshalb werden die Zeichen 0 und 1 für die binäre Grundmenge $B = \{0, 1\}$ verwendet. Weitere Unterschiede in der Notation ergeben sich bei der Besprechung der Operatoren.

4.1 Operatoren

In den folgenden Abschnitten werden die in diesem Artikel verwendeten Operatoren besprochen. Bei booleschen Operatoren kann eine endliche Wertetabelle angegeben werden, da die Grundmenge B endlich ist und die Operatoren endlich viele Elemente behandeln.

4.1.1 Kartesisches Produkt „×“

Das Kartesische Produkt wird über Mengen gebildet. Dabei werden alle Elemente der ersten Menge mit allen Elementen der zweiten Menge in einzelnen Tupeln zusammengefasst.

$$B \times B = (0,0); (0,1); (1,0); (1,1)$$

Das n-fache Produkt einer Menge bezeichnet alle n-fachen Kombinationen aus allen Elementen der Menge.

$$B \times B \times B ... \times B = B^n$$

4.1.2 Negation „NICHT“ „NOT“ „ $\overline{a}$ “

Die Negation ist eine einstellige Operation, welche den Wert invertiert.

$$f_{NOT} : B \mapsto B$$

$$f_{NOT}(a) = \overline{a} \text{ mit } a \in B$$

a	$\overline{a}$
0	1
1	0

4.1.3 Konjunktion „UND“ „AND“ „ · “

Die Konjunktion ist 1, wenn beide behandelten Werte 1 sind, ansonsten 0.

$$f_{OR} : B \times B \mapsto B$$

$$f_{OR}(a, b) = a \cdot b \text{ mit } a, b \in B$$

a	b	$a \cdot b$
0	0	0
0	1	0
1	0	0
1	1	1

4.1.4 Disjunktion „ODER“ „OR“ „ + “

Die Disjunktion ist 1, wenn einer der behandelten Werte 1 ist, ansonsten 0.

$$f_{OR} : B \times B \mapsto B$$

$$f_{OR}(a, b) = a + b \text{ mit } a, b \in B$$

a	b	$a + b$
0	0	0
0	1	1
1	0	1
1	1	1

4.1.5 Äquivalenz „ ↔ “

Die Äquivalenz ist 1, wenn beide behandelten Werte gleich sind, ansonsten 0.

$$f_{eq} : B \times B \mapsto B$$

$$f_{eq}(a, b) = a \leftrightarrow b \text{ mit } a, b \in B$$

a	b	$a \leftrightarrow b$
0	0	1
0	1	0
1	0	0
1	1	1

4.1.6 Antivalenz „XOR“ „ $\nleftrightarrow$ “ „ $\oplus$ “

Die Antivalenz ist 1, wenn beide behandelten Werte ungleich sind, ansonsten 0.

$$f_{an} : B \times B \mapsto B$$

$$f_{an}(a, b) = a \nleftrightarrow b \text{ mit } a, b \in B$$

a	b	$a \nleftrightarrow b$
0	0	0
0	1	1
1	0	1
1	1	0

4.1.7 „*NAND*“

NAND ist die Negation der Disjunktion. Mit *NAND* lassen sich alle logischen Funktionen abbilden. Dies ist bei der Produktion von Hardware von Vorteil.

$$f_{NOR} : B \times B \mapsto B$$

$$f_{NOR}(a, b) = a \ NAND \ b \text{ mit } a, b \in B$$

a	b	$a \ NAND \ b$
0	0	1
0	1	1
1	0	1
1	1	0

4.1.8 „*NOR*“

NOR ist die Negation der Disjunktion. Mit *NOR* lassen sich alle logischen Funktionen ebenfalls abbilden. Dies ist bei der Produktion von Hardware von Vorteil.

$$f_{NOR} : B \times B \mapsto B$$

$$f_{NOR}(a, b) = a \ NOR \ b \text{ mit } a, b \in B$$

a	b	$a \ NOR \ b$
0	0	1
0	1	0
1	0	0
1	1	0

4.1.9 Implikation „→“

Die Implikation von a nach b ist 0, wenn $a = 0$ ist und $b = 1$, ansonsten 1. Die Implikation kann auch geschrieben werden als $(\overline{a} + b)$.

$$f_{im} : B \times B \mapsto B$$

$$f_{im}(a, b) = a \to b = \overline{a} + b \text{ mit } a, b \in B$$

a	b	$a \to b$
0	0	1
0	1	1
1	0	0
1	1	1

4.2 Grundlegende Gesetze

Es gelten folgende grundlegende Gesetze in der booleschen Algebra:

Kommutativ-Gesetze

$$a \cdot b = b \cdot a$$
$$a + b = b + a$$

Assoziativ-Gesetze

$$(a \cdot b) \cdot c = a \cdot (b \cdot c)$$
$$(a + b) + c = a + (b + c)$$

Distributiv-Gesetze

$$a \cdot (b + c) = (a \cdot b) + (a \cdot c)$$
$$a + (b \cdot c) = (a + b) \cdot (a + c)$$

Neutrale Elemente

$$a \cdot 1 = a$$
$$a + 0 = a$$
$$a \cdot a = a$$
$$a + a = a$$

Komplemente

$$a \cdot \overline{a} = 0$$
$$a + \overline{a} = 1$$

Negation der Negation

$$\overline{\overline{a}} = a$$

Gesetze nach De Morgan und Shannon

$$\overline{a \cdot b} = \overline{a} + \overline{b}$$
$$\overline{a + b} = \overline{a} \cdot \overline{b}$$
$$\overline{a \leftrightarrow b} = \overline{a} \nleftrightarrow \overline{b}$$
$$\overline{a \nleftrightarrow b} = \overline{a} \leftrightarrow \overline{b}$$

Absorptionsgesetze

$$a + (a \cdot b) = a$$
$$a \cdot (a + b) = a$$

In diesem Artikel werden folgende Bindungsstärken festgelegt:

1. Negation bindet stärker als Konjunktion
2. Konjunktion bindet stärker als Disjunktion

Das Konjunktionsszeichen „ · “ zwischen Operanden wird zur Vereinfachung der Schreibweise weggelassen.

4.3 Entwurf von Schaltnetzen

Logische Funktionenen werden hardware-technisch in Logikgattern produziert. Wegen der hohen Flexibilität der Booleschen Algebra können verschiedene Gatteranordnungen gleiche Funktionen repräsentieren. Also ist Ziel des Entwurfs eine aufwandsgünstigste Lösung zu finden. Mit der heutigen Halbleitertechnologie werden hochintegrierte Schaltkreise gefertigt. Um den Testaufwand zu vereinfachen werden möglichst regelmäßige Strukturen bevorzugt.

Jede logische Funktionsvorschrift lässt sich als Funktionstabelle[1] oder Term[2] darstellen. Die Funktionsterme dienen als Vorlage für den Gatterbau. Nach den Regeln der Boolschen Algebra können Funktionsterme umgeformt und so in eine optimale Form gebracht werden. Zur Entwicklung des Gatterentwurfs werden in der Praxis SILICON-Compiler Programme eingesetzt.

4.3.1 Definition der Booleschen Normalformen

Ein Boolescher Funktionsterm kann aus mehreren Eingabe-Variablen oder deren Negation und deren logischen Verknüpfungen bestehen.

Die Konjunktive Form (KF) einer logischen Funktion ist die Konjunktion von Disjunktionen. Beispiel:

$$f_{KF}(a,b,c,d) = (a+c)\cdot(\overline{c}+d)\cdot(\overline{a}+c+d)$$

Die Disjunktive Form (DF) einer logischen Funktion ist die Disjunktion von Konjunktionen. Beispiel:

$$f_{DF}(a,b,c,d) = (a\cdot c)+(\overline{a}\cdot d)+(\overline{a}\cdot b\cdot d)$$

Die kanonische Normalform eines Funktionsterms ist die Form in der alle Variablen der Funktion, entweder als solche oder negiert, in einer Disjunktiven Form oder Konjunktiven Form verknüpft sind. Die jeweilige Form wird *Disjunktive kanonische Normalform (DNF)* bzw. *Konjunktive kanonische Normalform (KNF)* genannt. Beispiele:

$$f_{KNF}(a,b,c,d) = \underbrace{(a+\overline{b}+c+d)}_{Maxterm/Faktor}\cdot(a+b+\overline{c}+d)\cdot(\overline{a}+\overline{b}+c+d)$$

[1] Funktionstabelle: auch genannt Schaltbelegungstabelle; Krzl. SBT
[2] Term wird hier synonym für mathematischer Ausdruck verwendet

$$f_{DNF}(a,b,c,d) = \underbrace{(a \cdot b \cdot c \cdot d)}_{Minterm/Summand} + (\overline{a} \cdot \overline{b} \cdot \overline{c} \cdot d) + (\overline{a} \cdot b \cdot c \cdot d)$$

Die Faktoren der Konjunktiven kanonischen Normalform werden Maxterme genannt. Die Summanden der Disjunktiven kanonischen Normalform heißen Minterme.

Jeder logische Term lässt sich in der Konjunktiven Form oder der Disjunktiven Form darstellen. Durch Erweiterung kann jeder Term außerdem in die Disjunktive oder Konfjunktive Normalform gebracht werden. Die Disjunktive Form wird durch „Ausmultiplizieren“ und die Konjunktive Form durch „Ausaddieren“ erstellt.

Eine Disjunktive Formen kann in die Disjunktive Normalform umgeformt werden indem den Summanden das für Konjunktionen neutrale Element $1 = (x + \overline{x})$ je nach fehlender Variable hinzugefügt wird. Anschließend muss wiederum ausmultipliziert werden. Entsprechend kann die Konjunktive Normalform erschlossen werden indem den Faktoren das für Disjunktionen neutrale Element $0 = (x \cdot \overline{x})$ je nach fehlender Variable hinzugefügt wird. Danach wird wieder ausaddiert.

