Überblick zu Kapitel 3 von „Einführung in die Kanalcodierung"

Dieses Kapitel behandelt **Faltungscodes** (englisch: *Convolutional Codes*), die erstmals 1955 von **Peter Elias** beschrieben wurden **[Eli55].** Während bei den linearen Blockcodes (siehe Kapitel 1) und den Reed–Solomon–Codes (siehe Kapitel 2) die Codewortlänge jeweils n ist, basiert die Theorie der Faltungscdes auf „semi–infiniten" Informations– und Codesequenzen. Ebenso liefert die Maximum–Likelihood–Decodierung mittels des Viterbi–Algorithmuses per se die gesamte Sequenz.

Im Einzelnen werden in diesem Kapitel behandelt:

- Wichtige *Definitionen* für Faltungscodes: Coderate, Gedächtnis, Einflusslänge, freie Distanz
- *Gemeinsamkeiten* und *Unterschiede* zu den linearen Blockcodes,
- *Generatormatrix* und *Übertragungsfunktionsmatrix* eines Faltungscodes,
- *gebrochen–rationale Übertragungsfunktionen* für systematische Faltungscodes,
- Beschreibung mit *Zustandsübergangsdiagramm* und *Trellisdiagramm*,
- *Terminierung* und *Punktierung* von Faltungscodes,
- *Decodierung* von Faltungscodes ⇒ *Viterbi–Algorithmus*,
- *Gewichtsfunktionen* und Näherungen für die *Bitfehlerwahrscheinlichkeit*.

Geeignete Literatur: [BCJR74] – [Böch15] – [Bos98] – [Bos99] – [CT06] – [Eli55] – [Fri96] – [Hag88] – [Hag90] – [JW93] – [JZ99] – [KM+08] – [Liv10] – [Sha48] – [Vit67]

Die grundlegende Theorie wird auf 52 Seiten dargelegt. Dazu beinhaltet das Kapitel noch 104 Grafiken, 14 Aufgaben und neun Zusatzaufgaben mit insgesamt 101 Teilaufgaben, sowie ein Lernvideo (LV) und sieben Interaktionsmodule (IM):

- **Galoisfeld: Eigenschaften und Anwendungen** (LV zu den Grundlagen,Gesamtdauer 39:10)
- **Digitales Filter** (IM zu Kap. 3.1)
- **Zur Verdeutlichung der grafischen Faltung** (IM zu Kap. 3.2)
- **Korrelationskoeffizient und Regressionsgerade** (IM zu Kap. 3.4)
- **Viterbi–Empfänger** (IM zu Kap. 3.4)
- **M–PSK und Union Bound** (IM zu Kap. 3.5)
- **Diskrete Fouriertansformation** (IM zu den Grundlagen)
- **Komplementäre Gaußsche Fehlerfunktionen** (IM zu den Grundlagen)

Voraussetzungen und Definitionen

Wir betrachten in diesem Kapitel eine unendlich lange binäre Informationssequenz $\underline{u}$ und unterteilen diese in Informationsblöcke $\underline{u}_i$ zu je k Bit. Man kann diesen Sachverhalt wie folgt formalisieren:

$$\underline{u} = (\underline{u}_1, \underline{u}_2, \ldots, \underline{u}_i, \ldots) \quad \text{mit} \quad \underline{u}_i = \left(u_i^{(1)}, u_i^{(2)}, \ldots, u_i^{(k)}\right),$$

$$u_i^{(j)} \in \mathrm{GF}(2) \quad \text{für} \quad 1 \le j \le k \quad \Rightarrow \quad \underline{u}_i \in \mathrm{GF}(2^k).$$

Im Englischen bezeichnet man eine solche Sequenz ohne negative Indizes als *semi–infinite*.

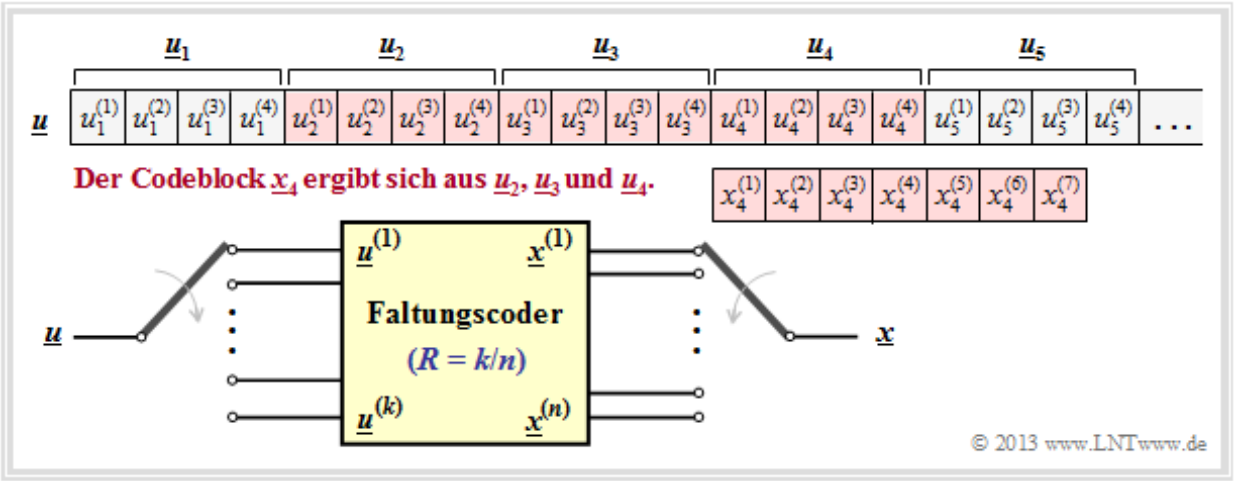
Definition: Bei einem **binären Faltungscode** (englisch: *Binary Convolutional Code*) wird zu dem Taktzeitpunkt i ein Codewort $\underline{x}_i$ bestehend aus n Codebits ausgegeben:

$$\underline{x}_i = \left(x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(n)}\right) \in \mathrm{GF}(2^n).$$

Dieses ergibt sich entsprechend

- den k Bit des aktuellen Informationsblockes $\underline{u}_i$, und
- den m vorherigen Informationsblöcken $\underline{u}_{i-1}$, ... , $\underline{u}_{i-m}$.

Die folgende Grafik verdeutlicht diesen Sachverhalt für die Parameter $k = 4$, $n = 7$, $m = 2$ und $i = 4$. Die $n = 7$ zum Zeitpunkt $i = 4$ erzeugten Codebits $x_4^{(1)}$, ... , $x_4^{(7)}$ können (direkt) von den $k \cdot (m + 1) = 12$ rot markierten Informationsbits abhängen und werden durch Modulo–2–Additionen erzeugt.



Gelb eingezeichnet ist zudem ein (n, k)–Faltungscodierer. Zu beachten ist, dass sich der Vektor $\underline{u}_i$ und die Sequenz $\underline{u}^{(i)}$ grundlegend unterscheiden. Während $\underline{u}_i = (u_i^{(1)}, u_i^{(2)}, \ldots, u_i^{(k)})$ die k zum Zeitpunkt i parallel anliegenden Informationsbits zusammenfasst, bezeichnet $\underline{u}^{(i)} = (u_1^{(i)}, u_2^{(i)}, \ldots)$ die (horizontale) Sequenz am i–ten Eingang des Faltungscodierers. Für den Faltungscode gelten folgende Definitionen:

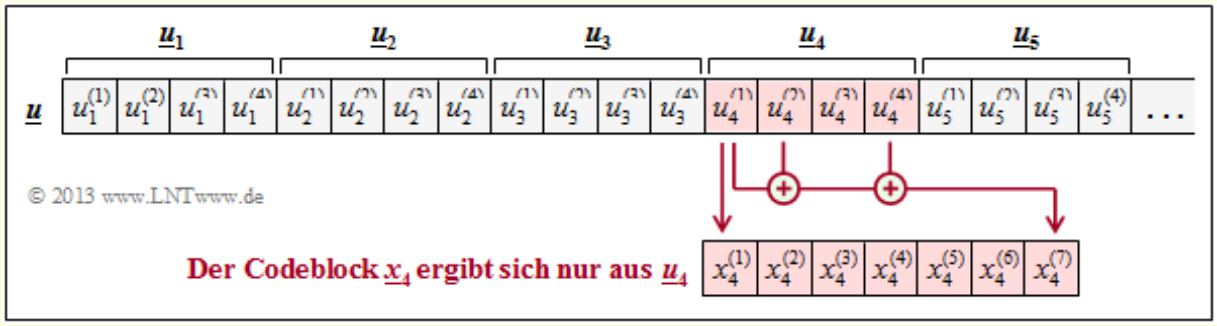
- Die **Coderrate** ergibt sich wie bei den Blockcodes zu $R = k/n$.
- Man bezeichnet m als das **Gedächtnis** (englisch: *Memory*) des Codes und den *Convolutional Code* selbst mit CC(n, k, m).
- Daraus ergibt sich die **Einflusslänge** (englisch: *Constraint Length*) zu $\nu = m + 1$.
- Für $k > 1$ gibt man diese Parameter oft auch in Bit an: $m_{\text{Bit}} = m \cdot k$ bzw. $\nu_{\text{Bit}} = (m + 1) \cdot k$.

Gemeinsamkeiten und Unterschiede gegenüber Blockcodes

Aus der **Definition** auf der letzten Seite ist ersichtlich, dass ein binärer Faltungscode mit $m = 0$ (also ohne Gedächtnis) identisch wäre mit einem binären Blockcode wie in **Kapitel 1** beschrieben. Wir schließen diesen Grenzfall aus und setzen deshalb für das Folgende stets voraus:

- Das Gedächtnis m sei größer oder gleich 1.
- Die Einflusslänge v sei größer oder gleich 2.

Beispiel: Bei einem (7, 4)–Blockcode hängt das Codewort $\underline{x}_4$ nur vom Informationswort $\underline{u}_4$ ab, nicht jedoch von $\underline{u}_2$ und $\underline{u}_3$, wie bei dem **beispielhaften Faltungscodes** (mit $m = 2$) auf der letzten Seite.



Gilt beispielsweise $x_4^{(1)} = u_4^{(1)}$, $x_4^{(2)} = u_4^{(2)}$, $x_4^{(3)} = u_4^{(3)}$, $x_4^{(4)} = u_4^{(4)}$ sowie

$$x_4^{(5)} = u_4^{(1)} + u_4^{(2)} + u_4^{(3)}, \quad x_4^{(6)} = u_4^{(2)} + u_4^{(3)} + u_4^{(4)}, \quad x_4^{(7)} = u_4^{(1)} + u_4^{(2)} + u_4^{(4)},$$

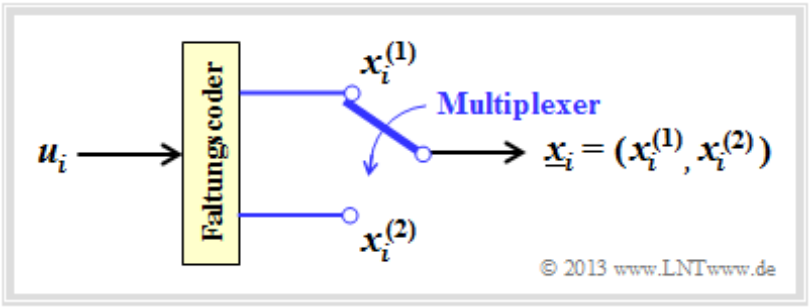
so handelt es sich um einen so genannten *systematischen Hamming–Code* (7, 4, 3) entsprechend **Kapitel 1.3.** In der Grafik sind diese speziellen Abhängigkeiten für $x_4^{(1)}$ und $x_4^{(7)}$ rot eingezeichnet.

In gewisser Weise könnte man auch einen (n, k)–Faltungscode mit Gedächtnis $m \geq 1$ als Blockcode interpretieren, dessen Codeparameter $n' >> n$ und $k' >> k$ allerdings sehr viel größere Werte annehmen müssten als die des vorliegenden Faltungscodes. Aufgrund der großen Unterschiede in der Beschreibung, in den Eigenschaften und insbesondere bei der Decodierung betrachten wir aber Faltungscodes in diesem Lerntutorial als etwas völlig Neues. Hierfür sprechen folgende Gründe:

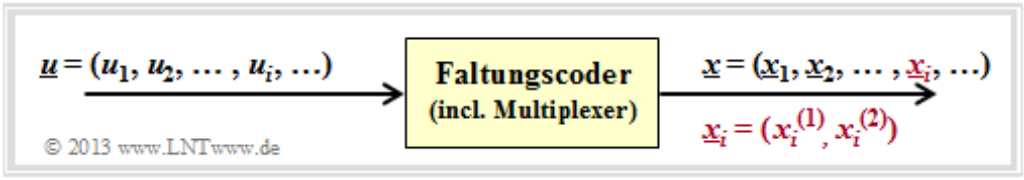
- Ein Blockcodierer setzt Informationsworte der Länge k Bit blockweise in Codeworte mit n Bit um. Der Blockcode ist dabei um so leistungsfähiger, je größer seine Codewortlänge n ist. Bei gegebener Coderate $R = k/n$ erfordert dies auch eine große Informationswortlänge k.
- Dagegen wird die Korrekturfähigkeit eines Faltungscodes im wesentlichen durch sein Gedächtnis m bestimmt. Die Codeparameter k und n werden hier meist sehr klein gewählt (1, 2, 3, ...). Von praktischer Bedeutung sind somit nur ganz wenige und zudem sehr einfache Faltungscodes.
- Auch schon bei kleinen Werten für k und n überführt ein Faltungscoder eine ganze Sequenz von Informationsbits ($k' \to \infty$) in eine sehr lange Sequenz von Codebits ($n' = k'/R$). Ein solcher Code bietet somit oft ebenfalls eine große Korrekturfähigkeit.
- Es gibt effiziente Faltungsdecoder ⇒ **Viterbi–Algorithmus** bzw. **BCJR–Algorithmus.** Diese können *Soft–Decision–Input* (Zuverlässigkeitsinformationen über den Kanal) einfach verarbeiten und liefern auch *Soft–Decision–Output* (Zuverlässigkeitsinformation über das Decodierergebnis).

Rate–1/2–Faltungscodierer (1)

Die Grafik zeigt einen ($n = 2$, $k = 1$)–Faltungscodierer. Zum Taktzeitpunkt i liegt das Informationsbit u_i am Codereingang an und es wird ein 2–Bit–Codeblock $\underline{x}_i = (x_i^{(1)}, x_i^{(2)})$ ausgegeben.



Unter Berücksichtigung der (halb–unendlich) langen Informationssequenz $\underline{u}$ ergibt sich folgendes Modell:



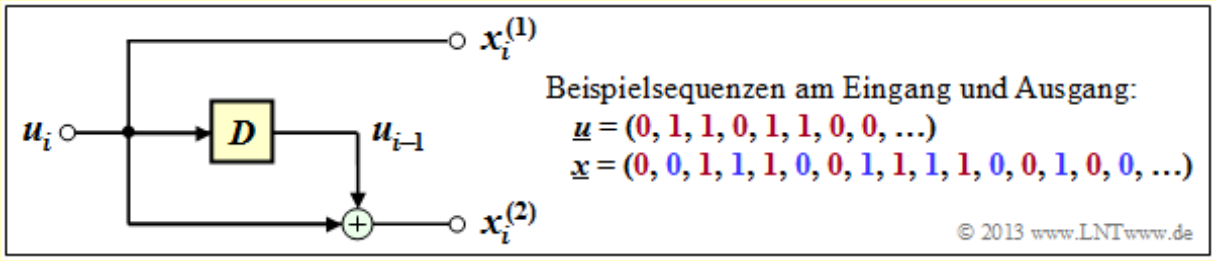
Um aus einem einzigen Informationsbit u_i zwei Codebits $x_i^{(1)}$ und $x_i^{(2)}$ generieren zu können, muss der Faltungscodierer mindestens ein Speicherelement beinhalten:

$$k = 1, n = 2 \quad \Rightarrow \quad m \geq 1 .$$

Anmerkung: Die beiden Begriffe *Faltungscodierer* und *Faltungscoder* werden in unserem Lerntutorial synonym verwendet und können beliebig ausgetauscht werden. Beide Begriffe bezeichnen die konkrete Umsetzung einer Informationssequenz $\underline{u}$ in eine Codesequenz $\underline{x}$.

Die Begriffe *Faltungscodierer* und *Faltungscode* sollte man allerdings nicht verwechseln. Unter einem Faltungscode CC (k, n, m) $\Rightarrow$ $R = k/n$ versteht man die Menge aller möglichen Codesequenzen $\underline{x}$, die mit diesem Code unter Berücksichtigung aller möglichen Informationssequenzen $\underline{u}$ (am Eingang) generiert werden kann. Es gibt verschiedene Faltungscodierer, die den gleichen Faltungscode realisieren.

Beispiel: Nachfolgend ist ein Faltungscodierer für die Parameter $k = 1$, $n = 2$ und $m = 1$ dargestellt. Das gelbe Quadrat kennzeichnet ein Speicherelement. Dessen Beschriftung D ist von *Delay* abgeleitet.



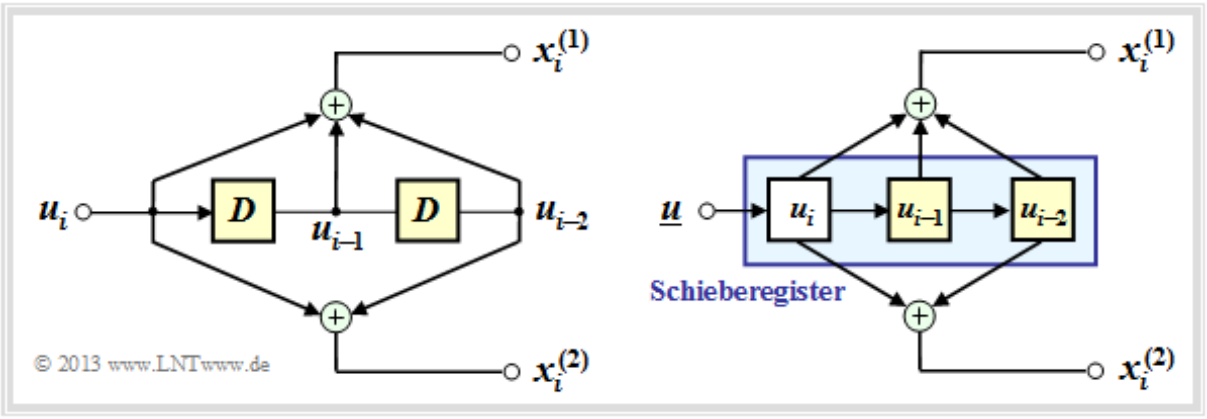
Es handelt sich hier um einen *systematischen* Faltungscodierer, gekennzeichnet durch $x_i^{(1)} = u_i$. Der zweite Ausgang liefert $x_i^{(2)} = u_i + u_{i-1}$. In der beispielhaften Ausgangssequenz nach *Multiplexing* sind alle $x_i^{(1)}$ rot und alle $x_i^{(2)}$ blau beschriftet.

Rate–1/2–Faltungscodierer (2)

Nachfolgend sehen Sie links das Ersatzschaltbild eines (n = 2, k = 1)–Faltungscodierers, aber nun mit m = 2 Speicherelementen. Die beiden Informationsbits lauten:

$$x_i^{(1)} = u_i + u_{i-1} + u_{i-2}\,,$$
$$x_i^{(2)} = u_i + u_{i-2}\,.$$

Wegen $x_i^{(1)} \neq u_i$ handelt es sich hier nicht um einen systematischen Code.



Rechts angegeben ist eine Realisierungsform dieses Coders. Man erkennt:

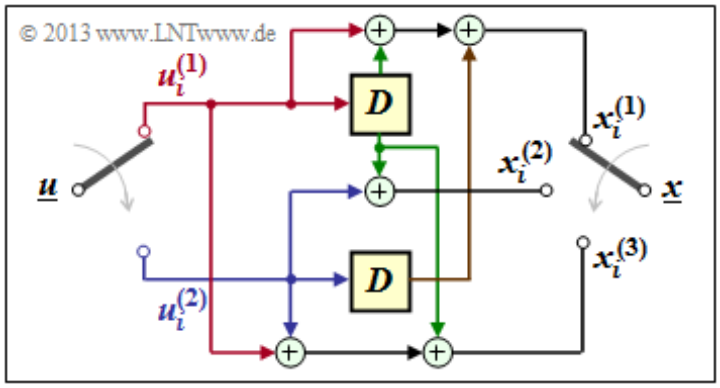
- Die Informationssequenz $\underline{u}$ wird in einem Schieberegister der Länge $L = m + 1 = 3$ abgelegt.
- Zum Taktzeitpunkt i beinhaltet das linke Speicherelement das aktuelle Informationsbit u_i, das zu den nächsten Taktzeitpunkten jeweils um eine Stelle nach rechts verschoben wird.
- Aus der Anzahl der gelben Quadrate ergibt sich wieder das Gedächtnis m = 2 des Coders.

Aus den beiden Darstellungen wird deutlich, dass $x_i^{(1)}$ und $x_i^{(2)}$ jeweils als der Ausgang eines Digitalen Filters über dem Galoisfeld GF(2) interpretiert werden kann, wobei beide Filter parallel mit der gleichen Eingangsfolge $\underline{u}$ arbeiten. Da sich (ganz allgemein) das Ausgangssignal eines Filters aus der **Faltung** des Eingangssignals mit der Filterimpulsantwort ergibt, spricht man von *Faltungscodierung*.

Faltungscodierer mit $k = 2$ Eingängen

Nun betrachten wir einen Faltungscodierer, der aus $k = 2$ Informationsbits $n = 3$ Codebits generiert. Die Informationssequenz $\underline{u}$ wird in Blöcke zu je zwei Bit aufgeteilt. Zum Taktzeitpunkt i liegt am oberen Eingang das Bit $u_i^{(1)}$ an und am unteren Eingang $u_i^{(2)}$. Für die $n = 3$ Codebits zum Zeitpunkt i gilt dann:

$$x_i^{(1)} = u_i^{(1)} + u_{i-1}^{(1)} + u_{i-1}^{(2)} ,$$
$$x_i^{(2)} = u_i^{(2)} + u_{i-1}^{(1)} ,$$
$$x_i^{(3)} = u_i^{(1)} + u_i^{(2)} + u_{i-1}^{(1)} .$$



In der Grafik sind die Info–Bits $u_i^{(1)}$ und $u_i^{(2)}$ rot bzw. blau gekennzeichnet, und die vorherigen Info–Bits $u_{i-1}^{(1)}$ und $u_{i-1}^{(2)}$ grün bzw. braun. Anzumerken ist:

- Das **Gedächtnis** m ist gleich der maximalen Speicherzellenzahl in einem Zweig ⇒ hier $m = 1$.
- Die **Einflusslänge** v ist gleich der Summe aller Speicherelemente ⇒ hier $v = 2$.
- Alle Speicherelemente seien zu Beginn der Codierung (*Initialisierung*) auf Null gesetzt.

Der hiermit definierte Code ist die Menge aller möglichen Codesequenzen $\underline{x}$, die sich bei Eingabe aller möglichen Informationssequenzen $\underline{u}$ ergeben. Sowohl $\underline{u}$ als auch $\underline{x}$ seien dabei (zeitlich) unbegrenzt.

Beispiel: Die Informationssequenz sei $\underline{u}$ = (0, 1, 1, 0, 0, 0, 1, 1, ...). Daraus ergeben sich die beiden Teilsequenzen $\underline{u}^{(1)}$ = (0, 1, 0, 1, ...) und $\underline{u}^{(2)}$ = (1, 0, 0, 1, ...). Mit der Festlegung $u_0^{(1)} = u_0^{(2)} = 0$ folgt aus den obigen Gleichungen für die $n = 3$ Codebits:

- im ersten Codierschritt ($i = 1$):

$$x_1^{(1)} = u_1^{(1)} = 0 , \quad x_1^{(2)} = u_1^{(2)} = 1 , \quad x_1^{(3)} = u_1^{(1)} + u_1^{(2)} = 0 + 1 = 1 ,$$

- im zweiten Codierschritt ($i = 2$):

$$x_2^{(1)} = u_2^{(1)} + u_1^{(1)} + u_1^{(2)} = 1 + 0 + 1 = 0 , \quad x_2^{(2)} = u_2^{(2)} + u_1^{(1)} = 0 + 0 = 0 ,$$
$$x_2^{(3)} = u_2^{(1)} + u_2^{(2)} + u_1^{(1)} = 1 + 0 + 0 = 1 ,$$

- im dritten Codierschritt ($i = 3$):

$$x_3^{(1)} = u_3^{(1)} + u_2^{(1)} + u_2^{(2)} = 0 + 1 + 0 = 1 , \quad x_3^{(2)} = u_3^{(2)} + u_2^{(1)} = 0 + 1 = 1 ,$$
$$x_3^{(3)} = u_3^{(1)} + u_3^{(2)} + u_2^{(1)} = 0 + 0 + 1 = 1 ,$$

- und schließlich im vierten Codierschritt ($i = 4$):

$$x_4^{(1)} = u_4^{(1)} + u_3^{(1)} + u_3^{(2)} = 1 + 0 + 0 = 1 , \quad x_4^{(2)} = u_4^{(2)} + u_3^{(1)} = 1 + 0 = 1 ,$$
$$x_4^{(3)} = u_4^{(1)} + u_4^{(2)} + u_3^{(1)} = 1 + 1 + 0 = 0 .$$

Damit lautet die Codesequenz nach dem Multiplexer: $\underline{x}$ = (0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, ...).

Definition und Interpretation der Teilmatrizen G_0, ... , G_m

Entsprechend den Ausführungen in **Kapitel 1.4** lässt sich das Codewort $\underline{x}$ eines linearen Blockcodes aus dem Informationswort $\underline{u}$ und der Generatormatrix $\mathbf{G}$ in einfacher Weise ermitteln:

$$\underline{x} = \underline{u} \cdot \mathbf{G}.$$

Dabei gilt:

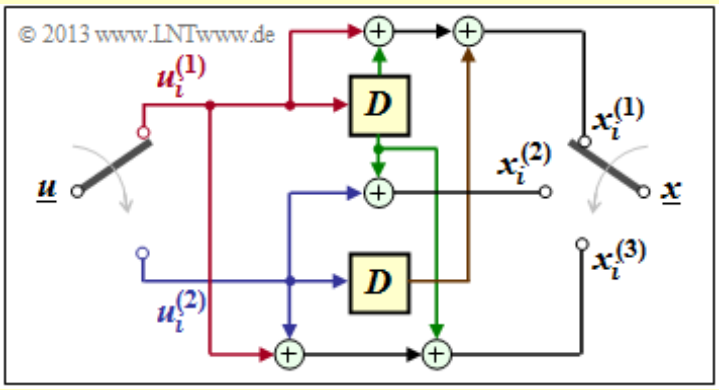
- Die Vektoren $\underline{u}$ und $\underline{x}$ haben die Länge k (Bitanzahl eines Informationswortes) bzw. n (Bitanzahl eines Codewortes) und $\mathbf{G}$ besitzt die Dimension $k \times n$ (k Zeilen und n Spalten).
- Bei Faltungscodierung bezeichnen dagegen $\underline{u}$ und $\underline{x}$ Sequenzen mit $k' \to \infty$ und $n' \to \infty$. Deshalb wird auch die Generatormatrix $\mathbf{G}$ in beiden Richtungen unendlich weit ausgedehnt sein.

Als Vorbereitung für die Einführung der Generatormatrix $\mathbf{G}$ auf der nächsten Seite definieren wir $m + 1$ Teilmatrizen, jeweils mit k Zeilen und n Spalten, die wir mit $\mathbf{G}_l$ bezeichnen, wobei $0 \le l \le m$ gilt.

Definition: Ist das Matrizenelement $\mathbf{G}_l(\kappa, j) = 1$, so sagt dies aus, dass das Codebit $x_i^{(j)}$ durch das Informationsbit $u_{i-l}^{(\kappa)}$ beeinflusst wird. Andernfalls ist dieses Matrixelement gleich 0.

Diese Definition wird nun an einem Beispiel verdeutlicht.

Beispiel: Wir betrachten wiederum den Faltungscodierer gemäß nebenstehender Grafik mit den folgenden Codebits:



$$x_i^{(1)} = u_i^{(1)} + u_{i-1}^{(1)} + u_{i-1}^{(2)},$$
$$x_i^{(2)} = u_i^{(2)} + u_{i-1}^{(1)},$$
$$x_i^{(3)} = u_i^{(1)} + u_i^{(2)} + u_{i-1}^{(1)}.$$

Wegen der Gedächtnisordnung $m = 1$ wird dieser Codierer durch die beiden Teilmatrizen $\mathbf{G}_0$ und $\mathbf{G}_1$ charakterisiert:

$$\mathbf{G}_0 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}, \quad \mathbf{G}_1 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix}.$$

Diese Matrizen sind wie folgt zu interpretieren:

- Erste Zeile von $\mathbf{G}_0$, rote Pfeile: $u_i^{(1)}$ beeinflusst sowohl $x_i^{(1)}$ als auch $x_i^{(3)}$, nicht jedoch $x_i^{(2)}$.
- Zweite Zeile von $\mathbf{G}_0$, blaue Pfeile: $u_i^{(2)}$ beeinflusst $x_i^{(2)}$ und $x_i^{(3)}$, aber nicht $x_i^{(1)}$.
- Erste Zeile von $\mathbf{G}_1$, grüne Pfeile: $u_{i-1}^{(1)}$ beeinflusst alle drei Coderausgänge.
- Zweite Zeile von $\mathbf{G}_1$, brauner Pfeil: $u_{i-1}^{(2)}$ beeinflusst nur $x_i^{(1)}$.