Beispiel: Der Funktionsterm $f = a(b+c)+c(a+\overline{b}c)$ soll in Konjunktive bzw. Disjunktive Form umgeformt werden. Anschließend werden die erhaltenen Formen in die Normalform überführt.

Umformung in Disjunktive Form durch „Ausmultiplizieren“ (siehe Abbildung 4.1):

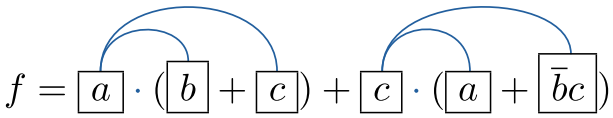


Abbildung 4.1: Ausmultiplizieren

$$\begin{aligned} f &= a(b+c) + c(a+\overline{b}c) \\ &= ab + ac + ca + c\overline{b}c \\ f_{DF} &= ab + ac + \overline{b}c \end{aligned}$$

Umformung in Konjunktive Form durch „Ausaddieren“ (siehe Abbildung 4.2):

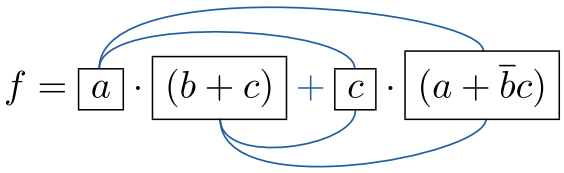


Abbildung 4.2: Ausaddieren

$$
\begin{aligned}
f &= a(b+c) + c(a+\overline{b}c) \\
&= (a+c)(a+(a+\overline{b}c))((b+c)+c)((b+c)+(a+\overline{b}c)) \\
&= (a+c)(a+\overline{b}c)(b+c)(b+a+\underbrace{c+\overline{b}c}_{Absorption}) \\
&= (a+c)(a+\overline{b}c)(b+c)(a+b+c) \\
&= (a+c)(a+\overline{b})(a+c)(b+c)(a+b+c) \\
f_{KF} &= (a+c)(a+\overline{b})(b+c)
\end{aligned}
$$

Herleitung der Disjunktiven Normalform durch Hinzufügen neutraler Elemente der Form $(x+\overline{x})$:

$$
\begin{aligned}
f_{DF} &= ab + ac + \overline{b}c \\
&= ab(c+\overline{c}) + ac(b+\overline{b}) + \overline{b}c(a+\overline{a}) \\
&= abc + ab\overline{c} + acb + ac\overline{b} + \overline{b}ca + \overline{b}c\overline{a} \\
f_{DNF} &= abc + ab\overline{c} + a\overline{b}c + \overline{a}\overline{b}c
\end{aligned}
$$

Herleitung der Konjunktiven Normalform durch Hinzufügen neutraler Elemente der Form $(x \cdot \overline{x})$:

$$
\begin{aligned}
f_{DF} &= (a+c)(a+\overline{b})(b+c) \\
&= (a+c+(b\cdot\overline{b}))(a+\overline{b}+(c\cdot\overline{c}))(b+c+(a\cdot\overline{a})) \\
&= ((a+c+b)\cdot(a+c+\overline{b}))((a+\overline{b}+c)\cdot(a+\overline{b}+\overline{c}))((b+c+a)\cdot(b+c+\overline{a})) \\
&= (a+b+c)(a+\overline{b}+c)(a+\overline{b}+\overline{c})(\overline{a}+b+c)
\end{aligned}
$$

Die Minterme beziehungsweise Maxterme einer logischen Funktion sind eindeutig bestimmt. Demnach besitzt jede logische Funktion genau eine Konjunktive und eine Disjunktive kanonische Normalform. Sie bilden den Ausgangspunkt allgemeingültiger Reduktionsmethoden.

4.3.2 Schaltbelegungstabelle

Logische Funktionen können auch mittels einer Schaltbelegungstabelle spezifiziert werden. Die Min- und Maxterme können aus dieser Tabelle direkt ausgelesen und damit die kanonischen Normalformen erstellt werden. Zur Erstellung der Schaltbelegungstabelle

Tabelle 4.1: Schaltbelegungstabelle für $f(a, b, c)$

i	0	1	2	3	4	5	6	7
a	0	0	0	0	1	1	1	1
b	0	0	1	1	0	0	1	1
c	0	1	0	1	0	1	0	1
y	0	0	1	0	1	1	0	0
m_i			$\overline{a}b\overline{c}$		$a\overline{b}\overline{c}$	$a\overline{b}c$		
M_i	$(a+b+c)$	$(a+b+\overline{c})$		$(a+\overline{b}+\overline{c})$			$(\overline{a}+\overline{b}+c)$	$(\overline{a}+\overline{b}+\overline{c})$

werden für eine n-stellige Funktion alle 2^n Kombinationen der Eingangsvariablen aufgeschrieben und das jeweilige Ergebnis dazu notiert. Damit ist für alle Eingangswerte der Funktion das Ergebnis der Funktion definiert. Beispiel:

Die Schaltbelegungstabelle gibt die Belegungen der Variablen an, bei denen der Funktionswert 1 ist. Da eine Disjunktion genau dann 1 ist, wenn eines ihrer Summanden 1 ist, lassen sich die Minterme aus der Schaltbelegungstabelle ablesen. Ein Minterm ergibt sich aus jeder Spalte, in der der Funktionswert 1 ist. Ist die Belegung der Variable in dieser Spalte 1, dann muss im Minterm die Variable als solche angegeben werden. Ist die Belegung der Variable 0, dann muss im Minterm die negierte Variable angegeben werden (vgl. Tabelle 4.2 Spalten $i \in \{2, 4, 5\}$). Im angegebenen Beispiel ist die Disjunktive kanonische Normalform der Funktion $f(a, b, c)$ also:

$$\sum_{i \in \{2,4,5\}} m_i = \sum m(2, 4, 5)$$

$$m_2 + m_4 + m_5 = \overline{a}b\overline{c} + a\overline{b}\overline{c} + a\overline{b}c$$

Für die Herleitung der Maxterme betrachten wir zuerst die negierte Funktion. Die Minterme der negierten Funktion befinden sich an genau den Stellen, an der die Funktion ihre Maxterme hat. Über Negation der negierten Funktion wird wieder die ursprüngliche Funktion erzeugt. Die negierten Minterme der negierten Funktion sind nach De Morgan also die Maxterme der ursprünglichen Funktion.

Mit den Maxtermen kann analog folgendermaßen verfahren werden: Die Schaltbelegungstabelle gibt die Belegungen der Variablen an, bei denen der Funktionswert 0 ist. Ein Maxterm ergibt sich aus jeder Spalte, in der der Funktionswert 0 ist. Ist die Belegung der Variable in dieser Spalte 1, dann muss im Maxterm die negierte Variable angegeben werden. Ist die Belegung der Variable 0, dann muss im Maxterm die Variable als solche angegeben werden (vgl. Tabelle 4.2 Spalten $i \in \{0, 1, 3, 6, 7\}$)[3]. Im angegebenen Beispiel ist die Konjunktive kanonische Normalform der Funktion $f(a, b, c)$ also:

$$\prod_{i \in \{0,1,3,6,7\}} M_i = \prod M(0, 1, 3, 6, 7)$$

$$M_0 \cdot M_1 \cdot M_3 \cdot M_6 \cdot M_7 = (a+b+c)(a+b+\overline{c})(a+\overline{b}+\overline{c})(\overline{a}+\overline{b}+c)(\overline{a}+\overline{b}+\overline{c})$$

[3] also salopp genau anders herum als vorher

4.3.3 Redundante Funktionen

In manchen Fällen kommen bestimmte Kombinationen der Belegung der Eingabevariablen nie vor. Folglich müssen Funktionswerte für diese Kombinationen nicht zwingend definiert werden. Der Funktionswert in solchen Fällen ist also beliebig. In der Schalttabelle werden nicht definierte Funktionswerte mit „ * “ gekennzeichnet. Bei der Optimierung können nicht zuerst nicht definierte Funktionswerte frei gewählt werden und können gegebenenfalls für so für eine Optimierung sorgen, die anders nicht möglich gewesen wäre.

4.3.4 Reduktion von Schaltfunktionenen (Minimierung)

Die Komplexität von Schaltfunktionen wird reduziert um folgende Vorteile zu nutzen:

- Vereinfachung der logischen Funktionsterme
- Verringerung der Schaltnetzkosten (Gatteranzahl, chip-/board-Fläche, Energie usw.)
- Zuverlässigkeit

„Regelmäßige“ Schaltfunktionen haben demgegenüber folgende Vorteile:

- regelmäßige Lösungsstrukturen
- Geringerer Wartungs-, Pflege und Entwurfsaufwand
- Es können VLS(I)-Schaltkeise (PROM, PAL, GAL, FPGA usw.)