Generatormatrix eines Faltungscodierers mit Gedächtnis *m*

Mit den Teilmatrizen $\mathbf{G}_0$, ... , $\mathbf{G}_m$ lassen sich die n Codebits zum Zeitpunkt i wie folgt ausdrücken:

$$\underline{x}_i = \sum_{l=0}^{m} \underline{u}_{i-l} \cdot \mathbf{G}_l = \underline{u}_i \cdot \mathbf{G}_0 + \underline{u}_{i-1} \cdot \mathbf{G}_1 + \ldots + \underline{u}_{i-m} \cdot \mathbf{G}_m .$$

Hierbei sind folgende vektorielle Größen zu berücksichtigen:

$$\underline{u}_i = \left(u_i^{(1)}, u_i^{(2)}, \ldots, u_i^{(k)}\right), \quad \underline{x}_i = \left(x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(n)}\right).$$

Betrachtet man die bei $i = 1$ beginnenden und sich zeitlich bis ins Unendliche erstreckenden Sequenzen

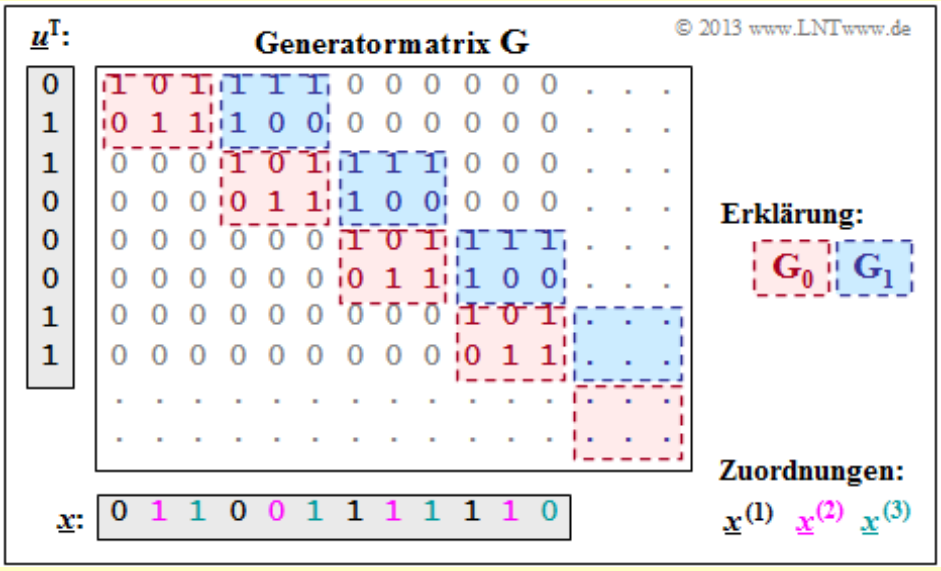
$$\underline{u} = \left(\underline{u}_1, \underline{u}_2, \ldots, \underline{u}_i, \ldots\right), \quad \underline{x} = \left(\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_i, \ldots\right),$$

so kann dieser Zusammenhang durch die Matrixgleichung $\underline{x} = \underline{u} \cdot \mathbf{G}$ ausgedrückt werden. Hierbei ist für die Generatormatrix $\mathbf{G}$ zu setzen:

$$\mathbf{G} = \begin{pmatrix} \mathbf{G}_0 & \mathbf{G}_1 & \mathbf{G}_2 & \cdots & \mathbf{G}_m & & \\ & \mathbf{G}_0 & \mathbf{G}_1 & \mathbf{G}_2 & \cdots & \mathbf{G}_m & \\ & & \mathbf{G}_0 & \mathbf{G}_1 & \mathbf{G}_2 & \cdots & \mathbf{G}_m \\ & & & \cdots & \cdots & & \cdots \end{pmatrix}.$$

Aus der Gleichung erkennt man sofort das Gedächtnis m des Faltungscodes. Die Parameter k und n sind direkt nicht ablesbar. Sie sind aber durch die Zeilen– und Spaltenzahl der Teilmatrizen $\mathbf{G}_l$ festgelegt.

Beispiel: Mit den zwei Matrizen $\mathbf{G}_0$ und $\mathbf{G}_1$ – siehe **letztes Beispiel** – erhält man die rechts skizzierte Matrix $\mathbf{G}$.



Anzumerken ist:

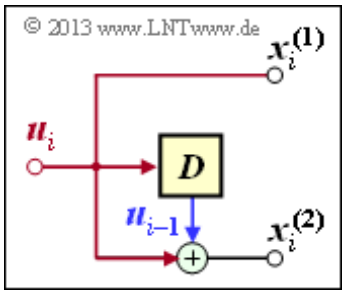
- Die Generatormatrix $\mathbf{G}$ erstreckt sich nach unten und nach rechts eigentlich bis ins Unendliche. Explizit dargestellt sind aber nur 8 Zeilen und 12 Spalten.
- Für die zeitlich begrenzte Informationssequenz $\underline{u} = (0, 1, 1, 0, 0, 0, 1, 1)$ ist der gezeichnete Matrixteil ausreichend. Die Codesequenz lautet dann: $\underline{x} = (0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0)$.
- Anhand der Beschriftungsfarben lassen sich die $n = 3$ Codewortstränge ablesen. Das gleiche Ergebnis haben wir (auf anderem Wege) im **Beispiel** am Ende von Kapitel 3.1 erhalten.

$$\underline{x}^{(1)} = (0,0,1,1), \quad \underline{x}^{(2)} = (1,0,1,1), \quad \underline{x}^{(3)} = (1,1,1,0).$$

Generatormatrix für Faltungscodierer der Rate 1/*n*

Wir betrachten nun den Sonderfall $k = 1$, zum einen aus Gründen einer möglichst einfachen Darstellung, aber auch, weil Faltungscodierer der Rate $1/n$ für die Praxis eine große Bedeutung besitzen.

Faltungscodierer mit $k = 1$, $n = 2$ und $m = 1$



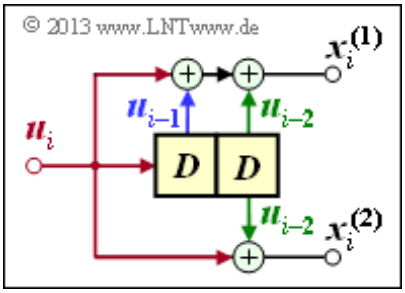
Aus der nebenstehenden Skizze kann abgeleitet werden:

$$\mathbf{G}_0 = \begin{pmatrix} 1 & 1 \end{pmatrix}, \quad \mathbf{G}_1 = \begin{pmatrix} 0 & 1 \end{pmatrix}$$

$$\Rightarrow \quad \mathbf{G} = \begin{pmatrix} 11 & 01 & 00 & 00 & 00 & \cdots \\ 00 & 11 & 01 & 00 & 00 & \cdots \\ 00 & 00 & 11 & 01 & 00 & \cdots \\ 00 & 00 & 00 & 11 & 01 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix}.$$

Für die Eingangssequenz $\underline{u} = (1, 0, 1, 1)$ beginnt die Codesequenz mit $\underline{x} = (1, 1, 0, 1, 1, 1, 1, 0, ...)$. Dieses Ergebnis ist gleich der Summe der Zeilen 1, 3 und 4 der Gewneratormatrix.

Faltungscodierer mit $k = 1$, $n = 2$ und $m = 2$



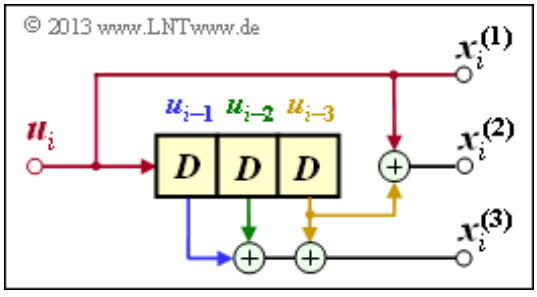
Aufgrund der Gedächtnisordnung $m = 2$ gibt es hier drei Teilmatrizen:

$$\mathbf{G}_0 = \begin{pmatrix} 1 & 1 \end{pmatrix}, \quad \mathbf{G}_1 = \begin{pmatrix} 1 & 0 \end{pmatrix}, \quad \mathbf{G}_2 = \begin{pmatrix} 1 & 1 \end{pmatrix}$$

$$\Rightarrow \quad \mathbf{G} = \begin{pmatrix} 11 & 10 & 11 & 00 & 00 & 00 & \cdots \\ 00 & 11 & 10 & 11 & 00 & 00 & \cdots \\ 00 & 00 & 11 & 10 & 11 & 00 & \cdots \\ 00 & 00 & 00 & 11 & 10 & 11 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \end{pmatrix}.$$

Hier führt die Eingangsssequenz $\underline{u} = (1, 0, 1, 1)$ zur Codesequenz $\underline{x} = (1, 1, 1, 0, 0, 0, 0, 1, ...)$.

Faltungscodierer mit $k = 1$, $n = 3$ und $m = 3$



Wegen $m = 3$ gibt es vier Teilmatrizen der Dimension 1×3:

$$\mathbf{G}_0 = \begin{pmatrix} 1 & 1 & 0 \end{pmatrix}, \quad \mathbf{G}_1 = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix},$$

$$\mathbf{G}_2 = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}, \quad \mathbf{G}_3 = \begin{pmatrix} 0 & 1 & 1 \end{pmatrix}.$$

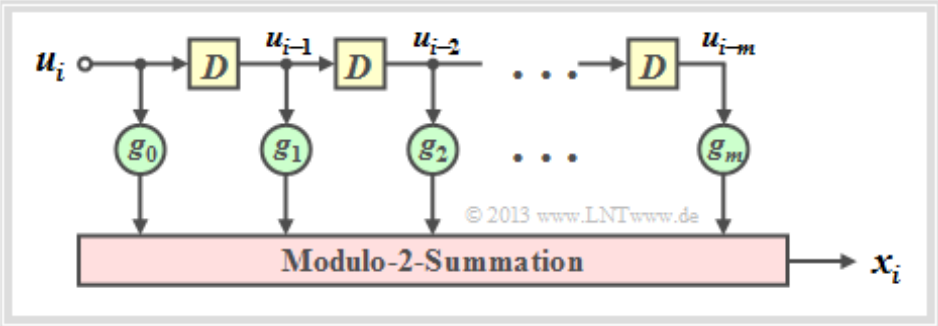
Damit lautet die resultierende Generatormatrix:

$$\mathbf{G} = \begin{pmatrix} 110 & 001 & 001 & 011 & 000 & 000 & 000 & \cdots \\ 000 & 110 & 001 & 001 & 011 & 000 & 000 & \cdots \\ 000 & 000 & 110 & 001 & 001 & 011 & 000 & \cdots \\ 000 & 000 & 000 & 110 & 001 & 001 & 011 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix},$$

und man erhält für $\underline{u} = (1, 0, 1, 1)$ die Codesequenz $\underline{x} = (1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, ...)$.

GF(2)–Beschreibungsformen eines Digitalen Filters (1)

In **Kapitel 3.1** wurde bereits darauf hingewiesen, dass ein Faltungscodierer der Rate $1/n$ durch mehrere Digitale Filter realisiert werden kann, wobei die Filter parallel mit der gleichen Eingangsfolge $\underline{u}$ arbeiten. Bevor wir diese Aussage vertiefen, sollen zuerst die Eigenschaften eines Digitalfilters für das Galoisfeld GF(2) genannt werden.



Die Grafik ist wie folgt zu interpretieren:

- Das Filter besitzt die Impulsantwort $\underline{g} = (g_0, g_1, g_2, \ ... \ , g_m)$, wobei für alle Filterkoeffizienten (mit den Indizes $0 \le l \le m$) gilt: $g_l \in \text{GF(2)} = \{0, 1\}$.
- Die einzelnen Symbole u_i der Eingangsfolge $\underline{u}$ seien ebenfalls binär: $u_i \in \{0, 1\}$. Damit gilt für das Ausgangssymbol zu den Zeitpunkten $i \ge 1$ mit Addition und Multiplikation in GF(2):

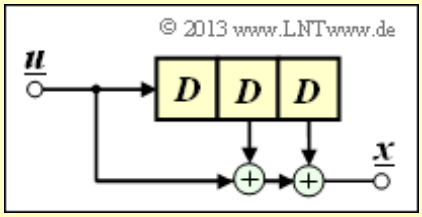
$$x_i = \sum_{l=0}^{m} g_l \cdot u_{i-l} \ .$$

- Dies entspricht der (zeitdiskreten) **Faltungsoperation** (englisch: *Convolution*), gekennzeichnet durch einen Stern. Damit kann für die gesamte Ausgangssequenz geschrieben werden:

$$\underline{x} = \underline{u} * \underline{g} \ .$$

- Wesentlicher Unterschied gegenüber dem **Kapitel 5.2** des Buches „Stochastische Signaltheorie" ist die Modulo–2–Addition (1 + 1 = 0) anstelle der herkömmlichen Addition (1 + 1 = 2).

Beispiel: Die Impulsantwort des dargestellten Digitalen Filters der Ordnung 3 lautet $\underline{g} = (1, 0, 1, 1)$. Die Eingangssequenz dieses Filters sei zeitlich unbegrenzt: $\underline{u} = (1, 1, 0, 0, 0, ...)$.



Damit ergibt sich die (unendliche) Ausgangssequenz $\underline{x}$ im binären Galoisfeld ⇒ GF(2):

$$\underline{x} = (1, 1, 0, 0, 0, ...) * (1, 0, 1, 1) =$$
$$= (1, 0, 1, 1, 0, 0, ...) \oplus (0, 1, 0, 1, 1, 0, ...) = (1, 1, 1, 0, 1, 0, ...).$$

Bei der herkömmlichen Faltung (für reelle Zahlen) hätte dagegen das Ergebnis gelautet:

$$\underline{x} = (1, 1, 1, 2, 1, 0, ...).$$

GF(2)–Beschreibungsformen eines Digitalen Filters (2)

Zeitdiskrete Signale kann man auch durch Polynome bezüglich einer Dummy–Variablen repräsentieren.

Definition: Die zum zeitdiskreten Signal $\underline{x} = (x_0, x_1, x_2, ...)$ gehörige ***D*–Transformierte** lautet:

$$X(D) = x_0 + x_1 \cdot D + x_2 \cdot D^2 + \ ... \ = \sum_{i=0}^{\infty} x_i \cdot D^i .$$

Für diese spezielle Transformation in einen Bildbereich verwenden wir auch die Notation:

$$\underline{x} = (x_0, x_1, x_2, ...) \quad \circ\!\!-\!\!\bullet^{\,D} \quad X(D) = \sum_{i=0}^{\infty} x_i \cdot D^i .$$

In der Literatur wird manchmal $x(D)$ anstelle von $X(D)$ verwendet. Wir schreiben in LNTwww aber alle Bildbereichsfunktionen mit Großbuchstaben, zum Beispiel Fourier–, Laplace– und *D*–Transformation:

$$x(t) \ \circ\!\!-\!\!\bullet \ X(f), \quad x(t) \ \circ\!\!-\!\!\bullet^{\,\rm L} \ X(p), \quad \underline{x} \ \circ\!\!-\!\!\bullet^{\,D} \ X(D).$$

Wir werden nun die *D*–Transformation auch auf die Informationssequenz $\underline{u}$ und die Impulsantwort $\underline{g}$ an. Aufgrund der zeitlichen Begrenzung von $\underline{g}$ ergibt sich die obere Summationsgrenze bei $G(D)$ zu $i = m$:

$$\underline{u} = (u_0, u_1, u_2, ...) \quad \circ\!\!-\!\!\bullet^{\,D} \quad U(D) = \sum_{i=0}^{\infty} u_i \cdot D^i ,$$

$$\underline{g} = (g_0, g_1, \ ..., g_m) \quad \circ\!\!-\!\!\bullet^{\,D} \quad G(D) = \sum_{i=0}^{m} g_i \cdot D^i .$$

Satz: Wie bei allen Spektraltransformationen gilt auch bei der *D*–Transformation im Bildbereich die **Multiplikation**, da die (diskreten) Zeitsignale $\underline{u}$ und $\underline{g}$ durch die **Faltung** verknüpft sind:

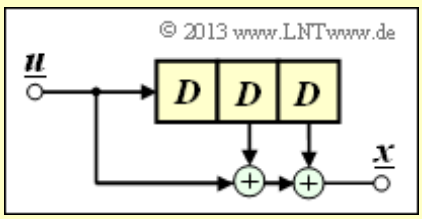
$$\underline{x} = \underline{u} * \underline{g} \quad \circ\!\!-\!\!\bullet^{\,D} \quad X(D) = U(D) \cdot G(D) .$$

Man bezeichnet, wie in der **Systemtheorie** allgemein üblich, auch die *D*–Transformierte $G(D)$ der Impulsantwort $\underline{g}$ als **Übertragungsfunktion** (englisch: *Transfer Function*).