Zur Reduktion können unterschiedliche Verfahren verwendet werden:

- Algebraische Reduktion
- Grafische Reduktion
- Algorithmische Reduktion

Logische Terme können durch Hinzufügen neutraler Elemente erweitert werden. 0 ist das neutrale Element der Disjunktion. 0 kann auch anders geschrieben werden als:

$$0 = x \cdot \overline{x} \quad \text{mit } x \text{ als logischer Term}$$

Also kann jeder logischer Term disjunktiv mit $x \cdot \overline{x}$ verknüpft werden ohne, dass sich der Wert des Terms ändert.

$$a = a + (x \cdot \overline{x}) \quad \text{mit } x, a \text{ als logische Terme}$$

Analog gilt für das neutralen Element 1 der Konjunktion:

$$1 = x + \overline{x} \quad \text{mit } x \text{ als logischer Term}$$

und jeder Term kann konjunktiv mit $x \cdot \overline{x}$ verknüpft werden ohne, dass sich der Wert des Terms ändert.

$$a = a \cdot (x + \overline{x}) \quad \text{mit } x, a \text{ als logische Terme}$$

Um einen Funktioinsterm zu minimieren müssen diese neutralen Elemente aus dem Funktionsterm eliminiert werden. Bei der Disjunktiven Kanonischen Normalform lässt sich die Form ausnutzen indem Summanden gesucht werden, deren Faktoren ähnlich sind. Ähnlich in diesem Fall bedeutet, dass die Faktoren der beiden zu betrachtenden Summanden gleich sind mit Ausnahme eines Faktors, welcher im einen Summand als solcher vorkommt und im anderen negiert. Dann lassen sich die gleichen Faktoren der Summanden ausklammern und aus den beiden übrigen Faktoren ergibt sich das Neutrale Element zur Disjunktion und entfallen daher. Beispiel:

$$\begin{aligned} f_{DNF} &= \overline{a}\overline{b}c + \overline{a}b\overline{c} + \underbrace{a\overline{b}\overline{c} + ab\overline{c}}_{\text{ähnlich}} \\ &= \overline{a}\overline{b}c + \overline{a}b\overline{c} + a(\overline{b}\overline{c} + b\overline{c}) \\ &= \overline{a}\overline{b}c + \overline{a}b\overline{c} + a\overline{c}(\overline{b} + b) \\ &= \overline{a}\overline{b}c + \overline{a}b\overline{c} + a\overline{c} \cdot 1 \\ f_{DF} &= \overline{a}\overline{b}c + \overline{a}b\overline{c} + a\overline{c} \end{aligned}$$

Die Form der Konjunktiven Kanonischen Normalform lässt sich analog über das neutrale Element der Konjunktion zur Minimierung nutzen. Es werden Faktoren gesucht, deren Summanden ähnlich sind. Beispiel:

$$\begin{aligned} f_{KNF} &= (\overline{a} + \overline{b} + c) \cdot (\overline{a} + b + \overline{c}) \cdot \underbrace{(a + \overline{b} + \overline{c}) \cdot (a + b + \overline{c})}_{\text{ähnlich}} \\ &= (\overline{a} + \overline{b} + c) \cdot (\overline{a} + b + \overline{c}) \cdot (a + (\overline{b} + \overline{c}) \cdot (b + \overline{c})) \\ &= (\overline{a} + \overline{b} + c) \cdot (\overline{a} + b + \overline{c}) \cdot (a + \overline{c} + (\overline{b} \cdot b)) \\ &= (\overline{a} + \overline{b} + c) \cdot (\overline{a} + b + \overline{c}) \cdot (a + \overline{c} + 0) \\ f_{KF} &= (\overline{a} + \overline{b} + c) \cdot (\overline{a} + b + \overline{c}) \cdot (a + \overline{c}) \end{aligned}$$

Um die Suche nach den neutralen Elementen für uns Menschen zu erleichtern wurde eine Grafische Methode zur Funktionsminimierung entwickelt. Kanonische Normalformen können mit Karnaugh-Veitch-Diagrammen[4] grafisch dargestellt und reduziert werden. Die mathematische Grundlage dafür bilden die neutralen Elemente und deren Umformung, wie zuvor beschrieben.

Je nach Anzahl der Eingangsvariablen eines Funktionsterms wird das KV-Diagramm erweitert. Abbildung 4.3 zeigt den Aufbau der KV-Diagramme für eins, zwei, drei und vier Variablen. Wie in der Abbildung zu sehen, sind die Bereiche des Diagramms passend für alle Kombinationen der Variablen aufgeteilt.

Mit dem Buchstaben der Variable und einem „ | “ wird definiert in welchem Bereich die Variable den Wert 1 hat. In allen anderen Bereichen hat die Variable den Wert 0. So

[4] Karnaugh-Veitch-Diagramm kurz KV-Diagramm

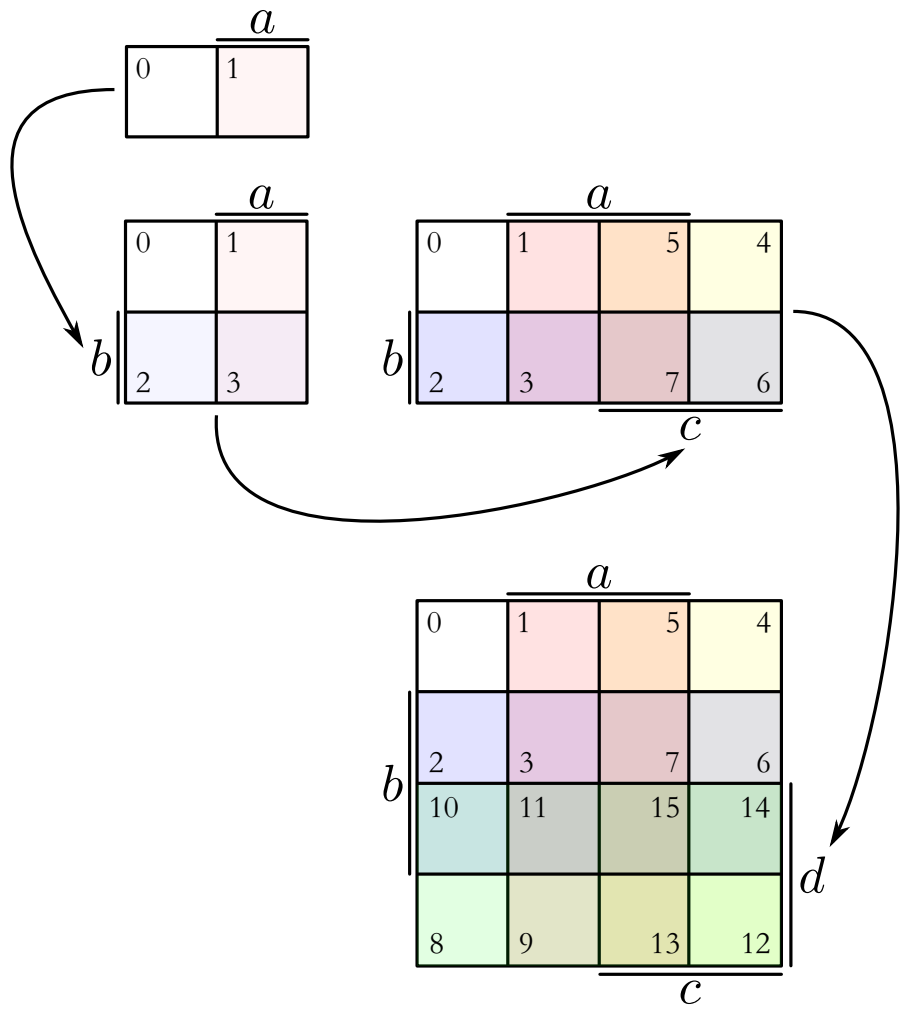


Abbildung 4.3: Entwicklung der KV-Diagramme für eins bis vier Variablen

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
a	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
b	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
c	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
d	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
f	1	0	0	0	1	0	1	1	1	0	0	0	1	0	1	1

Tabelle 4.2: SBT zur Referenz für eine günstige Anordnung der 16 Kombinationen von a, b, c, d

können die Funktionswerte aus der Schaltbelegungstabelle direkt in ein Diagramm übernommen werden. Das Diagramm kann also als weitere Darstellungsform einer logischen Funktion benutzt werden.

Damit die Übertragung aus der Schaltbelegungstabelle möglichst einfach wird, sollte die Belegung wie im folgenden Beispiel dargestellt gewählt werden. Die Kombinationen werden durchnummeriert anhand der Dualzahlendarstellung der jeweiligen Kombination. Die Variable mit der lexikografisch niedrigsten Ziffer bekommt die niedrigste Stellenwertigkeit (vgl. Tabelle 4.2). So können die Funktionswerte der nummerierten Kombinationen nach den Zahlen in Abbildung 4.3 korrekt übertragen werden ohne viele Überlegungen machen zu müssen.

Abbildung 4.4 zeigt das Diagramm zu Tabelle 4.2.

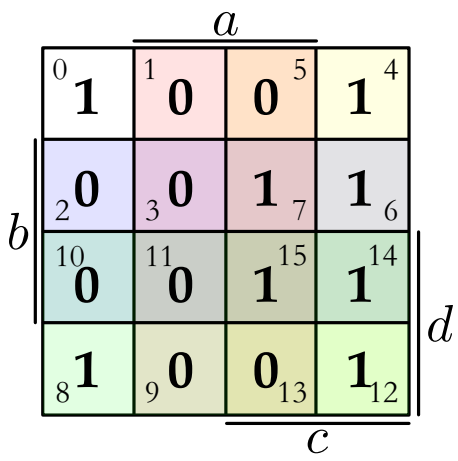


Abbildung 4.4: Ausgefülltes Diagramm zu Tabelle 4.2

Felder, die sich durch Negation einer einzigen Variablen unterscheiden, werden benachbart genannt. Benachbarte Felder sind teilweise sofort im planaren Raum der Diagramme als Felder in direkter Nachbarschaft ersichtlich. Beim Diagramm für drei Variablen kommt es zusätzlich zu einer Ringbildung der Seiten. Damit sind im KV_3-Diagramm die Felder 0 mit 4 und 2 mit 6 benachbart.

Im KV_4-Diagramm ist der linke Rand mit dem rechten und der obere Rand mit dem unteren miteinander benachbart. Im KV_4-Diagramm sind also, neben den offensichtlichen, die Felder 0 mit 4, 2 mit 6, 10 mit 14, 8 mit 12, 0 mit 8, 1 mit 9, 5 mit 13 und 4 mit 12 benachbart. Die Abbildung 4.5 veranschaulicht die weniger offensichtlichen Nachbarschaften.