Der (recht einfache) Beweis dieses wichtigen Ergebnisses finden Sie in der Angabe zu **Aufgabe Z3.3.**

Beispiel: Wir betrachten wieder die zeitdiskreten Signale

$$\underline{u} = (1, 1, 0, 0, ...) \quad \circ\!\!-\!\!\bullet^{\,D} \quad U(D) = 1 + D ,$$
$$\underline{g} = (1, 0, 1, 1) \quad \circ\!\!-\!\!\bullet^{\,D} \quad G(D) = 1 + D^2 + D^3 .$$



Wie im **letzten Beispiel:** erhält man auch auf diesem Lösungsweg:

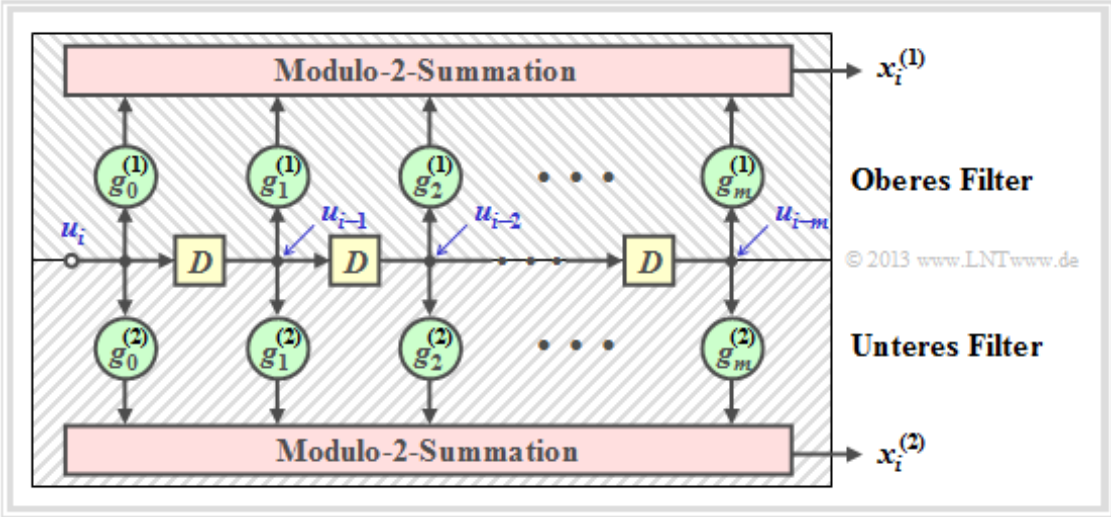
$$X(D) = U(D) \cdot G(D) = (1 + D) \cdot (1 + D^2 + D^3) =$$
$$= 1 + D^2 + D^3 + D + D^3 + D^4 = 1 + D + D^2 + D^4$$
$$\Rightarrow \quad \underline{x} = (1, 1, 1, 0, 1, 0, ...).$$

Die Multiplikation mit D im Bildbereich entspricht im Zeitbereich einer Verschiebung um eine Stelle nach rechts, weshalb man D als *Verzögerungsoperator* (englisch: *Delay Operator*) bezeichnet:

$$W(D) = D \cdot X(D) \quad \bullet\!\!-\!\!\circ^{\,D} \quad \underline{w} = (0, 1, 1, 1, 0, 1, 0, ...).$$

Anwendung der *D*–Transformation auf Rate–1/*n*–Faltungscoder (1)

Wir wenden nun die Ergebnisse der letzten Seite auf einen Faltungscoder an, wobei wir uns zunächst auf den Sonderfall $k = 1$ beschränken. Ein solcher $(n, k = 1)$–Faltungscode lässt sich mit n Digitalen Filtern realisieren, die auf der gleichen Informationssequenz $\underline{u}$ parallel arbeiten. Die Grafik zeigt die Anordnung für den Codeparameter $n = 2 \Rightarrow$ Coderate $R = 1/2$.



Die nachfolgenden Gleichungen gelten für beide Filter gleichermaßen, wobei für das obere Filter $j = 1$ und für das untere Filter $j = 2$ zu setzen ist:

- Die **Impulsantworten** der beiden Filter ergeben sich zu

$$\underline{g}^{(j)} = (g_0^{(j)}, g_1^{(j)}, \ldots, g_m^{(j)}), \quad \text{mit } j \in \{1, 2\}.$$

- Die beiden **Ausgangssequenzen** lauten:

$$\underline{x}^{(j)} = (x_0^{(j)}, x_1^{(j)}, x_2^{(j)}, \ldots) = \underline{u} \cdot \underline{g}^{(j)}, \quad \text{mit } j \in \{1, 2\}.$$

  Hierbei ist berücksichtigt, dass das obere Filter und das untere Filter beide auf der gleichen Eingangssequenz $\underline{u} = (u_0, u_1, u_2, \ldots)$ arbeiten.

- Für die ***D*–Transformierten** der Ausgangssequenzen gilt:

$$X^{(j)}(D) = U(D) \cdot G^{(j)}(D), \quad \text{mit } j \in \{1, 2\}.$$

Auf der nächsten Seite verwenden wir eine kompaktere Schreibweise.

Anwendung der *D*–Transformation auf Rate–1/*n*–Faltungscoder (2)

Um den soeben dargelegten Sachverhalt kompakter darstellen zu können, definieren wir nun folgende vektorielle Größen eines Faltungscodes der Rate $1/n$:

Definition: Die ***D*–Übertragungsfunktionen** der n parallel angeordneten digitalen Filter werden im Vektor $\underline{G}(D)$ zusammengefasst:

$$\underline{G}(D) = \big(G^{(1)}(D), G^{(2)}(D), \ \text{...} \ , G^{(n)}(D)\big) \ .$$

Der Vektor $\underline{X}(D)$ beinhaltet die ***D*–Transformierten** der n Codesequenzen $\underline{x}^{(1)}$, $\underline{x}^{(2)}$, ... , $\underline{x}^{(n)}$:

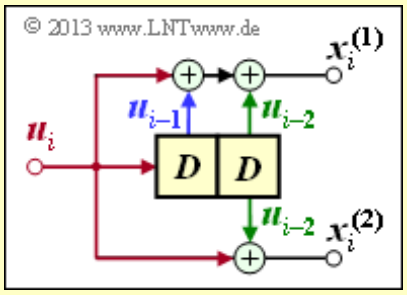
$$\underline{X}(D) = \big(X^{(1)}(D), X^{(2)}(D), \ \text{...} \ , X^{(n)}(D)\big) \ .$$

Damit erhält man die folgende Vektorgleichung:

$$\underline{X}(D) = U(D) \cdot \underline{G}(D) .$$

Aufgrund des Codeparameters $k = 1$ ist $U(D)$ hier keine vektorielle Größe.

Beispiel: Wir betrachten beispielhaft den skizzierten Faltungscode mit den Codeparametern $n = 2$, $k = 1$ und $m = 2$. Für diesen gilt:



$$\underline{g}^{(1)} = (1, 1, 1) \quad \circ\!\!-\!\!\overset{D}{\bullet}\!\!-\!\!\bullet \quad G(D) = 1 + D + D^2 ,$$
$$\underline{g}^{(2)} = (1, 0, 1) \quad \circ\!\!-\!\!\overset{D}{\bullet}\!\!-\!\!\bullet \quad G(D) = 1 + D^2$$
$$\Rightarrow \quad \underline{G}(D) = \big(1 + D + D^2, \ 1 + D^2\big) .$$

Die Informationssequenz sei $\underline{u} = (1, 0, 1, 1)$, was zur D–Transformierten $U(D) = 1 + D^2 + D^3$ führt. Damit erhält man

$$\underline{X}(D) = \big(X^{(1)}(D), \ X^{(2)}(D)\big) = U(D) \cdot \underline{G}(D),$$

wobei

$$X^{(1)}(D) = (1 + D^2 + D^3) \cdot (1 + D + D^2) =$$
$$= 1 + D + D^2 + D^2 + D^3 + D^4 + D^3 + D^4 + D^5 = 1 + D + D^5$$
$$\Rightarrow \quad \underline{x}^{(1)} = (1, 1, 0, 0, 0, 1, 0, 0, \text{...}) ,$$
$$X^{(2)}(D) = (1 + D^2 + D^3) \cdot (1 + D^2) =$$
$$= 1 + D^2 + D^2 + D^4 + D^3 + D^5 = 1 + D^3 + D^4 + D^5$$
$$\Rightarrow \quad \underline{x}^{(2)} = (1, 0, 0, 1, 1, 1, 0, 0, \text{...}) .$$

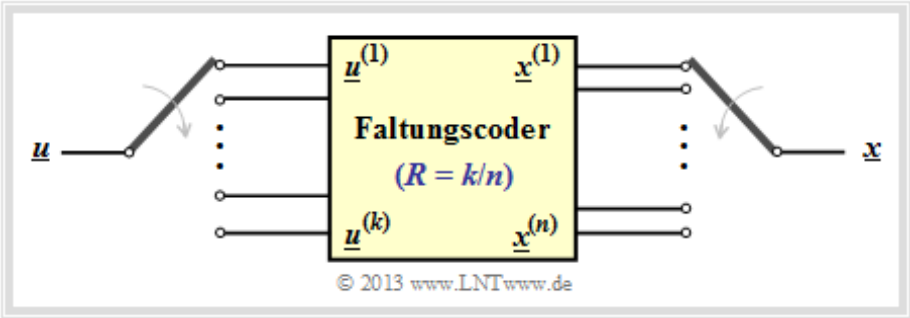
Das gleiche Ergebnis haben wir in der **Aufgabe Z3.1** auf anderem Wege erhalten. Nach dem Multplexen der beiden Sränge erhält man wieder: $\underline{x}$ = (11, 10, 00, 01, 01, 11, 00, 00, ...).

Übertragungsfunktionsmatrix – Transfer Function Matrix (1)

Auf der letzten Seite haben wir gesehen, dass ein Faltungscode der Rate $1/n$ sich am kompaktesten als Vektorgleichung im D–transformierten Bereich beschreiben lässt:

$$\underline{X}(D) = U(D) \cdot \underline{G}(D).$$

Nun erweitern wir das Resultat auf Faltungscodierer mit mehr als einem Eingang $\Rightarrow$ $k \geq 2$ (siehe Grafik).



Um einen Faltungscode der Rate k/n im D–Bereich abbilden zu können, muss die Dimension obiger Vektorgleichung hinsichtlich Eingang und Übertragungsfunktion erhöht werden:

$$\underline{X}(D) = \underline{U}(D) \cdot \mathbf{G}(D),$$

mit folgenden Maßnahmen:

- Aus der skalaren Funktion $U(D)$ wird der Vektor $\underline{U}(D) = (U^{(1)}(D), U^{(2)}(D), \ ... \ , U^{(k)}(D))$.
- Aus dem Vektor $\underline{G}(D)$ wird die $k \times n$–Matrix $\mathbf{G}(D)$, die man als **Übertragungsfunktionsmatrix** bezeichnet (englisch: *Transfer Function Matrix* oder auch *Polynomial Generator Matrix*):

$$\mathbf{G}(D) = \begin{pmatrix} G_1^{(1)}(D) & G_1^{(2)}(D) & \ldots & G_1^{(n)}(D) \\ G_2^{(1)}(D) & G_2^{(2)}(D) & \ldots & G_2^{(n)}(D) \\ \vdots & \vdots & & \vdots \\ G_k^{(1)}(D) & G_k^{(2)}(D) & \ldots & G_k^{(n)}(D) \end{pmatrix}.$$

- Jedes der $k \cdot n$ Matrixelemente $G_i^{(j)}(D)$ mit $1 \leq i \leq k$, $1 \leq j \leq n$ ist ein Polynom über der Dummy–Variablen D im Galoisfeld GF(2), maximal vom Grad m, wobei m das Gedächtnis angibt.
- Für die obige *Übertragungsfunktionsmatrix* kann mit den zu Beginn dieses Kapitels definierten **Teilmatrizen** $\mathbf{G}_0$, ... , $\mathbf{G}_m$ auch geschrieben werden (als Index verwenden wir wieder l):

$$\mathbf{G}(D) = \sum_{l=0}^{m} \mathbf{G}_l \cdot D^l = \mathbf{G}_0 + \mathbf{G}_1 \cdot D + \mathbf{G}_2 \cdot D^2 + \ ... \ + \mathbf{G}_m \cdot D^m.$$

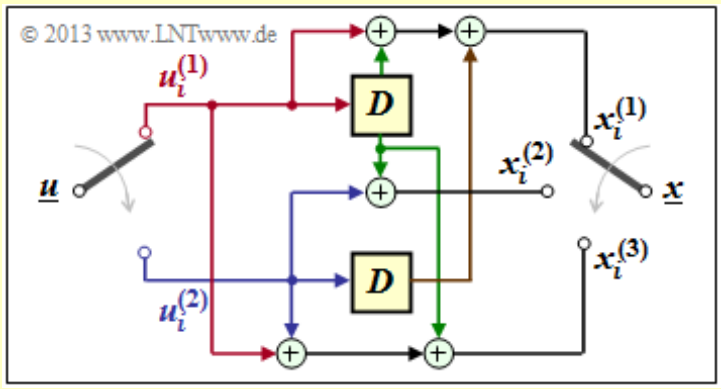
Auf der nächsten Seite werden diese Definitionen und Gesetzmäßigkeiten an einem ausführlichen Beispiel verdeutlicht.

Übertragungsfunktionsmatrix – Transfer Function Matrix (2)

Beispiel: Wir betrachten nun wieder den ($n = 3, k = 2, m = 1$)–Faltungscoder, dessen Teilmatrizen in einem **früheren Beispiel** wie folgt ermittelt wurden:

$$\mathbf{G}_0 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix},$$

$$\mathbf{G}_1 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix}.$$



Wegen $m = 1$ existieren keine Teilmatrizen für $l \geq 2$. Damit lautet die Übertragungsfunktionsmatrix:

$$\mathbf{G}(D) = \mathbf{G}_0 + \mathbf{G}_1 \cdot D = \begin{pmatrix} 1+D & D & 1+D \\ D & 1 & 1 \end{pmatrix}.$$

Die (zeitlich begrenzte) Informationssequenz sei $\underline{u} = (0, 1, 1, 0, 0, 0, 1, 1)$, woraus sich die beiden Eingangsfolgen wie folgt ergeben:

$$\underline{u}^{(1)} = (0, 1, 0, 1) \quad \circ\!\!-\!\!\overset{D}{\bullet} \quad U^{(1)}(D) = D + D^3,$$
$$\underline{u}^{(2)} = (1, 0, 0, 1) \quad \circ\!\!-\!\!\overset{D}{\bullet} \quad U^{(2)}(D) = 1 + D^3.$$

Daraus folgt für den Vektor der D–Transformierten am Coderausgang:

$$\underline{X}(D) = \big(X^{(1)}(D), X^{(2)}(D), X^{(3)}(D)\big) = \underline{U}(D) \cdot \mathbf{G}(D)$$
$$= \big(D + D^3 \quad 1 + D^3\big) \cdot \begin{pmatrix} 1+D & D & 1+D \\ D & 1 & 1 \end{pmatrix}.$$

Damit ergeben sich in den drei Strängen folgende Codesquenzen:

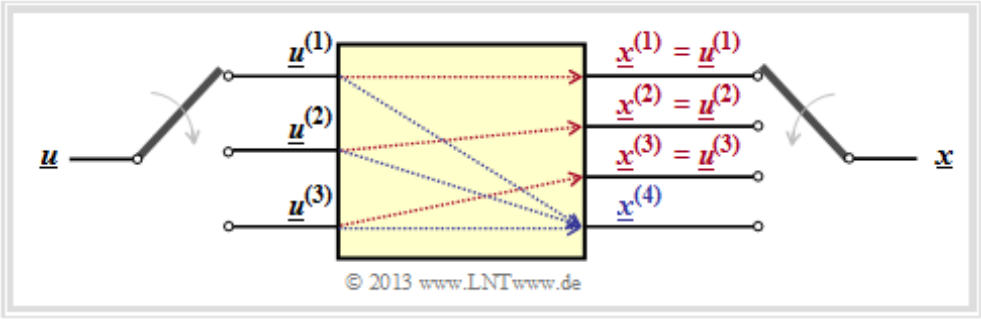
$$X^{(1)}(D) = (D + D^3) \cdot (1 + D) + (1 + D^3) \cdot D =$$
$$= D + D^2 + D^3 + D^4 + D + D^4 = D^2 + D^3$$
$$\Rightarrow \quad \underline{x}^{(1)} = (0, 0, 1, 1, 0, 0, \ldots),$$
$$X^{(2)}(D) = (D + D^3) \cdot D + (1 + D^3) \cdot 1 =$$
$$= D^2 + D^4 + 1 + D^3 = 1 + D^2 + D^3 + D^4$$
$$\Rightarrow \quad \underline{x}^{(2)} = (1, 0, 1, 1, 1, 0, \ldots),$$
$$X^{(3)}(D) = (D + D^3) \cdot (1 + D) + (1 + D^3) \cdot 1 =$$
$$= D + D^2 + D^3 + D^4 + 1 + D^3 = 1 + D + D^2 + D^4$$
$$\Rightarrow \quad \underline{x}^{(3)} = (1, 1, 1, 0, 1, 0, \ldots).$$

Die gleichen Ergebnisse haben wir auf anderen Wegen bereits in vorherigen Beispielen erhalten:

- im Beispiel von **Kapitel 3.1, Seite 4,**
- im Beispiel von **Kapitel 3.2, Seite 2.**

Systematische Faltungscodes (1)

Die Polynomrepräsentation anhand der Übertragungsfunktionsmtrix $\mathbf{G}(D)$ ermöglicht Einblicke in die Struktur eines Faltungscodes. Beispielsweise erkennt man anhand dieser $k \times n$–Matrix, ob es sich um einen **systematischen Code** handelt. Darunter versteht man einen Code, bei dem die Codesequenzen $\underline{x}^{(1)}$, ... , $\underline{x}^{(k)}$ mit den Informationssequenzen $\underline{u}^{(1)}$, ... , $\underline{u}^{(k)}$ identisch sind. Die Grafik zeigt beispielhaft einen systematischen ($n = 4$, $k = 3$)–Faltungscode.



Ein systematischer (n, k)–Faltungscode liegt immer dann vor, wenn die Übertragungsfunktionsmatrix (mit k Zeilen und n Spalten) folgendes Aussehen hat:

$$\mathbf{G}(D) = \mathbf{G}_{\rm sys}(D) = \big[\,\mathbf{I}_k\,;\,\mathbf{P}(D)\big]\,.$$

Hierbei ist folgende Nomenklatur verwendet:

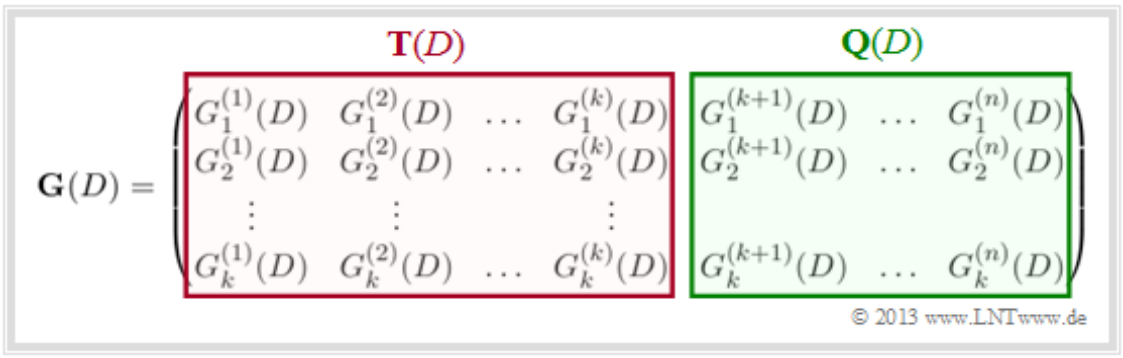
- $\mathbf{I}_k$ bezeichnet eine diagonale Einheitsmatrix der Dimension $k \times k$.
- $\mathbf{P}(D)$ ist eine $k \times (n-k)$–Matrix, wobei jedes Matrixelement ein Polynom in D beschreibt.

Beispiel: Ein systematischer Faltungscode mit den Codeparametern $n = 3$, $k = 2$, $m = 2$ könnte beispielsweise die folgende Übertragungsfunktionsmatrix aufweisen:

$$\mathbf{G}_{\rm sys}(D) = \begin{pmatrix} 1 & 0 & 1+D^2 \\ 0 & 1 & 1+D \end{pmatrix}.$$

Andere systematische Faltungscodes mit gleichem n und gleichem k unterscheiden sich demgegenüber nur durch die beiden Matrixelemente in der letzten Spalte.

Zu jedem (n, k)–Faltungscode mit der Generatormatrix $\mathbf{G}(D)$ kann ein **äquivalenter systematischer Code** gefunden werden, dessen D–Matrix wir mit $\mathbf{G}_{\rm sys}(D)$ benennen.



Auf der nächsten Seite wird gezeigt, wie man von einer beliebigen Matrix $\mathbf{G}(D)$ durch Aufspalten in zwei Teilmatrizen $\mathbf{T}(D)$ und $\mathbf{Q}(D)$ und verschiedene Matrizenoperationen zur Matrix $\mathbf{G}_{\rm sys}(D)$ kommt.

Systematische Faltungscodes (2)

Um von einer Übertragungsfunktionsmatrix $\mathbf{G}(D)$ zur Matrix $\mathbf{G}_{\rm sys}(D)$ eines äquivalenten systematischen Faltungscodes zu kommen, geht man entsprechend der **Grafik** auf der letzten Seite wie folgt vor:

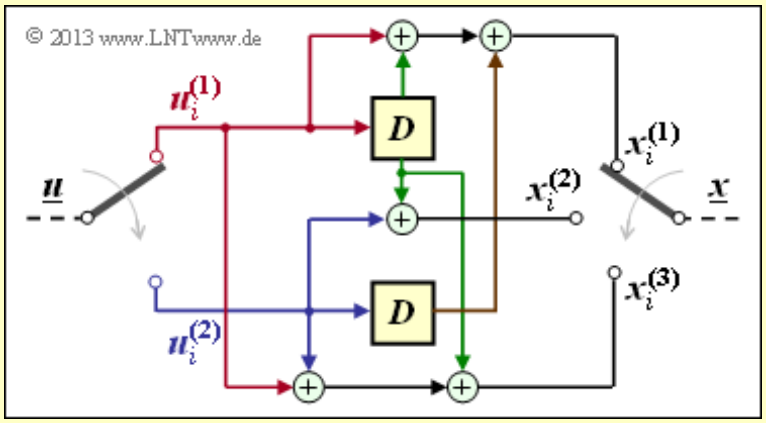
- Man unterteilt die $k\times n$–Matrix $\mathbf{G}(D)$ in eine quadratische Matrix $\mathbf{T}(D)$ mit k Zeilen und k Spalten und bezeichnet den Rest mit $\mathbf{Q}(D)$.
- Anschließend berechnet man die zu $\mathbf{T}(D)$ inverse Matrix $\mathbf{T}^{-1}(D)$ und daraus die Matrix für den äquivanten systematischen Code:

$$\mathbf{G}_{\rm sys}(D) = \mathbf{T}^{-1}(D) \cdot \mathbf{G}(D) .$$

- Da $\mathbf{T}^{-1}(D)\ \cdot \mathbf{T}(D)$ die $k\times k$–Einheitsmatrix $\mathbf{I}_k$ ergibt, kann die Übertragungsfunktionsmatrix des äquivalenten systematischen Codes in der gewünschten Form geschrieben werden:

$$\mathbf{G}_{\rm sys}(D) = \big[\mathbf{I}_k\,;\,\mathbf{P}(D)\big] \quad \text{mit} \quad \mathbf{P}(D) = \mathbf{T}^{-1}(D) \cdot \mathbf{Q}(D).$$

Beispiel: Der auf den letzten Seiten schon häufiger betrachtete Coder der Rate 2/3 ist nicht systematisch, weil zum Beispiel $\underline{x}^{(1)} \neq \underline{u}^{(1)}$, $\underline{x}^{(2)} \neq \underline{u}^{(2)}$ gilt (siehe nebenstehende Coderschaltung).



Man erkennt dies aber auch anhand der Übertragungsfunktionsmatrix:

$$\mathbf{G}(D) = \big[\mathbf{T}(D);\ \mathbf{Q}(D)\big]$$

$$\Rightarrow \quad \mathbf{T}(D) = \begin{pmatrix} 1+D & D \\ D & 1 \end{pmatrix},\ \mathbf{Q}(D) = \begin{pmatrix} 1+D \\ 1 \end{pmatrix}.$$

Die Determinante von $\mathbf{T}(D)$ ergibt sich zu $(1 + D) \cdot 1 + D \cdot D = 1 + D + D^2$ und ist ungleich 0. Somit kann für die Inverse von $\mathbf{T}(D)$ geschrieben werden (Vertauschung der Diagonalelemente!):

$$\mathbf{T}^{-1}(D) = \frac{1}{1+D+D^2} \cdot \begin{pmatrix} 1 & D \\ D & 1+D \end{pmatrix}.$$

Das Produkt $\mathbf{T}(D) \cdot \mathbf{T}^{-1}(D)$ ergibt die Einheitsmatrix $\mathbf{I}_2$, und für die dritte Spalte von $\mathbf{G}_{\rm sys}(D)$ gilt:

$$\mathbf{P}(D) = \mathbf{T}^{-1}(D) \cdot \mathbf{Q}(D) = \frac{1}{1+D+D^2} \cdot \begin{pmatrix} 1 & D \\ D & 1+D \end{pmatrix} \cdot \begin{pmatrix} 1+D \\ 1 \end{pmatrix}$$

$$\Rightarrow \quad \mathbf{P}(D) = \frac{1}{1+D+D^2} \cdot \begin{pmatrix} (1+D)+D \\ D\cdot(1+D)+(1+D) \end{pmatrix} = \frac{1}{1+D+D^2} \cdot \begin{pmatrix} 1 \\ 1+D^2 \end{pmatrix}$$

$$\Rightarrow \quad \mathbf{G}_{\rm sys}(D) = \begin{pmatrix} 1 & 0 & \frac{1}{1+D+D^2} \\ 0 & 1 & \frac{1+D^2}{1+D+D^2} \end{pmatrix}.$$

Es ist noch zu klären, wie das Filter einer solchen gebrochen–rationalen Übertragungsfunktion aussieht.

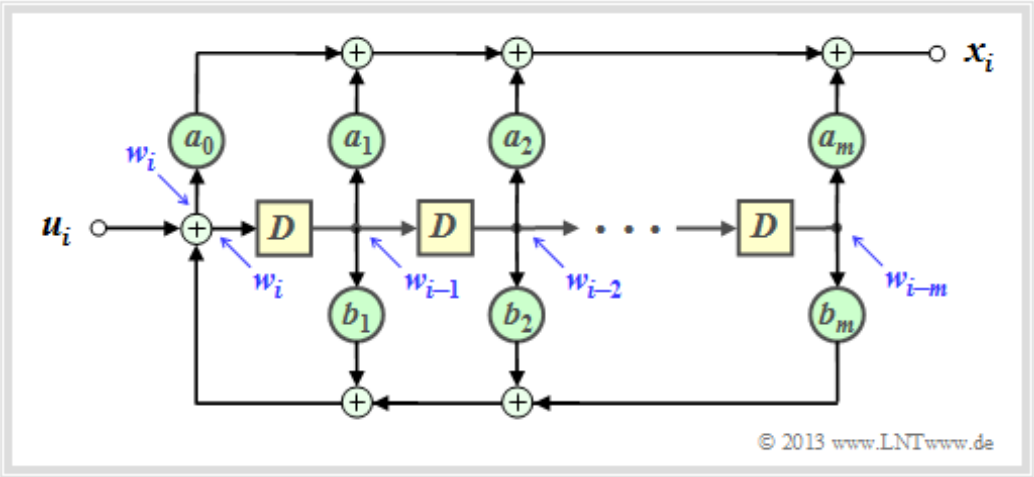
Filterstruktur bei gebrochen–rationaler Übertragungsfunktion

Hat eine Übertragungsfunktion die Form $G(D) = A(D)/B(D)$, so bezeichnet man das zugehörige Filter als *rekursiv*. Bei einem rekursiven Faltungscodierer mit dem Gedächtnis m kann für die beiden Polynome $A(D)$ und $B(D)$ allgemein geschrieben werden:

$$A(D) = \sum_{l=0}^{m} a_l \cdot D^l = a_0 + a_1 \cdot D + a_2 \cdot D^2 + \ldots + a_m \cdot D^m ,$$

$$B(D) = 1 + \sum_{l=1}^{m} b_l \cdot D^l = 1 + b_1 \cdot D + b_2 \cdot D^2 + \ldots + b_m \cdot D^m .$$

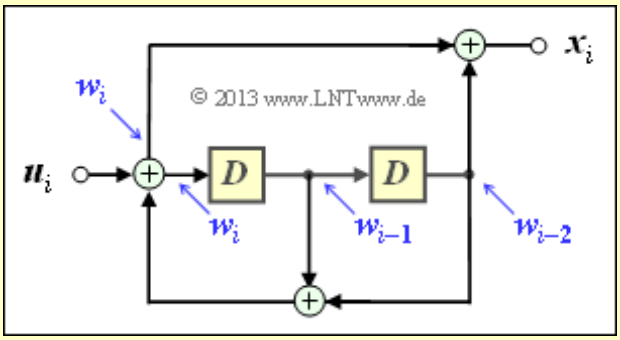
Die Grafik zeigt die entsprechende Filterstruktur in der so genannten *Controller Canonical Form*.



Die Koeffizienten a_0, ... , a_m beschreiben den Vorwärtszweig, während b_1, ... , b_m eine Rückkopplung bilden. Alle Koeffizienten sind binär, also 1 (durchgehende Verbindung) oder 0 (fehlende Verbindung).

Beispiel: Die rechts skizzierte Filterstruktur lässt sich durch folgende Gleichungen beschreiben:

$$x_i = w_i + w_{i-2} ,$$
$$w_i = u_i + w_{i-1} + w_{i-2} .$$



Entsprechend gilt für die D–Transformierten:

$$X(D) = W(D) + W(D) \cdot D^2 = W(D) \cdot \left(1 + D^2\right) ,$$

$$W(D) = U(D) + W(D) \cdot D + W(D) \cdot D^2 \quad \Rightarrow \quad U(D) = W(D) \cdot \left(1 + D + D^2\right) .$$

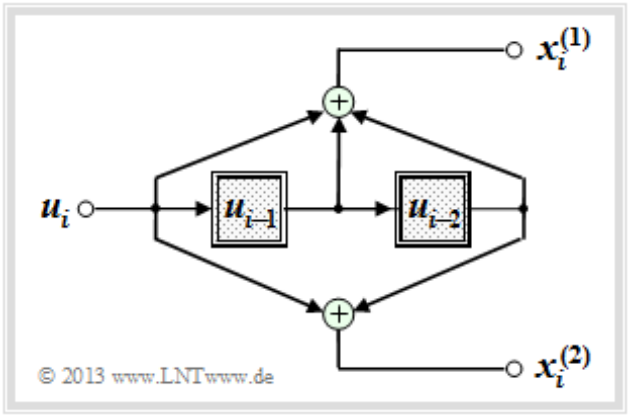
Somit erhält man für die Übertragungsfunktion dieses Filters:

$$G(D) = \frac{X(D)}{U(D)} = \frac{1 + D^2}{1 + D + D^2} .$$

Im **Beispiel zu den systematischen Faltungscodes** hat sich genau ein solcher Ausdruck ergeben.