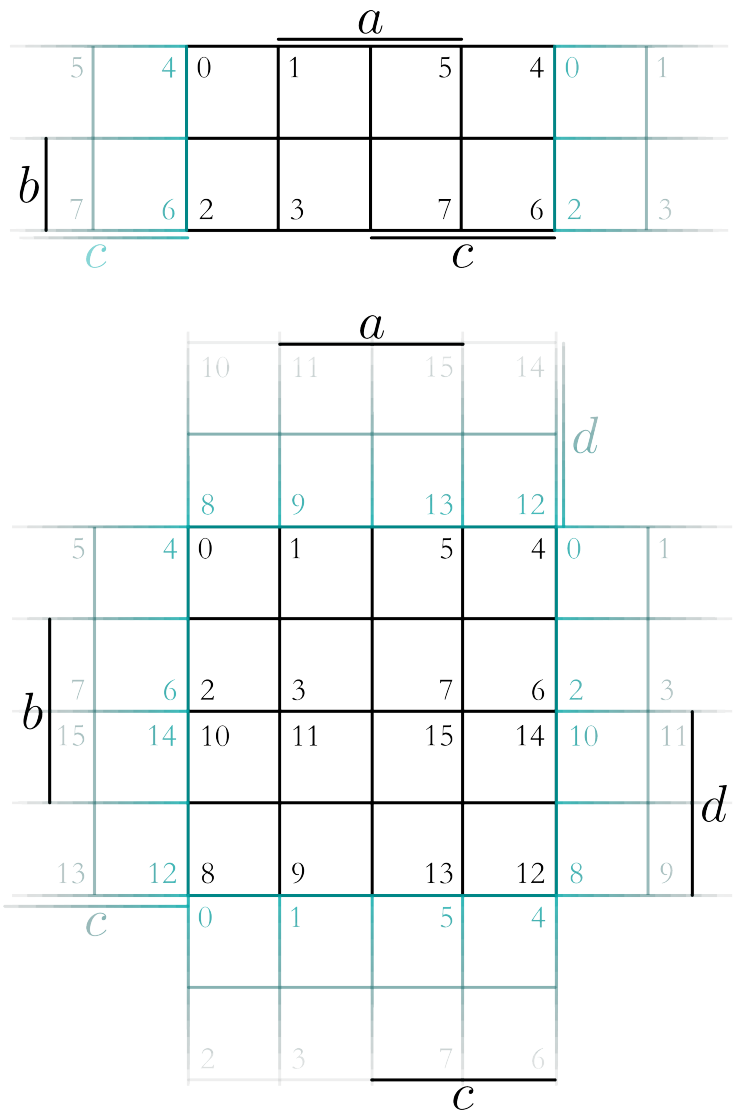


Abbildung 4.5: durch Aneinanderreihung der Diagramme sind die weniger offensichtlichen Nachbarschaften angedeutet.

Mit Nachbarschafts-Eigenschaft können Funktionen mittels des Diagramms minimiert werden. Laut der zuvor in diesem Abschnitt vorgestellten algebraischen Regeln dürfen sich um eine Minimierung zweier Terme zu ermöglichen die beiden Terme ausschließlich in der Negation einer einzigen Variablen unterscheiden. Da aus den mit dem Wert 1

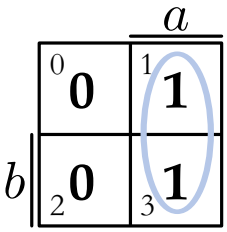
belegten Feldern direkt die Minterme resultieren, können diejenigen Minterme zusammengefasst werden, deren Felder benachbart sind. Analog gilt dies für die Maxterme, welche aus den mit 0 belegten Feldern resultieren.

Beispiele:

- Beispiel: Die Funktion $f = ab + a\bar{b}$ soll minimiert werden. Algebraisch kann folgende Vereinfachung vorgenommen werden:

$$\begin{aligned} f &= ab + a\bar{b} \\ &= a(b + \bar{b}) \\ f &= a \end{aligned}$$

  Die Funktion im KV-Diagramm:

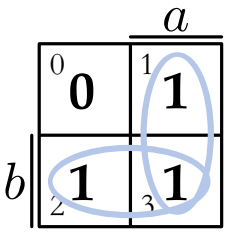


  Aus dem Diagramm kann der aus den beiden Mintermen $m_1 = \bar{b}a$ und $m_3 = ba$ der zusammengefasste Term a direkt abgelesen werden. So ergibt sich ebenfalls die minimierte Formel $f = a$.

- Beispiel: Die Funktion $f = \bar{b}a + b\bar{a} + ba$ soll minimiert werden. Algebraisch kann folgende Vereinfachung vorgenommen werden:

$$\begin{aligned} f &= \bar{b}a + b\bar{a} + ba \\ &= \bar{b}a + b\bar{a} + ba + ba \\ &= \bar{b}a + b(\bar{a} + a) + ba \\ &= a(\bar{b} + b) + b(\bar{a} + a) \\ f &= a + b \end{aligned}$$

  Die Funktion im KV-Diagramm:



  Aus dem Diagramm kann der aus den beiden Mintermen $m_1 = \bar{b}a$ und $m_3 = ba$ der zusammengefasste Term a direkt abgelesen werden. Weiterhin ergibt sich aus

den beiden Mintermen $m_2 = b\overline{a}$ und $m_3 = ba$ der zusammengefasste Term b. So ergibt sich ebenfalls die minimierte Formel $f = b + a$.

Für eine minimale Konjunktive Form wird zuerst die minimale Disjunktive Form der negierten Funktion erstellt. Dies geschieht indem 0en zusammengefasst werden und die zusammengefassten Terme Disjunktiv miteinander verknüpft werden, wie bisher. Um aus der Disjunktiven Form der negierten Funktion zur Konjunktiven Form der Funktion als solche zu kommen muss die erhalgene Disjunktive Form negiert werden. Somit ergibt sich nach De Morgan die minimierte Konjunktive Form der Funktion an sich.

Vorgehensweise für die Vereinfachung in Disjunktiver Form

1. übertrage die SBT oder die Kanonische Disjunktive Normalform in das KV-Diagramm.
2. gruppiere alle 1en in insgesamt möglichst wenigen möglichst großen 1er, 2er, 4er oder 8er Gruppen bis jede 1 abgedeckt ist.
3. lese den Gruppen entsprechend die reduzierten Terme aus.
4. verknüpfe die reduzierten Terme disjunktiv und erhalte die minimierte Disjunktive Form.

4.3.5 Halb- und Volladdierer

Die Addition von Dualzahlen kann über logische Funktionen und damit in logischen hardwaretechnischen Schaltungen abgebildet werden. Ein Halbaddierer berechnet die Summe s von zwei Bits a und b mit Berücksichtigung des Übertrags[5] c. Der Volladdierer berücksichtigt in der Eingabe zusätzlich das carry-Bit.

Halbaddierer SBT

i	0	1	2	3
a	0	1	0	1
b	0	0	1	1
s	0	1	1	1
c	0	0	0	1

$$s = a + b$$
$$c = a \cdot b$$

Volladdierer SBT

i	0	1	2	3	4	5	6	7
c	0	1	0	1	0	1	0	1
b	0	0	1	1	0	0	1	1
a	0	0	0	0	1	1	1	1
s	0	1	1	0	1	0	0	1
c	0	0	0	1	0	1	1	1

$$s_{DNF} = \overline{c}\overline{b}a + \overline{c}b\overline{a} + c\overline{b}\overline{a} + cba$$
$$c_{DNF} = \overline{a}bc + c\overline{b}a + cb\overline{a} + cba$$

[5] Übertrag engl. carry

5 Informationstheorie

In der Informationstheorie geht es um die Definition und quantitative Bewertung von Information. Im Folgenden wird der Austausch von Informationen mittels Nachrichten über Kanäle besprochen. Ein Kommunikationssystem wird durch die Kommunikationspartner, der Kommunikationsrichtung und der Informationsart charakterisiert. Als Kommunikationsparter können Menschen und Maschinen auftreten. Die Kommunikationsrichtung wird allgemein mit simplex oder duplex beschrieben. Ist ein Kommunikationskanal simplex aufgebaut, so erfolgt das Senden der Nachrichten immer vom selben Kommunikationsteilnehmer. Als duplex werden Kommunikationskanäle bezeichnet über die beide Kommunikationspartner gleichzeitig einander Nachrichten senden und empfangen. Zusätzlich gibt es die Bezeichnung halb-duplex, damit ist gemeint, dass nicht gleichzeitig gesendet und empfangen werden kann, sondern die beiden Kommunikationspartner sich mit senden und empfangen von Nachrichten abwechseln. Die Art der Information kann beispielsweise in Daten oder Befehle klassifiziert werden.

Die Information einer Nachricht ist nur dann von Bedeutung, wenn der Empfänger daraus etwas ihm bislang unbekanntes erfährt. Im Fall von Befehlen kann dies zusätzlich zu einer Änderung von Verhaltensweisen führen. Bei der Analyse von Nachrichten wird in drei Betrachtungsweisen unterschieden:

- **Syntax** ist die äußere Form in Form von Zeichen und grammatikalischer Regeln der Sprache, in der die Nachricht verfasst ist. Hierbei kann die mathematische Formulierung und quantitative Bewertung untersucht werden.
- **Semantik** ist die Bedeutung und Sinn der Nachricht.
- **Pragmatik** ist die Untersuchung der Auswirkung der Information auf das Verhalten des Empfängers.

Die Informationstheorie befasst sich ausschließlich mit der Syntax. In den folgenden Abschnitten wird näher auf die formale Beschreibung und quantitative Bewertung von Übertragungskanälen eingegangen. Abschließend werden geeignete Codierungsverfahren vorgestellt, die sich aus neuen Erkenntnissen im Hinblick auf Optimierung erschließen.

5.1 Begriffe

5.2 Informationsquellen

Digitale Informationsquellen[1] bilden Nachrichten mit Zeichen aus einem Alphabet. Die Informationserzeugung ist aus Sicht des Empfängers ein stochastischer Prozess. Dabei kann die Wahrscheinlichkeit, dass ein bestimmtes Zeichen gesendet wird von vorn herein angegeben werden. Damit kann eine Quelle mit n unterschiedlichen Zeichen a_i aus dem Alphabet $A = \{a_i \mid i \in \{1, \ldots, n\}\}$ und der Wahrscheinlichkeit $P(a_i)$, dass das Zeichen a_i gesendet wird, mit folgendem *Wahrscheinlichkeitsfeld* beschrieben werden:

$$X = \begin{pmatrix} a_1 & a_2 & a_3 \\ P(a_1) & P(a_2) & P(a_3) \end{pmatrix}$$

Da beim Senden genau alle Zeichen des Alphabetes zur Auswahl stehen ist die Summe aller Zeichenwahrscheinlichkeiten 1.

$$\sum_{i=1}^{n} P(a_i) = 1$$

Die Eigenschaften von Quellen können folgendermaßen beschrieben werden:

- Eine *stationäre Quelle* charakterisiert eine Quelle als zeitunabhängig. Das heißt über alle Zeitpunkte hinweg ist die Zeichenwahrscheinlichkeit eines jeden Zeichens konstant.
- Quellen ohne Gedächtnis werden als *unabhängige Quellen* bezeichnet. Dabei gilt, dass die Wahrscheinlichkeit eines Zeichens unabhängig von jedem anderen Zeichen zu betrachten ist. Für die Wahrscheinlichkeit, dass ein Zeichen a_j auf das Zeichen a_i folgt gilt dabei:
$$P(a_i, a_j) = P(a_i) \cdot P(a_j)$$

Eine stationäre unabhängige Quelle ist also eine Quelle, die zeichenunabhängig und zeitunabhängig Zeichen sendet.

5.2.1 Entscheidbarkeit und Entropie von Quellen

Die absolute Entscheidbarkeit einer Quelle ist abzulesen an der Anzahl Entscheidungsfragen[2], die gestellt werden müssen um ein gesendetes Zeichen zu erfragen. Für die absolute Entscheidbarkeit $H_0(X)$ von einer Quelle X, deren Alphabet n viele unterschiedliche Zeichen hat, gilt:

$$H_0(X) = ld(n)$$

[1] Digitale Informaitonsquellen = Diskrete Informationsquellen

[2] wikipedia: „Die Entscheidungsfrage (auch: Ja/nein-Frage, Satzfrage) ist ein Typ von Fragesatz. Entscheidungsfragen sind die Fragen, auf die man nur mit ja oder mit nein antworten kann."

Der Informationsgehalt einer Quelle wird durch die Entropie bestimmt. Die Entropie der Quelle lässt sich mit dem Informationsgehalt eines jeden Zeichns berechnen. Der Informationsgehalt $H(a_i)$ eines Zeichens a_i ist:

$$H(a_i) = ld\left(\frac{1}{P(a_i)}\right)\left[\frac{bit}{Zeichen}\right]$$

Hinweis: wegen $0 < P(a_i) \leq 1$ gilt $\frac{1}{P(a_i)} \geq 1$ und $H(a_i) \geq 0$
Die Definition des Informationsgehaltes eines Zeichens $H(a_i)$ ist praktisch anwendbar, da damit die Zeichenwahrscheinlichkeit berücksichtigt wird. Der Logarithmus wird wegen der Entscheidbarkeit gezogen.

Die Entropie einer Quelle ist definiert mit:

$$\begin{aligned} H(X) &= P(a_1) \cdot H(a_1) + P(a_2) \cdot H(a_2) + \ldots + P(a_n) \cdot H(a_n) \\ &= \sum_{i=1}^{n} P(a_i) \cdot H(a_i) \\ &= \sum_{i=1}^{n} P(a_i) \cdot ld\left(\frac{1}{P(a_i)}\right)\left[\frac{bit}{Zeichen}\right] \end{aligned}$$

So ergibt sich für eine Quelle G mit gleichwahrscheinlichen Zeichen, dass die Entropie gleich der absoluten Entscheidbarkeit ist.

$$\begin{aligned} H(G) &= \sum_{i=1}^{n} P(a_i) \cdot ld\left(\frac{1}{P(a_i)}\right) \\ &\quad \text{wegen } P(a_i) = P(a_j) = \frac{1}{n} \text{ gilt:} \\ &= \sum_{i=1}^{n} \frac{1}{n} \cdot ld\left(\frac{1}{\frac{1}{n}}\right) \\ &= n \cdot \left(\frac{1}{n} \cdot ld\left(\frac{1}{\frac{1}{n}}\right)\right) \\ &= \frac{n}{n} \cdot ld(n) \\ H(G) &= ld(n) = H_0(G)\left[\frac{bit}{Zeichen}\right] \end{aligned}$$

Wenn eine Quelle Zeichen mit unterschiedlicher Wahrscheinlichkeit erzeugt, lässt sich im Bezug auf die Entropie und die absolute Entscheidbarkeit eine Redundanz feststellen. Die absolute Redundanz R wird berechnet mit:

$$R = H_0 - H\left[\frac{bit}{Zeichen}\right]$$

Die relatve Redundanz r ergibt sich aus:

$$r = \frac{R}{H_0} = \frac{H_0 - H}{H_0}$$

Entropie von Verbundquellen

Eine Quelle kann aus mehreren anderen Quellen zusammengesetzt sein. Das Alphabet der Verbundquelle kann als kartesisches Produkt der Alphabete der Einzelquellen aufgefasst werden. Wird eine Verbundquelle aus einer Quelle mit dem Alphabet A und einer anderen mit dem Alphabet B in dieser Reihenfolge gebildet so gilt für das Alphabet C der Verbundquelle:

$$\begin{aligned} C &= A \times B \\ &= \{a_1, a_2, \ldots, a_n\} \times \{b_1, b_2, \ldots, b_m\} \\ &= \{(a_1, b_1), (a_1, b_2), \ldots, (a_1, b_m), \ldots, (a_n, b_m)\} \end{aligned}$$

Dieses Prinzip kann für k viele Teilquellen verallgemeinert werden:

$$C = A_1 \times A_1 \times A_1 \times \ldots \times A_k$$

Handelt es sich bei der Verbundquelle um eine unabhängige Verbindung der Teilquellen so wird jede Kombination von Zeichen gleichwahrscheinlich generiert. Wird eine Verbundquelle Z aus einer Quelle X mit dem Alphabet A der Länge n und einer anderen Quelle Y mit dem Alphabet B der Länge m in dieser Reihenfolge gebildet so gilt für die Wahrscheinlichkeitsfelder der Teilquellen X und Y, wie bisher:

$$X = \begin{pmatrix} a_1 & a_2 & \ldots & a_n \\ P(a_1) & P(a_2) & \ldots & P(a_n) \end{pmatrix}$$

$$X = \begin{pmatrix} b_1 & b_2 & \ldots & b_m \\ P(b_1) & P(b_2) & \ldots & P(b_m) \end{pmatrix}$$

Für das Wahrscheinlichkeitsfeld der Verbundquelle $Z = X \times Y$ ergibt sich demnach:

$$X = \begin{pmatrix} a_1 b_1 & a_1 b_2 & \ldots & a_n b_m \\ P(a_1, b_1) & P(a_1, b_2) & \ldots & P(a_n, b_m) \end{pmatrix}$$

Die Entropie der Verbundquelle ist also:

$$
\begin{aligned}
H(Z) &= H(X,Y) \\
H(Z) &= \sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i,y_j)\cdot ld\left(\frac{1}{P(x_i,x_j)}\right) \\
&\quad \text{dies lässt sich vereinfachen:} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i)\cdot P(y_j)\cdot ld\left(\frac{1}{P(x_i)\cdot P(x_j)}\right) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i)\cdot P(y_j)\cdot \left(ld\left(\frac{1}{P(x_i)}\right)+ld\left(\frac{1}{P(x_j)}\right)\right) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i)\cdot P(y_j)\cdot ld\left(\frac{1}{P(x_i)}\right)+P(x_i)\cdot P(y_j)\cdot ld\left(\frac{1}{P(x_j)}\right) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i)\cdot P(y_j)\cdot ld\left(\frac{1}{P(x_i)}\right)+\sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i)\cdot P(y_j)\cdot ld\left(\frac{1}{P(x_j)}\right) \\
&= \sum_{i=1}^{n} P(x_i)\cdot ld\left(\frac{1}{P(x_i)}\right)\cdot\sum_{j=1}^{m} P(y_j)+\sum_{j=1}^{m} P(y_j)\cdot ld\left(\frac{1}{P(x_j)}\right)\cdot\sum_{i=1}^{n} P(x_i) \\
&\quad \text{wegen } \sum_{j=1}^{m} P(y_j)=1 \text{ und } \sum_{i=1}^{m} P(x_i)=1 \\
&= \sum_{i=1}^{n} P(x_i)\cdot ld\left(\frac{1}{P(x_i)}\right)+\sum_{j=1}^{m} P(y_j)\cdot ld\left(\frac{1}{P(x_j)}\right) \\
H(Z) &= H(X,Y) = H(X)+H(Y)
\end{aligned}
$$

Allgemein für Verbundquellen mit k vielen unabhängigen Teilquellen gilt:

$$
\begin{aligned}
H(X_1,X_2,\ldots,X_k) &= \sum_{i_1=1}^{n_1}\sum_{i_2=2}^{n_1}\ldots\sum_{i_k=1}^{n_k} P(a_1i_1,a_2i_2,\ldots,a_ki_k)\cdot ld\left(\frac{1}{P(a_1i_1,a_2i_2,\ldots,a_ki_k)}\right) \\
&= H(X_1)+H(X_2)+\ldots+H(X_k)
\end{aligned}
$$

5.2.2 Informationsübertragung

Um allgemein die Informationsübertragung zu betrachten sehen wir die Informations-Quelle und Informations-Senke als neutraler Beobachter. Dabei wird die Informations-Senke ebenfalls als Quelle betrachtet, da sich auf dem Übertragungskanal wegen auftretender Störungen Nachrichten der Ursprünglichen Quelle verändert werden können. Der Übertragungskanal kann aus Perspektive des Beobachters wie eine Verbundquelle aus Informations-Quelle und Informations-Senke betrachtet werden.