Zustandsdefinition für ein Speicherregister (1)

Ein Faltungscodierer kann auch als Automat mit endlicher Anzahl von Zuständen aufgefasst werden. Die Zustandsanzahl ergibt sich dabei aus der Zahl der Speicherelemente $\Rightarrow$ Gedächtnis m zu 2^m.



Im Kapitel 3.3 gehen wir meist vom gezeichneten Faltungscodierer aus, der durch folgende Kenngrößen charakterisiert wird:

- $k = 1$, $n = 2$ $\Rightarrow$ Coderate $R = 1/2$,
- Gedächtnis $m = 2$ $\Rightarrow$ Einflusslänge $\nu = 3$,
- Übertragungsfunktionsmatrix in Oktalform (7, 5) $\Rightarrow$ $\mathbf{G}(D) = (1 + D + D^2,\ 1 + D^2)$.

Die Codesequenz zum Zeitpunkt i $\Rightarrow$ $\underline{x}_i = (x_i^{(1)}, x_i^{(2)})$ hängt außer vom Informationsbit u_i auch vom Inhalt (u_{i-1}, u_{i-2}) des Speichers ab. Hierfür gibt es $2^m = 4$ Möglichkeiten, die man als die Zustände S_0, S_1, S_2 und S_3 bezeichnet. Der Registerzustand S_μ sei dabei über den Index definiert:

$$\mu = u_{i-1} + 2 \cdot u_{i-2}\,, \quad \text{allgemein: } \mu = \sum_{l=1}^{m} 2^{l-1} \cdot u_{i-l}\,.$$

Im Englischen verwendet man für „Zustand” den Begriff *State*. Entsprechend ist auch im deutschen Text manchmal vom *Registerstatus* die Rede.

Um Verwechslungen zu vermeiden, unterscheiden wir im Weiteren durch Groß– bzw. Kleinbuchstaben:

- die möglichen Zustände S_μ mit den Indizes $0 \le \mu \le 2^m - 1$,
- die aktuellen Zustände s_i zu den Zeitpunkten $i = 1, 2, 3, ...$

Auf der nächsten Seite verdeutlichen wir die Zustände an einem Beispiel.

Zustandsdefinition für ein Speicherregister (2)

Beispiel: Die folgende Grafik zeigt für obigen Faltungscodierer jeweils den Beginn ($i \le 15$)

- der Informationssequenz $\underline{u} \Rightarrow$ Informationsbits u_i,
- der aktuellen Zustände $s_i \in \{S_0, S_1, S_2, S_3\}$ zu den Zeitpunkten i, sowie
- der jeweiligen Codesequenzen $\underline{x}_i = (x_i^{(1)}, x_i^{(2)})$.

Zeitpunkt i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	...
Informationsbit u_i	1	1	1	0	0	0	1	0	1	1	0	0	1	0	1	...
Zustand $s_i = S_\mu$	S_0	S_1	S_3	S_3	S_2	S_0	S_0	S_1	S_2	S_1	S_3	S_2	S_0	S_1	S_2	...
Codesequenz $\underline{x}_i$	11	01	10	01	11	00	11	10	00	01	01	11	11	10	00	...

Die Farbkennzeichnungen sollen den Übergang zu den nachfolgenden Grafiken auf den nächsten Seiten erleichtern. Man erkennt aus obiger Darstellung beispielsweise:

- Zum Zeitpunkt $i = 5$ gilt $u_{i-1} = u_4 = 0$, $u_{i-2} = u_3 = 1$. Das heißt, der Automat befindet sich im Zustand $s_5 = S_2$. Mit dem Informationsbit $u_i = u_5 = 0$ erhält man die Codesequenz $\underline{x}_5 = (11)$.
- Der Zustand für den Zeitpunkt $i = 6$ ergibt sich aus $u_{i-1} = u_5 = 0$ und $u_{i-2} = u_4 = 0$ zu $s_6 = S_0$. Wegen $u_6 = 0$ wird nun $\underline{x}_6 = (00)$ ausgegeben und der neue Zustand s_7 ist wiederum S_0.
- Auch zum Zeitpunkt $i = 12$ wird wegen $u_{12} = 0$ die Codesequenz $\underline{x}_{12} = (11)$ ausgegeben und man geht vom Zustand $s_{12} = S_2$ in den Zustand $s_{13} = S_0$ über.
- Dagegen wird zum Zeitpunkt $i = 9$ die Codesequenz (00) ausgegeben und auf $s_9 = S_2$ folgt $s_{10} = S_1$. Gleiches gilt auch für $i = 15$. In beiden Fällen lautet das aktuelle Informationsbit $u_i = 1$.

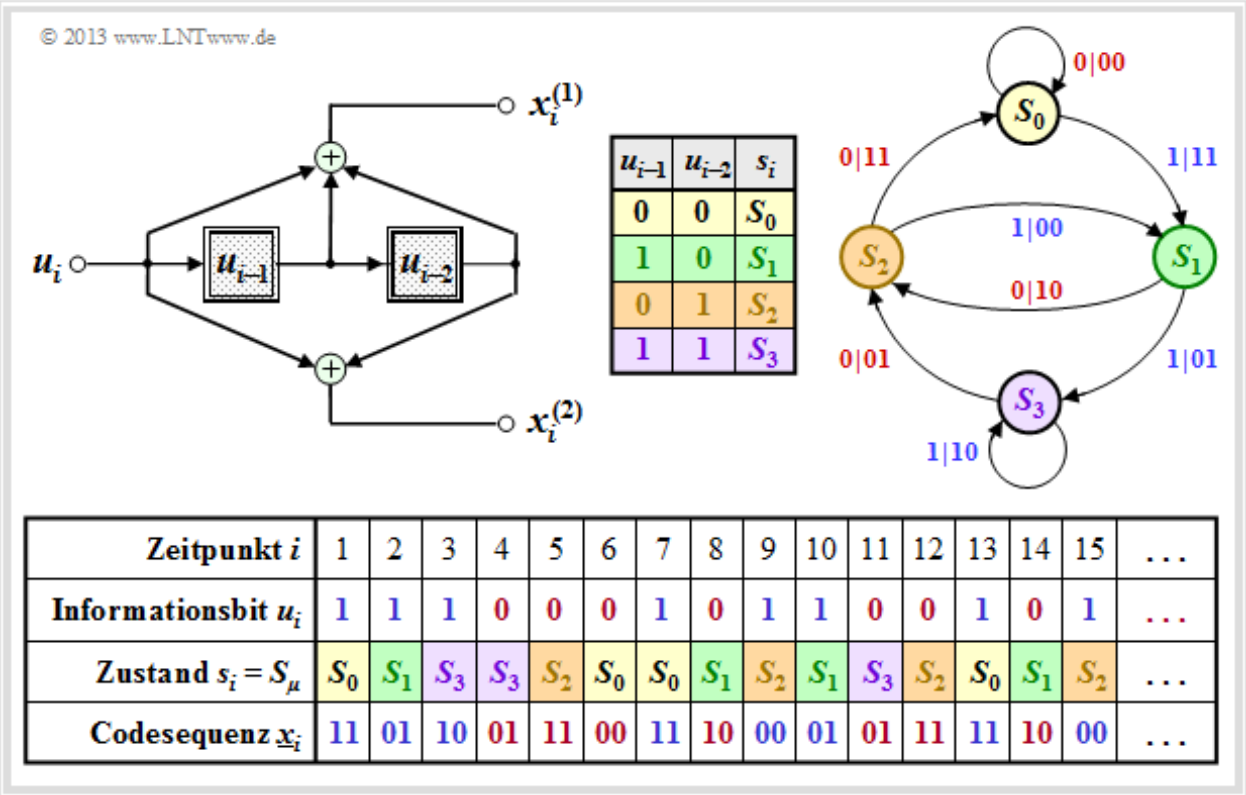
Aus diesem Beispiel ist zu erkennen, dass die Codesequenz $\underline{x}_i$ zum Zeitpunkt i allein

- vom aktuellen Zustand $s_i = S_\mu$ ($0 \le \mu \le 2^m - 1$), sowie
- vom aktuellen Informationsbit u_i

abhängt. Ebenso wird der Nachfolgezustand s_{i+1} allein durch s_i und u_i bestimmt. Diese Eigenschaften werden im so genannten *Zustandsübergangsdiagramm* auf der nächsten Seite berücksichtigt.

Darstellung im Zustandsübergangsdiagramm (1)

Die Grafik zeigt das **Zustandsübergangsdiagramm** (englisch: *State Transition Diagram*) für unseren Standardcodierer. Dieses liefert alle Informationen über den ($n = 2$, $k = 1$, $m = 2$)–Faltungscodierer in kompakter Form. Die Farbgebung ist mit der **sequenziellen Darstellung** auf der vorherigen Seite abgestimmt. Der Vollständigkeit halber ist auch die Herleitungstabelle nochmals angegeben.



Zeitpunkt i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	...
Informationsbit u_i	1	1	1	0	0	0	1	0	1	1	0	0	1	0	1	...
Zustand $s_i = S_\mu$	S_0	S_1	S_3	S_3	S_2	S_0	S_0	S_1	S_2	S_1	S_3	S_2	S_0	S_1	S_2	...
Codesequenz $\underline{x}_i$	11	01	10	01	11	00	11	10	00	01	01	11	11	10	00	...

Die 2^{m+k} Übergänge sind mit „$u_i \mid \underline{x}_i$" beschriftet. Beispielsweise ist abzulesen:

- Durch das Informationsbit $u_i = 0$ (gekennzeichnet durch eine rote Beschriftung) gelangt man vom Zustand $s_i = S_1$ zum Zustand $s_{i+1} = S_2$ und die beiden Codebits lauten $x_i^{(1)} = 1$, $x_i^{(2)} = 0$.
- Mit dem Informationsbit $u_i = 1$ (blaue Beschriftung) im Zustand $s_i = S_1$ ergeben sich dagegen die Codebits zu $x_i^{(1)} = 0$, $x_i^{(2)} = 1$, und man kommt zum neuen Zustand $s_{i+1} = S_3$.

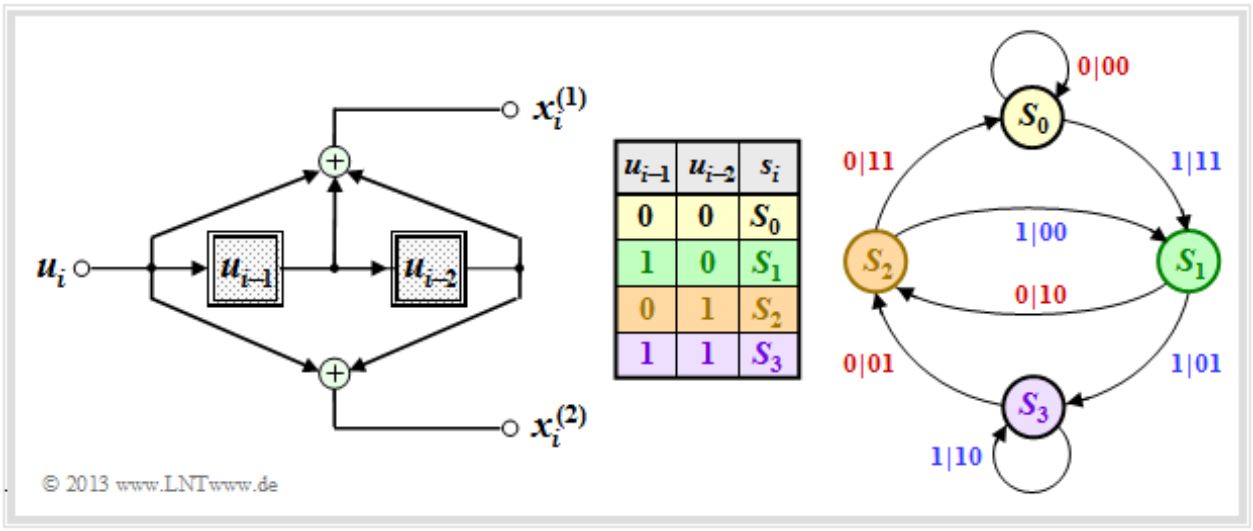
Die Struktur des Zustandsübergangsdiagramms ist allein durch die Parameter m und k festgelegt:

- Die Anzahl der Zustände ist $2^{m \cdot k}$.
- Von jedem Zustand gehen 2^k Pfeile ab.

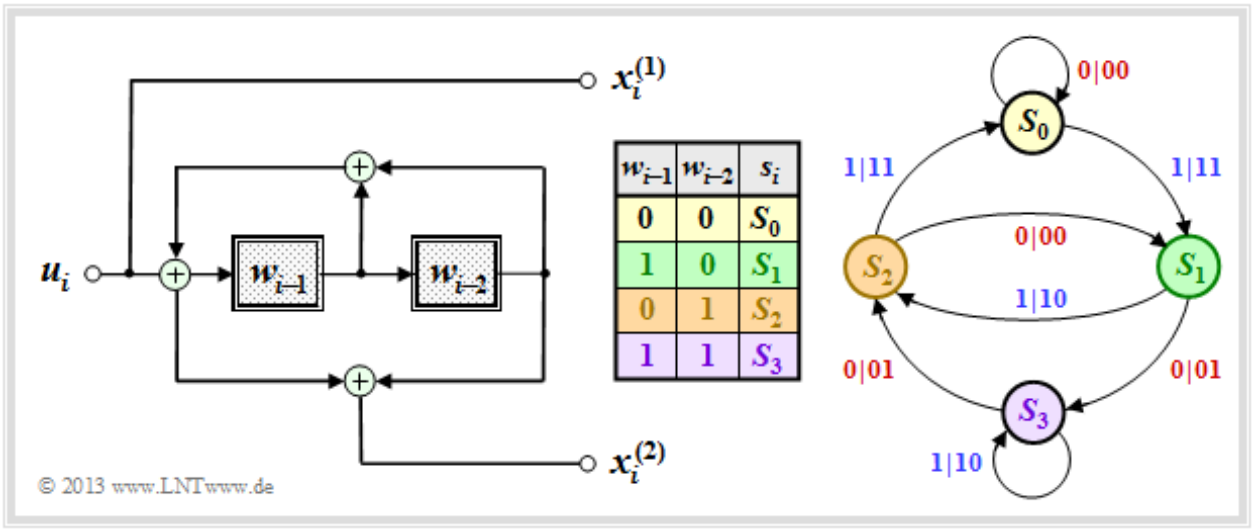
Ein weiteres Beispiel folgt auf der nächsten Seite.

Darstellung im Zustandsübergangsdiagramm (2)

Die obere Grafik zeigt nochmals das Zustandsübergangsdiagramm für unseren Standardcodierer. Dieses dient lediglich als Vergleichsgrundlage für das nachfolgende Beispiel.



Die untere Grafik gilt für einen systematischen Code, ebenfalls mit den Kenngrößen $n = 2$, $k = 1$, $m = 2$. Es handelt sich um die **äquivalente systematische Repräsentation** des obigen Codierers. Man bezeichnet diesen auch als RSC–Codierer (englisch: *Recursive Systematic Convolutional Encoder*).



Gegenüber dem oberen Zustandsübergangsdiagramm erkennt man folgende Unterschiede:

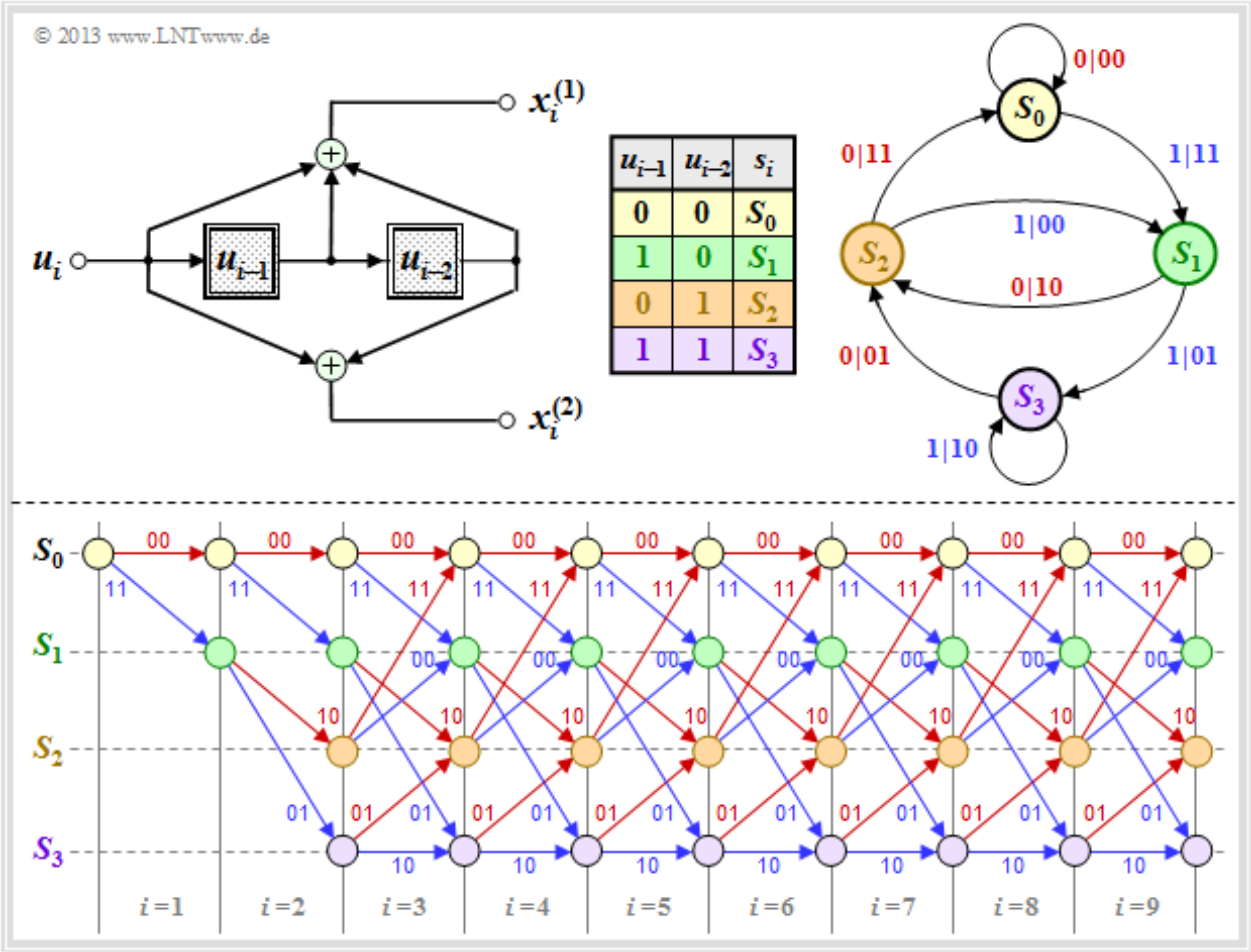
- Da die früheren Informationsbits $u_{i–1}$ und $u_{i–2}$ nicht abgespeichert werden, beziehen sich hier die Zustände $s_i = S_\mu$ auf die verarbeiteten Größen $w_{i–1}$ und $w_{i–2}$, wobei $w_i = u_i + w_{i–1} + w_{i–2}$ gilt.
- Da dieser Code systematisch ist, gilt $x_i^{(1)} = u_i$. Die Herleitung der zweiten Codebits $x_i^{(2)}$ finden Sie in **Aufgabe A3.5.** Es handelt sich um ein rekursives Filter, wie in **Kapitel 3.2** beschrieben.

Der Bildervergleich zeigt, dass sich für beide Codierer ein ähnliches Zustandsübergangsdiagramm ergibt:

- Man gelangt von jedem Zustand $s_i \in \{S_0, S_1, S_2, S_3\}$ zu den gleichen Nachfolgezuständen s_{i+1}.
- Ein Unterschied besteht hinsichtlich der ausgegebenen Codesequenzen $\underline{x}_i \in \{00, 01, 10, 11\}$.

Darstellung im Trellisdiagramm (1)

Man kommt vom Zustandsübergangsdiagramm zum so genannten *Trellisdiagramm* (oder kurz: Trellis), indem man zusätzlich eine Zeitkomponente ⇒ Laufvariable i berücksichtigt. Die folgende Grafik stellt die beiden Diagramme für unseren Standardcodierer ($n = 2$, $k = 1$, $m = 2$) gegenüber.



Die untere Darstellung hat Ähnlichkeit mit einem Gartenspalier – etwas Phantasie vorausgesetzt. Die englische Übersetzung hierfür ist „Trellis". Weiter ist anhand dieser Grafik zu erkennen:

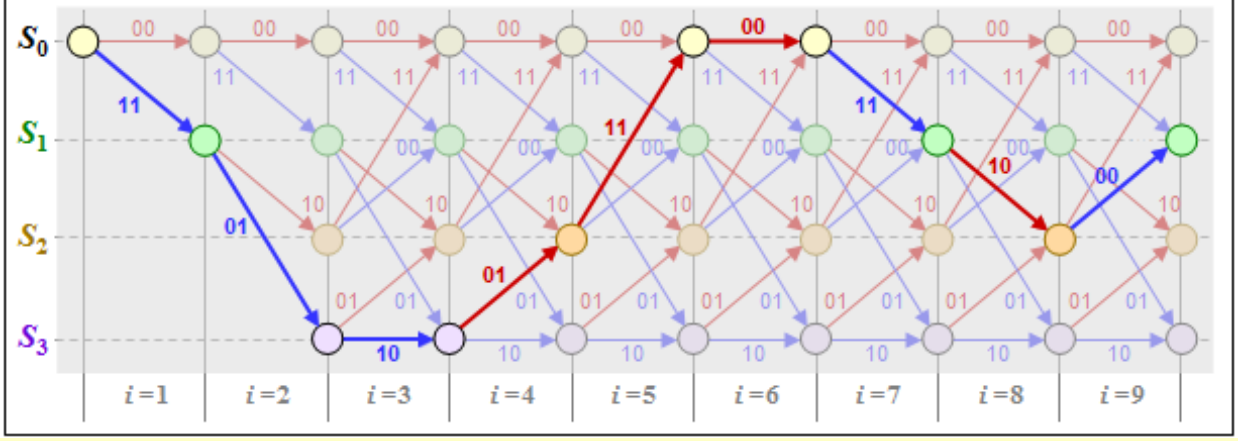
- Da alle Speicherregister mit Nullen vorbelegt sind, startet das Trellis stets vom Zustand $s_1 = S_0$. Zum nächsten Zeitpunkt ($i = 2$) sind dann nur die beiden Zustände S_0 und S_1 möglich.
- Ab dem Zeitpunkt $i = m + 1 = 3$ hat das Trellis für jeden Übergang von s_i nach s_{i+1} genau das gleiche Aussehen. Jeder Zustand S_μ ist durch einen roten Pfeil ($u_i = 0$) und einen blauen Pfeil ($u_i = 1$) mit einem Nachfolgezustand verbunden.
- Gegenüber einem *Codebaum*, der mit jedem Zeitschritt i exponentiell anwächst – siehe zum Beispiel **Kapitel 3.8, Seite 2a** im Buch „Digitalsignalübertragung" – ist hier die Zahl der Knoten (also der möglichen Zustände) auf 2^m begrenzt.
- Diese erfreuliche Eigenschaft eines jeden Trellisdiagramms nutzt auch der **Viterbi–Algorithmus** zur effizienten Maximum–Likelihood–Decodierung von Faltungscodes.

Darstellung im Trellisdiagramm (2)

Das folgende Beispiel soll zeigen, dass zwischen der Aneinanderreihung der Codesequenzen $\underline{x}_i$ und den Pfaden durch das Trellisdiagramm eine 1:1–Zuordnung besteht. Auch die Informationssequenz $\underline{u}$ ist aus dem ausgewählten Trellispfad anhand der Farben der einzelnen Zweige ablesbar. Ein roter Zweig steht für das Informationsbit $u_i = 0$, ein blauer für $u_i = 1$.

Beispiel: Auf der ersten Seite dieses Abschnitts wurde für unseren Rate–1/2–Standardcodierer mit Gedächtnis $m = 2$ sowie die Informationssequenz $\underline{u} = (1, 1, 1, 0, 0, 0, 1, 0, 1, ...)$ die Codesequenz $\underline{x}$ hergeleitet, die in nachfolgender Tabelle für $i \le 9$ nochmals angegeben ist.

Zeitpunkt i	1	2	3	4	5	6	7	8	9	...
Informationsbit u_i	1	1	1	0	0	0	1	0	1	...
Zustand $s_i = S_\mu$	S_0	S_1	S_3	S_3	S_2	S_0	S_0	S_1	S_2	...
Codesequenz $\underline{x}_i$	11	01	10	01	11	00	11	10	00	...



Darunter gezeichnet ist das Trellisdiagramm. Man erkennt:

- Der ausgewählte Pfad durch das Trellis ergibt sich durch die Aneinanderreihung roter und blauer Pfeile, die für die möglichen Informationsbits $u_i = 0$ bzw. $u_i = 1$ stehen. Diese Aussage gilt für jeden Rate–1/n–Faltungscode. Bei einem Code mit $k > 1$ gäbe es 2^k verschiedenfarbige Pfeile.
- Bei einem Rate–1/n–Faltungscode sind die Pfeile mit den Codeworten $\underline{x}_i = (x_i^{(1)}, \ ... \ , (x_i^{(n)})$ beschriftet, die sich aus dem Informationsbit u_i und den aktuellen Registerzuständen s_i ergeben. Für $n = 2$ gibt es nur vier mögliche Codeworte, nämlich 00, 01, 10 und 11.
- In vertikaler Richtung erkennt man aus dem Trellis die möglichen Registerzustände S_μ. Bei einem Rate–k/n–Faltungscode mit der Gedächtnisordnung m gibt es $2^{k \cdot m}$ verschiedene Zustände. Im vorliegenden Beispiel ($k = 1$, $m = 2$) sind dies nur die Zustände S_0, S_1, S_2 und S_3.

Definition der freien Distanz (1)

Als eine wichtige Kenngröße der linearen Blockcodes wurde in **Kapitel 1.1** die **minimale Hamming–Distanz** zwischen zwei beliebigen Codeworten $\underline{x}$ und $\underline{x}'$ eingeführt:

$$d_{\rm min}(\mathcal{C}) = \min_{\substack{\underline{x},\underline{x}' \in \mathcal{C} \\ \underline{x} \neq \underline{x}'}} d_{\rm H}(\underline{x}, \underline{x}') .$$

Aufgrund der Linearität gehört zu jedem Blockcode auch das Nullwort $\underline{0}$. Damit ist $d_{\rm min}$ auch gleich dem minimalen **Hamming–Gewicht** $w_{\rm H}\,(\underline{x})$ eines Codewortes $\underline{x} \neq \underline{0}$.

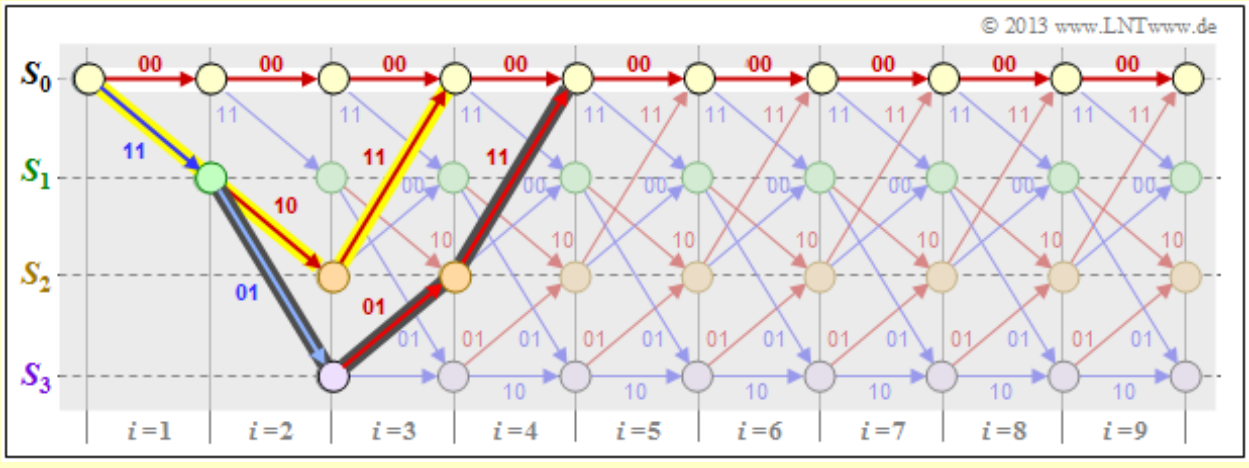
Bei Faltungscodes erweist sich die Beschreibung der Distanzverhältnisse als wesentlich aufwändiger, da ein Faltungscode aus unendlich langen und unendlich vielen Codesequenzen besteht.

Definition: Die **freie Distanz** $d_{\rm F}$ eines Faltungscodes ist gleich der Anzahl der Bits, in dem sich zwei beliebige Codesequenzen $\underline{x}$ und $\underline{x}'$ (mindestens) unterscheiden. Anders ausgedrückt: Die freie Distanz ist gleich der **minimalen** Hamming–Distanz zwischen zwei beliebigen Pfaden durch das Trellis.

Da Faltungscodes ebenfalls linear sind, kann man auch hier als Bezugssequenz die unendlich lange Nullsequenz heranziehen: $\underline{x} = \underline{0}$. Damit ist die freie Distanz $d_{\rm F}$ gleich dem minimalen Hamming–Gewicht (Anzahl der Einsen) einer Codesequenz $\underline{x} \neq \underline{0}$.

Beispiel: Wir betrachten die Nullsequenz $\underline{0}$ (weiß markiert) sowie zwei andere Codesequenzen $\underline{x}$ sowie $\underline{x}'$ (mit gelber bzw. dunkler Hinterlegung) in unserem Standard–Trellis und charakterisieren diese Sequenzen anhand der Zustandsfolgen:

$$\underline{0} = (S_0 \to S_0 \to S_0 \to S_0 \to S_0 \to \ \ldots) = (00, 00, 00, 00, 00, \ \ldots) ,$$
$$\underline{x} = (S_0 \to S_1 \to S_2 \to S_0 \to S_0 \to \ \ldots) = (11, 10, 11, 00, 00, \ \ldots) ,$$
$$\underline{x}' = (S_0 \to S_1 \to S_3 \to S_2 \to S_0 \to \ \ldots) = (11, 01, 01, 11, 00, \ \ldots) .$$



Man erkennt aus diesen Darstellungen:

- Die freie Distanz $d_{\rm F} = 5$ ist gleich dem Hamming–Gewicht $w_{\rm H}(\underline{x})$. Keine andere Sequenz als die gelb hinterlegte unterscheidet sich von $\underline{0}$ um weniger als 5 Bit. Beispielsweise ist $w_{\rm H}(\underline{x}') = 6$.
- In diesem Beispiel ergibt sich die freie Distanz $d_{\rm F} = 5$ auch als die Hamming–Distanz zwischen den Sequenzen $\underline{x}$ und $\underline{x}'$, die sich genau in fünf Bitpositionen unterscheiden.