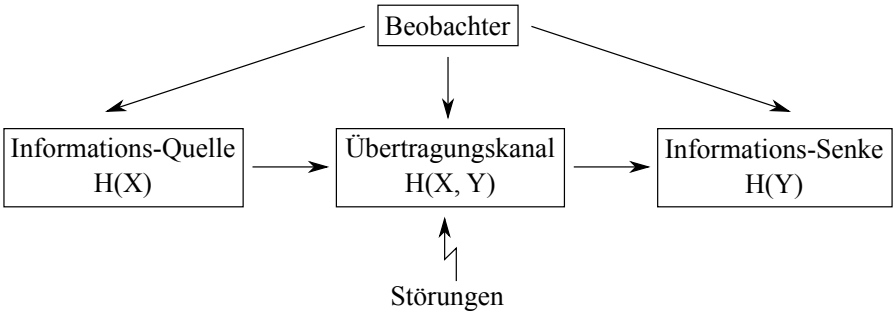


Abbildung 5.1: Betrachtung der Informationsübertragung als neutraler Beobachter

Bei einer störungsfreien Übertragung gilt müssen die Entropien aller im Modell als Quellen betrachteten Informationsträger gleich sein.

$$H(X,Y) = H(X) = H(Y)$$

Abbildung 5.2 zeigt die Beziehung der Entropien im Diagramm.

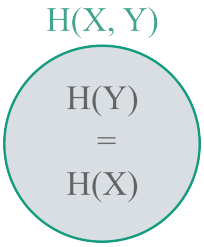


Abbildung 5.2: $H(X)$ gleich $H(Y)$ und damit auch gleich $H(X,Y)$

Bei einer vollständig gestörten Übertragung erhält die Informations-Senke völlig andere Informationen als ursprünglich gesendet wurden. Der Beobachter sieht zwei unabhängige Informationsquelen. In diesem Fall ist die Entropie des Übertragungskanals als Verbundquelle maximal und es gilt

$$H(X,Y) = H(X) + H(Y)$$

Abbildung 5.3 zeigt die Beziehung der Entropien im Diagramm.

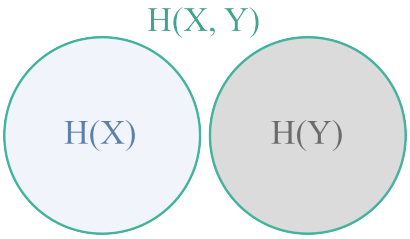


Abbildung 5.3: $H(X,Y)$ ist maximal

Allgemein schwankt die Entropie der Verbundquelle zwischen maximalem und minimalem Wert je nachdem wie stark die Störungen ausfallen.

$$H(X) \leq H(X,Y) \leq H(X) + H(Y)$$

Abbildung 5.4 zeigt die Beziehung der Entropien im Diagramm.

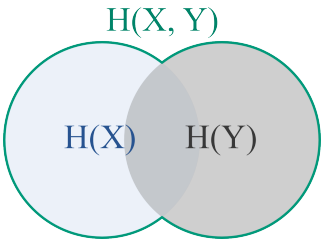


Abbildung 5.4: $H(X,Y)$ zwischen maximalem und minimalem Wert

Die bei der Übertragung verloren gegangene Quellinformation wird *Äquivokation* genannt. Deren Entropie wird mit $H(X\backslash Y)$ bezeichnet.[3] Die Entropie der Äquivokation lässt sich berechnen mit:

$$H(X\backslash Y) = H(X,Y) - H(Y)$$

Information, die bei der Übertragung durch Störungen hinzu gefügt wurde, wird *Irrelevanz* genannt. Die Bezeichnung der Entropie der Irrelevanz ist $H(Y\backslash X)$ und wird wie folgt berechnet:

$$H(Y\backslash X) = H(X,Y) - H(X)$$

Die *Transinformation* ist die Information, welche von der Quelle an den Empfänger tatsächlich übertragen wurde. Die Tansinformation ist die Schnittmenge der Quellinformation mit der empfangenen Information. Die Entropie der Transinformation wird mit $H(X;Y)$ bezeichnet und wie folgt berechnet:

$$\begin{aligned} H(X;Y) &= H(X,Y) - H(X\backslash Y) - H(Y\backslash X) \\ &= H(X) + H(Y) - H(X,Y) \end{aligned}$$

Die Abbildungen 5.5 und 5.6 veranschaulichen die Begriffe Äquivokation, Transinformation und Irrelevanz.

[3] $H(X\backslash Y)$ lies „Entropie von X ohne Y“

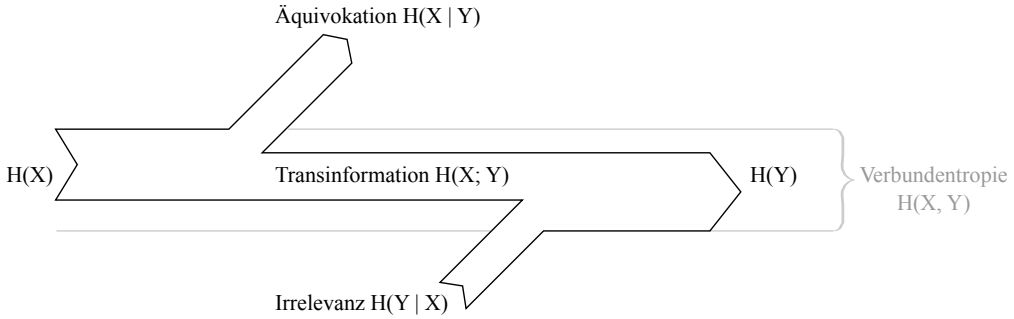


Abbildung 5.5: Betrachtung des Informationskanals mit entsprechenden Teilinformationen Äquivokation, Transinformation und Irrelevanz

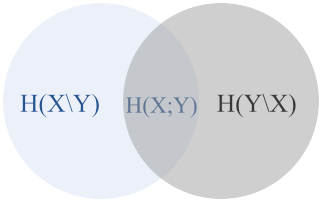


Abbildung 5.6: Beziehungen der Entropien der Teilinformationen

5.3 Gestörte Übertragungskanäle

In diesem Abschnitt wird am Beispiel eines symmetrisch gestörten Übertragungskanals die Analyse der Entropien durchgeführt. Es wird eine Quelle X mit dem Binäralphabet $A = \{a_1, a_2\}$ betrachtet. Die Quelle X und die Senke Y mit dem Alphabet $B = \{b_1, b_2\}$ sind über folgende Wahrscheinlichkeitsfelder allgemein definiert:

$$X = \begin{pmatrix} a_1 & a_2 \\ P(a_1) & 1 - P(a_1) \end{pmatrix} \qquad Y = \begin{pmatrix} b_1 & b_2 \\ P(b_1) & 1 - P(b_1) \end{pmatrix}$$

Bei Fehlerfreier Übertragung entspricht das Zeichen a_1 dem Zeichen b_1 und a_2 entspricht b_2. Die Wahrscheinlichkeit, dass ein Zeichen einem falschen Zeichen zugeordnet wird, weil es sich hier um Binäralphabete handelt, Bitfehlerwahrscheinlichkeit P_f genannt. Dass statt a_1 b_2 beim Empfänger empfangen wird geschieht mit Wahrscheinlichkeit $P(a_1 \mid b_2)$. Die Bezeichnung $P(a_1 \mid b_2)$ beschreibt das Ereignis, dass bei der Übertragung a_1 gesendet und b_2 empfangen wird. Entsprechend gilt für $P(a_1 \mid b_1)$ die Wahrscheinlichkeit, dass wenn a_1 gesendet wird auch b_1 empfangen wird. Da die Bitfehlerwahrscheinlichkeit in diesem Beispiel symmetrisch ist gilt:

$$P(a_1 \mid b_2) = P(a_2 \mid b_1) = P_f$$

Das die unterschiedlichen Ereignisse im Diagramm dargestellt zeigt Abbildung 5.7. Die Pfeile zeigen mit ihrer Beschriftung, wie die Wahrscheinlichkeiten bezeichnet werden und

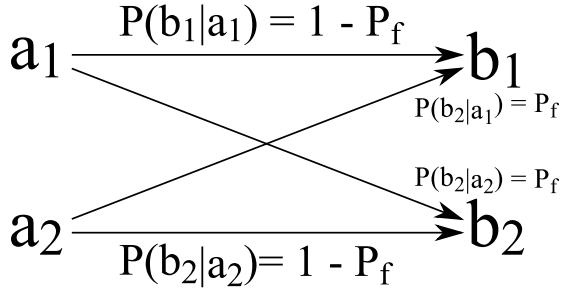


Abbildung 5.7: Wahrscheinlichkeiten bei der Übertragung der Zeichen

wie diese in diesem Beispiel zusammenhängen. Diese stochastischen Ereignisse lassen sich in der Übertragungsmatrix[4] Ü zusamenfassen. Für das Beispiel gilt:

$$\ddot{\mathrm{U}} = [P(Y \mid X)] = \begin{pmatrix} P(b_1 \mid a_1) & P(b_1 \mid a_2) \\ P(b_2 \mid a_1) & P(b_2 \mid a_2) \end{pmatrix}$$

Im Folgenden wird das Wahrscheinlichkeitsfeld der Verbundquelle erstellt, also der Quelle in der wir Informationserzeuger und -Empfänger gemeinsam betrachten. Jede Wahrscheinlichkeit eines Ereignisses berechnet sich aus der Wahrscheinlichkeit, dass die Quelle ein bestimmtes Zeichen sendet und der Wahrscheinlichkeit, dass dieses Zeichen als solches empfangen oder verfälscht wird. Für das Zeichen a_1 werden so die Wahrscheinlichkeiten der beiden Ereignisse berechnet, dass entweder das Zeichen b_1 oder das Zeichen b_2 empfangen wird. Es gibt also folgende vier Ereignisse:

$$\begin{array}{ll} a_1 \text{ wird gesendet und } b_1 \text{ empfangen} & P(a_1) \cdot P(b_1 \mid P(a_1)) \\ a_1 \text{ wird gesendet und } b_2 \text{ empfangen} & P(a_1) \cdot P(b_2 \mid P(a_1)) \\ a_2 \text{ wird gesendet und } b_2 \text{ empfangen} & P(a_2) \cdot P(b_2 \mid P(a_2)) \\ a_2 \text{ wird gesendet und } b_1 \text{ empfangen} & P(a_2) \cdot P(b_1 \mid P(a_2)) \end{array}$$

Als Beobachter gilt für den Kanal K also das Wahrscheinlichkeitsfeld:

$$K = \begin{pmatrix} a_1 b_1 & a_2 b_2 & a_1 b_2 & a_2 b_1 \\ P(a_1)P(b_1 \mid P(a_1)) & P(a_2)P(b_2 \mid a_2) & P(a_1)P(b_2 \mid P(a_1)) & P(a_2)P(b_1 \mid P(a_2)) \end{pmatrix}$$

Das Wahrscheinlichkeitsfeld kann auch als Verbundmatrix V geschrieben werden:

$$\begin{aligned} V &= \begin{pmatrix} P(a_1, b_1) & P(a_1, b_2) \\ P(a_2, b_1) & P(a_2, b_2) \end{pmatrix} \\ &= \begin{pmatrix} P(a_1)P(b_1 \mid P(a_1)) & P(a_1)P(b_2 \mid P(a_1)) \\ P(a_2)P(b_1 \mid P(a_2)) & P(a_2)P(b_2 \mid P(a_2)) \end{pmatrix} \end{aligned}$$

[4] Die Übertragungsmatrix wird auch Rauschmatrix genannt.

Aus der Verbundmatrix lassen sich die Wahrscheinlichkeiten der Quellenzeichen über die Summe der Zeilen rückberechnen. Dies gilt, da die Ereignisse je Spalte alle dem Ereignis, dass ein bestimmtest Zeichen gesendet wurde, zuzuordnen sind. Ebenso gilt, dass die Wahrscheinlichkeiten für die empfangenen Zeichen sich über die Summe der jeweiligen Spalten ergeben.

Die Entropie einer Quelle ist maximal, wenn jedes Zeichen des Alphabetes gleich wahrscheinlich generiert werden. Für die Quelle wird die maximale Entropie $H(X) = 1$ angenommen. Also muss $P(a_1) = \frac{1}{2}$ und $P(a_2) = \frac{1}{2}$ gelten. Da die Störungen symmetrisch wirken muss auch $P(b_1) = \frac{1}{2}$ und $P(b_2) = \frac{1}{2}$ gelten. Damit ergibt sich für die Verbundmatrix:

$$V = \begin{pmatrix} P(a_1, b_1) & P(a_1, b_2) \\ \\ P(a_2, b_1) & P(a_2, b_2) \end{pmatrix} = \begin{pmatrix} \frac{1-P_f}{2} & \frac{P_f}{2} \\ \\ \frac{P_f}{2} & \frac{1-P_f}{2} \end{pmatrix}$$

Welches Zeichen mit welcher Wahrscheinlichkeit empfangen wird, kann über die Spaltensummen berechnet werden. Für dieses Beispiel gilt:

$$\begin{aligned} P(b_1) &= P(a_1, b_1) + P(a_2, b_1) \\ &= \frac{1 - P_f}{2} + \frac{P_f}{2} \\ &= \frac{1}{2} \end{aligned}$$

$$\begin{aligned} P(b_2) &= P(a_1, b_1) + P(a_2, b_1) \\ &= \frac{P_f}{2} + \frac{1 - P_f}{2} \\ &= \frac{1}{2} \end{aligned}$$

Damit ist das Wahrscheinlichkeitsfeld der Senke Y aus diesem Beispiel eindeutig bestimmt:

$$Y = \begin{pmatrix} b_1 & b_2 \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

Mit diesem Ergebnis lässt sich die Entropie der Senke $H(Y)$ als Konstante berechnen:

$$\begin{aligned}
H(Y) &= \sum_{i=1}^{2} P(b_i) \cdot H(b_i) \\
&= \sum_{i=1}^{2} P(b_i) \cdot ld\left(\frac{1}{P(b_i)}\right) \\
&= \sum_{i=1}^{2} \frac{1}{2} \cdot ld\left(\frac{1}{\frac{1}{2}}\right) \\
&= \sum_{i=1}^{2} \frac{1}{2} \cdot 1 \\
&= 1\left[\frac{bit}{Zeichen}\right]
\end{aligned}$$

Das Ergebnis $H(Y) = 1$ ist in sofern nicht überraschend, da die Entropie der Quelle $H(X) = 1$ ist und wir eine symmetrische Störung angenommen haben.

Die Verbundentropie $H(X, Y)$ ist in diesem Beispiel also:

$$\begin{aligned}
H(X,Y) &= \sum_{i=1}^{2}\sum_{j=1}^{2} P(a_i, b_j) \cdot ld\left(\frac{1}{P(a_i, b_j)}\right) \\
&= 2 \cdot \frac{1-P_f}{2} \cdot ld\left(\frac{2}{1-P_f}\right) + 2 \cdot \frac{P_f}{2} \cdot ld\left(\frac{2}{P_f}\right) \\
&= (1-P_f) \cdot ld\left(\frac{2}{1-P_f}\right) + P_f \cdot ld\left(\frac{2}{P_f}\right) \\
&= (1-P_f) \cdot (ld(2) - ld(1-P_f)) + P_f \cdot (ld(2) - ld(P_f)) \\
&= (1-P_f) \cdot (1 - ld(1-P_f)) + P_f \cdot (1 - ld(P_f)) \\
&= (1-P_f) - (1-P_f)ld(1-P_f) + P_f - P_f ld(P_f) \\
&= 1 - (1-P_f)ld(1-P_f) - P_f ld(P_f)
\end{aligned}$$

Die Transinformation ergibt sich aus:

$$\begin{aligned}
H(X;Y) &= H(X) + H(Y) - H(X,Y) \\
&= 1 + 1 - (1 - (1-P_f)ld(1-P_f) - P_f ld(P_f)) \\
&= 1 + 1 - 1 + (1-P_f)ld(1-P_f) + P_f ld(P_f)) \\
&= 1 + (1-P_f)ld(1-P_f) + P_f ld(P_f))
\end{aligned}$$

Für die Äquivokation folgt:

$$\begin{aligned} H(X\backslash Y) &= H(X,Y) - H(Y) \\ &= 1 - (1-P_f)ld(1-P_f) - P_f ld(P_f) - 1 \\ &= -(1-P_f)ld(1-P_f) - P_f ld(P_f) \end{aligned}$$

Für die Irrelevanz gilt:

$$\begin{aligned} H(Y\backslash X) &= H(X,Y) - H(X) \\ &= 1 - (1-P_f)ld(1-P_f) - P_f ld(P_f) - 1 \\ &= -(1-P_f)ld(1-P_f) - P_f ld(P_f) \end{aligned}$$

Praktischer Weise ist die Transinformation für $P_f = 1$ und $P_f = 0$ maximal. Dies ist sinnvoll, da bei völliger Störung ($P_f = 1$) jedes Bit invertiert wird. Um die ursprüngliche Information zu erhalten reicht es das Bitmuster zu invertieren.

5.4 Konstruktion von binären Optimalcodes

5.4.1 Fano-Code

Konstruktionsmethode:

1. Sortiere abfallend die zu codierenden Zeichen nach deren Wahrscheinlichkeiten.
2. Teile die Gruppe der Zeichen in zwei Gruppen, wobei die Summe der Wahrscheinlichkeiten der einen Gruppe möglichst der Summe der Wahrscheinlichkeiten der anderen Gruppe entspricht. Einer Gruppe wird die 0 der anderen die 1 als erste Stelle im Codewort zugeordnet.
3. Wiederhole Schritt 2 mit den entstandenen Teilgruppen, wenn die jeweilige Teilgruppe mehr als ein Zeichen enthält. Andernfalls ist dem einzigen Zeichen der Gruppe das entstandene Codewort zuzuordnen.

Beispiel:

$$X = \begin{pmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{16} & \frac{1}{16} \end{pmatrix}$$

- Die ersten beiden Gruppen sind $\{a_1, a_2\}$ und $\{a_3, \ldots, a_7\}$, da

$$\sum_{i \in \{a_1, a_2\}} P(a_i) = \frac{1}{2} = \sum_{i \in \{a_3, \ldots, a_7\}} P(a_i)$$

- $\{a_1, a_2\}$ wird als erste Ziffer 0 und $\{a_3, \ldots, a_7\}$ als erste Ziffer 1 zugeordnet.

- $\{a_1, a_2\}$ wird aufgeteilt in a_1 und a_2.
- a_1 erhält 0 und a_2 1 als zweite Ziffer. Damit gilt:

$$a_1 \mapsto 00$$

$$a_2 \mapsto 01$$

- $\{a_3, \ldots, a_7\}$ wird aufgeteilt in $\{a_3, a_4\}$ und $\{a_5, a_6, a_7\}$, da

$$\sum_{i \in \{a_3, a_4\}} P(a_i) = \frac{1}{4} = \sum_{i \in \{a_5, a_6, a_7\}} P(a_i)$$

- $\{a_3, a_4\}$ wird 0 und $\{a_5, a_6, a_7\}$ als zweite Ziffer 1 zugeordnet.
- $\{a_3, a_4\}$ wird aufgeteilt in a_3 und a_4.
- a_3 erhält 0 und a_2 1 als dritte Ziffer. Damit gilt:

$$a_3 \mapsto 100$$

$$a_4 \mapsto 101$$

- $\{a_5, a_6, a_7\}$ wird aufgeteilt in a_5 und $\{a_6, a_7\}$, da

$$P(a_5) = \frac{1}{8} = P(a_6) + P(a_7)$$

- a_5 erhält 0 und $\{a_6, a_7\}$ 1 als dritte Ziffer. Damit gilt:

$$a_5 \mapsto 110$$

- $\{a_6, a_7\}$ wird aufgeteilt in a_6 und a_7.
- a_6 erhält 0 und a_7 1 als vierte Ziffer. Damit gilt:

$$a_6 \mapsto 1110$$

$$a_7 \mapsto 1111$$

Dieselbe Konstruktionsmethode aufgeschrieben in einer Tabelle:

a_i	$P(a_i)$	1. Aufteilung 1.Stelle	2. Aufteilung 2. Stelle	3. Aufteilung 3. Stelle	4. Aufteilung 4. Stelle	Code
a_1	$\frac{1}{4}$	0	0			00
a_2	$\frac{1}{4}$	0	1			01
a_3	$\frac{1}{8}$	1	0	0		100
a_4	$\frac{1}{8}$	1	0	1		111
a_5	$\frac{1}{8}$	1	1	0		110
a_6	$\frac{1}{16}$	1	1	1	0	1110
a_7	$\frac{1}{16}$	1	1	1	1	1111

5.4.2 Huffman-Code

Konstruktionsmethode:

1. Sortiere abfallend die zu codierenden Zeichen nach deren Wahrscheinlichkeiten.
2. Das Zeichen mit der niedrigsten Wahrscheinlichkeit erhält als erste Stelle eine 0. Das Zeichen mit der zweitniedrigsten Wahrscheinlichkeit erhält als erste Stelle eine 1. Die Stellen werden bei diesem Verfahren von rechts nach links angehängt.
3. Fasse das Zeichen mit der niedrigsten Wahrscheinlichkeit und das Zeichen mit der zweitniedrigsten Wahrscheinlichkeit in einer Gruppe zusammen.
4. Ordne der Gruppe die Summe der Wahrscheinlichkeiten der Zeichen zu, die der Gruppe angehören.
5. Gruppen mit ihren zugeordneten Wahrscheinlichkeiten sind wie Zeichen zu behandeln. Gruppen können ebenso, wie Zeichen, zu größeren Gruppen zusammengefasst werden. Zusammengefassten Gruppen wird ebenso die Summe der Wahrscheinlichkeiten der kleineren Gruppen zugeordnet, aus denen sie bestehen. Wiederhole alle Schritte bis eine einzige große Gruppe übrig ist.

Beispiel:

$$X = \begin{pmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{16} & \frac{1}{16} \end{pmatrix}$$

- Absteigende Sortierung nach Wahrscheinlichkeiten ist

$$((a_1, 1/4), (a_2, 1/4), (a_3, 1/8), (a_4, 1/8), (a_5, 1/8), (a_6, 1/16), (a_7, 1/16))$$

- a_6 wird als erste Stelle 0 und a_7 wird als erste Stelle 1 zugeordnet.

$$a_6 \mapsto 0$$

$$a_7 \mapsto 1$$

- a_6 und a_7 werden in einer Gruppe zusammengefasst. Der Gruppe wird die Wahrscheinlichkeit $1/16 + 1/16 = 1/8$ zugeordnet.
- Absteigende Sortierung nach Wahrscheinlichkeiten ist

$$((a_1, 1/4), (a_2, 1/4), (\{a_6, a_7\}, 1/8)(a_3, 1/8), (a_4, 1/8), (a_5, 1/8))$$

- a_4 erhält als erste Stelle 0 und a_5 erhält als erste Stelle 1.

$$a_4 \mapsto 0$$

$$a_5 \mapsto 1$$

- a_4 und a_5 werden in einer Gruppe zusammengefasst. Der Gruppe wird die Wahrscheinlichkeit $1/8 + 1/8 = 1/4$ zugeordnet.

- Absteigende Sortierung nach Wahrscheinlichkeiten ist

$$((\{a_4, a_5\}, 1/4), (a_1, 1/4), (a_2, 1/4), (\{a_6, a_7\}, 1/8)(a_3, 1/8))$$

- $\{a_6, a_7\}$ erhalten jeweils als zweite Stelle 0 und a_3 erhält als erste Stelle 1.

$$a_6 \mapsto 00$$

$$a_7 \mapsto 01$$

$$a_3 \mapsto 1$$

- $\{a_6, a_7\}$ und a_3 werden in einer Gruppe zusammengefasst. Der Gruppe wird die Wahrscheinlichkeit $1/8 + 1/8 = 1/4$ zugeordnet.
- Absteigende Sortierung nach Wahrscheinlichkeiten ist

$$((\{a_6, a_7, a_3\}, 1/4), (\{a_4, a_5\}, 1/4), (a_1, 1/4), (a_2, 1/4))$$

- a_1 erhält als erste Stelle 0 und a_2 erhält als erste Stelle 1.

$$a_1 \mapsto 0$$

$$a_2 \mapsto 1$$

- a_1 und a_2 werden in einer Gruppe zusammengefasst. Der Gruppe wird die Wahrscheinlichkeit $1/4 + 1/4 = 1/2$ zugeordnet.
- Absteigende Sortierung nach Wahrscheinlichkeiten ist

$$((\{a_1, a_2\}, 1/2), (\{a_6, a_7, a_3\}, 1/4), (\{a_4, a_5\}, 1/4))$$

- $\{a_6, a_7, a_3\}$ erhalten als weitere Stelle 0 und $\{a_4, a_5\}$ erhalten als weitere Stelle 1.

$$a_6 \mapsto 000$$

$$a_7 \mapsto 001$$

$$a_3 \mapsto 01$$

$$a_4 \mapsto 10$$

$$a_5 \mapsto 11$$

- $\{a_6, a_7, a_3\}$ und $\{a_4, a_5\}$ werden in einer Gruppe zusammengefasst. Der Gruppe wird die Wahrscheinlichkeit $1/4 + 1/4 = 1/2$ zugeordnet.
- Absteigende Sortierung nach Wahrscheinlichkeiten ist

$$((\{a_6, a_7, a_3, a_4, a_5\}, 1/2), (\{a_1, a_2\}, 1/2))$$

- $\{a_6, a_7, a_3, a_4, a_5\}$ erhalten als weitere Stelle 0 und $\{a_1, a_2\}$ erhalten als weitere Stelle 1.

$$a_6 \mapsto 0000$$

$$a_7 \mapsto 0001$$

$$a_3 \mapsto 001$$

$$a_4 \mapsto 010$$

$$a_5 \mapsto 011$$

$$a_1 \mapsto 10$$

$$a_2 \mapsto 11$$

- $\{a_6, a_7, a_3, a_4, a_5\}$ und $\{a_1, a_2\}$ werden in einer Gruppe zusammengefasst. Der Gruppe wird die Wahrscheinlichkeit $1/2 + 1/2 = 1$ zugeordnet. Die Konstruktion ist abgeschlossen, da eine einzige Gruppe übrig ist.

Da bei diesem Verfahren kein Codewort erzeugt wird, das im Präfix eines anderen Codewortes enthalten ist, kann beim einlesen einer Zeichenkette eindeutig direkt entschieden werden um welches Zeichen es sich jeweils handelt.

5.4.3 Mittlere Codewortlänge von Optimalcodes

Die mittlere Codewortlänge von Optimalcodes wird sinnvoll über das Mittel der Codewortlängen in Abhängigkeit der Häufigkeit des jeweiligen Codewortes ermittelt. Die mittlere Codewortlänge len_{mid} eines Codes mit den Codewörtern cw aus der Menge C aller Codeworter ist:

$$len_{mid} = \sum_{cw \in C} P(cw) \cdot len(cw)$$

6 Nachwort

Vielen Dank dafür, dass Du Dir die Zeit nimmst die paar Zeilen des Nachwortes zu lesen. Ich hoffe du hattest Erfolg beim Lernen und Verstehen der Kapitel.

Ich habe dieses Dokument in Eigeninitiative verfasst und hoffe, dass außer mir auch viele andere mit diesem Dokument lernen können. Die gesamte Zusammenstellung hat ungefähr einen Monat gedauert. Wenn dieses Dokument Dir weiterhelfen konnte und Du mir gerne ein Bier oder eine Pizza dafür ausgeben möchtest, kannst Du das sehr gerne tun, indem Du per PayPal auf die Addresse

`weissk@hochschule-trier.de`

einen Betrag Deiner Wahl spendest. Ich bin ein armer Student, wie Du wahrscheinlich auch und freue mich übere jede kleine Zuwendung. Es lebe das Crowd-Funding! `<(^_^)>`

Dieses Dokument ist, wie ihr dem Vorwort entnehmen könnt open-source. Bitte unterstütze open-source Projekte und spende für Projekte, die Dir gefallen oder beteilige Dich am besten an einem oder stelle selbst eines zur Verfügung.

Kajetan Weiß, Trier den 24. Februar 2014