Definition der freien Distanz (2)

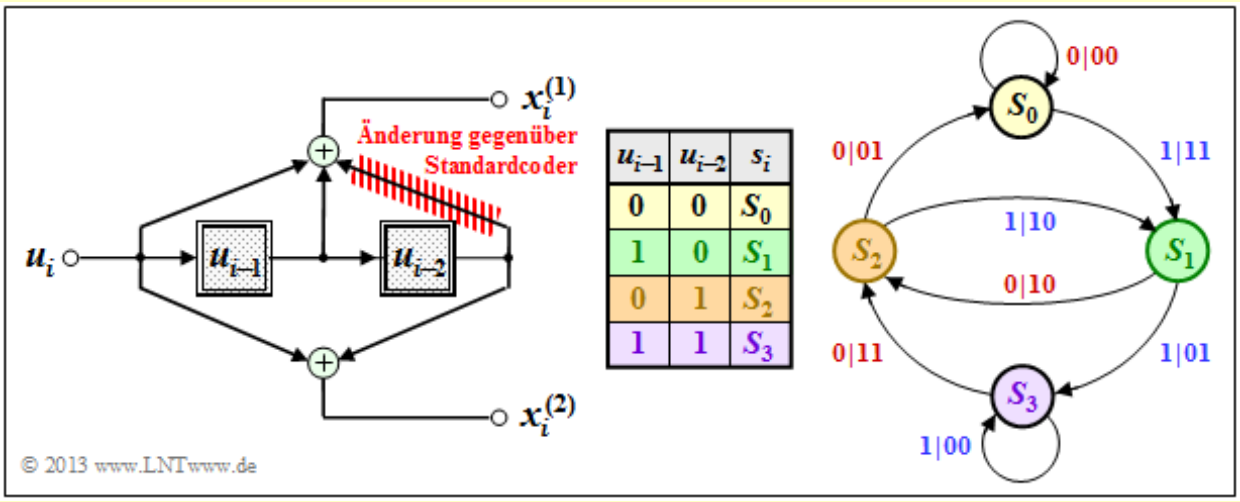
Je größer die freie Distanz d_F ist, desto besser ist das asymptotische Verhalten des Faltungscoders. Zur genauen Berechnung von Fehlerwahrscheinlichkeit benötigt man allerdings – ebenso wie bei den linearen Blockcodes – die **Gewichtsfunktion** (englisch: *Weight Enumerator Function*) ⇒ siehe **Kapitel 3.5.**

Die freie Distanz d_F nimmt mit wachsendem Gedächtnis m zu, vorausgesetzt, man verwendet für die Übertragungsfunktionsmatrix $\mathbf{G}(D)$ geeignete Polynome. In der Tabelle sind für Rate–1/2–Faltungscodes die $n = 2$ Polynome zusammen mit dem d_F–Wert angegeben. Von Bedeutung ist insbesondere der sog. *Industriestandardcode* mit $m = 6$ ⇒ 64 Zustände und der freien Distanz $d_F = 10$.

Gedächtnis	oberes Filter $G^{(1)}(D)$	unteres Filter $G^{(2)}(D)$	freie Distanz
$m = 1$	1	$1+D$	$d_F = 3$
$m = 2$	$1+D+D^2$	$1+D^2$	$d_F = 5$
$m = 3$	$1+D+D^3$	$1+D+D^2+D^3$	$d_F = 6$
$m = 4$	$1+D^3+D^4$	$1+D+D^2+D^4$	$d_F = 7$
$m = 5$	$1+D^2+D^4+D^5$	$1+D+D^2+D^3+D^5$	$d_F = 8$
$m = 6$	$1+D^2+D^3+D^5+D^6$	$1+D+D^2+D^3+D^6$	$d_F = 10$

Das folgende Beispiel zeigt, welche Auswirkungen es hat, wenn man ungünstige Polynome zugrundelegt.

Beispiel: Unser ($n = 2$, $k = 1$, $m = 2$)–Standard–Coder basiert auf der Übertragungsfunktionsmatrix $\mathbf{G}(D) = (1+D+D^2, 1+D^2)$. Dieser weist die freie Distanz $d_F = 5$ auf.

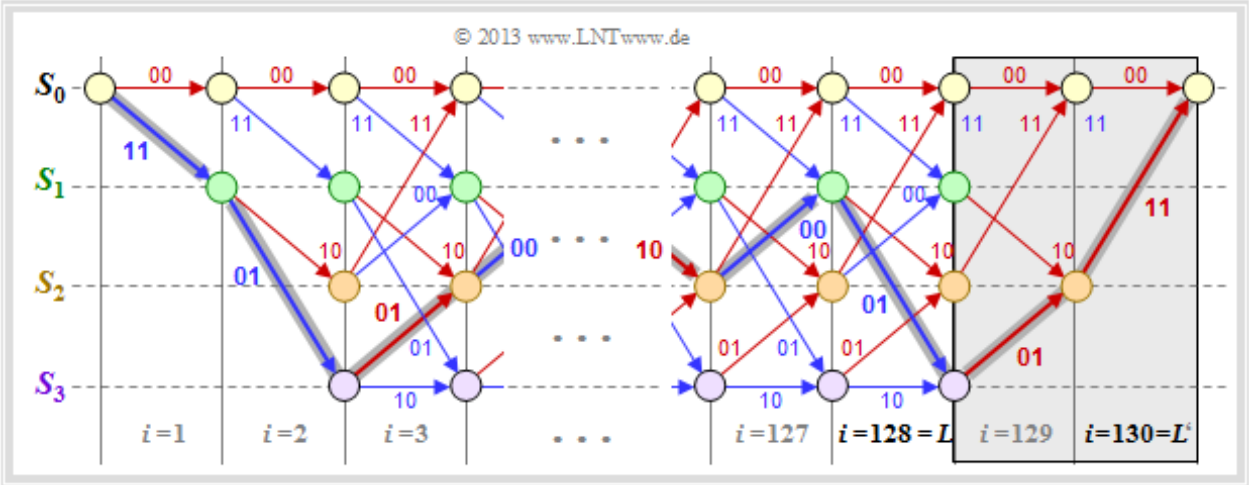


Verwendet man $\mathbf{G}(D) = (1+D, 1+D^2)$, so ändert sich das Zustandsübergangsdiagramm gegenüber dem Standard–Coder ⇒ **Seite 2a** nur wenig. Die Auswirkungen sind aber gravierend:

- Die freie Distanz ist nun nicht mehr $d_F = 5$, sondern nur noch $d_F = 3$, wobei die Codesequenz $\underline{x} = (11, 01, 00, 00, ...)$ zur Informationssequenz $\underline{u} = \underline{1} = (1, 1, 1, 1, ...)$ gehört.
- Das heißt: Durch nur drei Übertragungsfehler an den Positionen 1, 2, und 4 verfälscht man die Einssequenz ($\underline{1}$) in die Nullsequenz ($\underline{0}$) und produziert so unendlich viele Decodierfehler.
- Einen Faltungscodierer mit diesen Eigenschaften bezeichnet man als *katastrophal*. Der Grund für dieses extrem ungünstige Verhalten ist, dass hier $1+D$ und $1+D^2$ nicht teilerfemd sind.

Terminierte Faltungscodes

Bei der theoretischen Beschreibung der Faltungscodes geht man stets von Informationssequenzen $\underline{u}$ und Codesequenzen $\underline{x}$ aus, die per Definition unendlich lang sind. In praktischen Anwendungen, siehe zum Beispiel **GSM** und **UMTS,** verwendet man dagegen stets eine Informationssequenz endlicher Länge L. Bei einem Rate–1/n–Faltungscode hat dann die Codesequenz mindestens die Länge $n \cdot L$.



Die Grafik zeigt ohne Hinterlegung das Trellis unseres Standard–Rate–1/2–Faltungscodes bei binärer Eingangsfolge $\underline{u}$ der endlichen Länge $L = 128$. Damit hat die Codefolge $\underline{x}$ die Länge $2 \cdot L = 256$. Aufgrund des undefinierten Endzustands ist eine vollständige Maximum–Likelihood–Decodierung der gesendeten Folge allerdings nicht möglich. Da man nicht weiß, welcher der Zustände S_0, ... , S_3 sich für $i > L$ einstellen würden, wird die Fehlerwahrscheinlichkeit (etwas) größer sein als im Grenzfall $L \to \infty$.

Um dies zu verhindern, terminiert man den Faltungscode entsprechend der Hinterlegung in obiger Grafik:

- Man fügt an die $L = 128$ Informationsbits noch $m = 2$ Nullen an $\Rightarrow$ $L' = 130$.
- Damit ergibt sich beispielsweise für den farblich hervorgehobenen Pfad durch das Trellis:

$$\underline{x}' = (11, 01, 01, 00, \ldots, 10, 00, 01, 01, 11)$$
$$\Rightarrow \quad \underline{u}' = (1, 1, 0, 1, \ldots, 0, 1, 1, 0, 0).$$

- Das Trellis endet nun stets (also unabhängig von den Daten) im definierten Endzustand S_0 und man erreicht so die bestmögliche Fehlerwahrscheinlichkeit entsprechend Maximum–Likelihood.
- Die Verbesserung hinsichtlich der Fehlerwahrscheinlichkeit erkauft man sich allerdings auf Kosten einer kleineren Coderate. Gilt $L >> m$, so ist dieser Verlust nur gering. Im betrachteten Beispiel ergibt sich mit Terminierung die Coderate $R' = 0.5 \cdot L/(L + m) \approx 0.492$ anstelle von $R = 0.5$.

Punktierte Faltungscodes

Wir gehen von einem Faltungscode der Rate $R_0 = 1/n_0$ aus, den wir *Muttercode* nennen. Streicht man von dessen Codesequenz einige Bits entsprechend einem vorgegebenen Muster, so spricht man von einem *punktierten Faltungscode* (englisch: *Punctured Convolutional Code*) mit der Coderate $R > R_0$.

Die Punktierung geschieht mittels der Punktierungsmatrix **P** mit folgenden Eigenschaften:

- Die Zeilenzahl ist n_0, die Spaltenzahl gleich der Punktierungsperiode p, die durch die gewünschte Coderate bestimmt wird.
- Die $n_0 \cdot p$ Matrixelemente P_{ij} sind binär (0 oder 1). Bei $P_{ij} = 1$ wird das entsprechende Codebit übernommen, bei $P_{ij} = 0$ punktiert.
- Die Rate des punktierten Faltungscodes ergibt sich als der Quotient aus p und der Anzahl N_1 der Einsen in der **P**–Matrix.

Man findet günstig punktierte Faltungscodes üblicherweise nur mittels computergestützter Suche. Dabei bezeichnet man einen punktierten Faltungscode dann als günstig, wenn er

- die gleiche Gedächtnisordnung m aufweist wie der Muttercode (auch die Gesamteinflusslänge ist in beiden Fällen gleich: $\nu = m + 1$),
- eine möglichst große freie Distanz d_F besitzt, die natürlich kleiner ist als die des Muttercodes.

Beispiel: Ausgehend von unserem **Rate–1/2–Standardcode** mit den Parametern $n_0 = 2, m = 2$ erhält man mit der Punktierungsmatrix

$$\mathbf{P} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \quad \Rightarrow \quad p = 3, \quad N_1 = 4$$

einen punktierten Faltungscode der Rate $R = p/N_1 = 3/4$. Wir betrachten hierzu folgende Konstellation:

- Informationssequenz: $\underline{u} = (1, 0, 0, 1, 1, 0, ...)$,
- Codesequenz ohne Punktierung: $\underline{x} = (11, 10, 11, 11, 01, 01, ...)$,
- Codesequenz mit Punktierung: $\underline{x}' = (11, 1\_, \_1, 11, 0\_, \_1, ...)$,
- Empfangssequenz ohne Fehler: $\underline{y} = (11, 1\_, \_1, 11, 0\_, \_1, ...)$,
- Modifizierte Empfangssequenz: $\underline{y}' = (11, 1E, E1, 11, 0E, E1, ...)$.

Jedes punktierte Bit in der Empfangssequenz $\underline{y}$ (markiert durch einen Unterstrich) wird also durch ein *Erasure* E ersetzt – siehe **Binary Erasure Channel.** Ein solches durch die Punktierung entstandene *Erasure* wird vom Decoder genauso behandelt wie eine Auslöschung durch den Kanal.

Natürlich erhöht sich durch die Punktierung die Fehlerwahrscheinlichkeit. Dies kann man bereits daran erkennen, dass die freie Distanz nach der Punktierung auf $d_F = 3$ sinkt. Demgegenüber besitzt der Muttercode die freie Distanz $d_F = 5$.

Eine Anwendung findet die Punktierung zum Beispiel bei den so genannten RCPC–Codes (*Rate Compatible Punctered Convolutional Codes*). Näheres hierzu in der **Aufgabe A3.8.**

Blockschaltbild und Voraussetzungen

Ein wesentlicher Vorteil der Faltungscodierung ist, dass es hierfür mit dem Viterbi–Algorithmus ein sehr effizientes Decodierverfahren gibt. Dieser von **Andrew J. Viterbi** entwickelte Algorithmus wurde bereits im **Kapitel 3.8** des Buches „Digitalsignalübertragung" im Hinblick auf seinen Einsatz zur Entzerrung im Detail beschrieben. Für seinen Einsatz als Faltungsdecodierer gehen wir von folgendem Blockschaltbild und den folgenden Voraussetzungen aus:

- Die Informationssequenz $\underline{u} = (u_1, u_2, ...)$ ist hier im Gegensatz zur Beschreibung der linearen Blockcodes ⇒ **Kapitel 1.5** oder von Reed–Solomon–Codes ⇒ **Kapitel 2.4** im allgemeinen unendlich lang (*„semi–infinite"*). Für die Informationssymbole gilt stets $u_i \in \{0, 1\}$.

- Die Codesequenz $\underline{x} = (x_1, x_2, ...)$ mit $x_i \in \{0, 1\}$ hängt außer von $\underline{u}$ auch noch von der Coderate $R = 1/n$, dem Gedächtnis m und der Übertragungsfunktionsmatrix $\mathbf{G}(D)$ ab. Bei endlicher Anzahl L an Informationsbits sollte der Faltungscode durch Anfügen von m Nullen terminiert werden:

$$\underline{u} = (u_1, u_2, \ldots, u_L, 0, \ldots, 0) \quad \Rightarrow \quad \underline{x} = (x_1, x_2, \ldots, x_{2L}, x_{2L+1}, \ldots, x_{2L+2m}).$$

- Die Empfangssequenz $\underline{y} = (y_1, y_2, ...)$ ergibt sich entsprechend dem angenommenen Kanalmodell. Bei einem digitalen Modell wie dem **Binary Symmetric Channel** (BSC) gilt $y_i \in \{0, 1\}$, so dass die Verfälschung von $\underline{x}$ auf $\underline{y}$ mit der **Hamming–Distanz** $d_H(\underline{x}, \underline{y})$ quantifiziert werden kann. Die erforderlichen Modifikationen für den **AWGN–Kanal** folgen auf **Seite 6** dieses Kapitels.

- Der nachfolgend beschriebene Viterbi–Algorithmus liefert eine Schätzung $\underline{z}$ für die Codesequenz $\underline{x}$ und eine weitere Schätzung $\underline{v}$ für die Informationssequenz $\underline{u}$. Dabei gilt:

$$\Pr(\underline{z} \neq \underline{x}) \stackrel{!}{=} \text{Minimum} \quad \Rightarrow \quad \Pr(\underline{v} \neq \underline{u}) \stackrel{!}{=} \text{Minimum}.$$

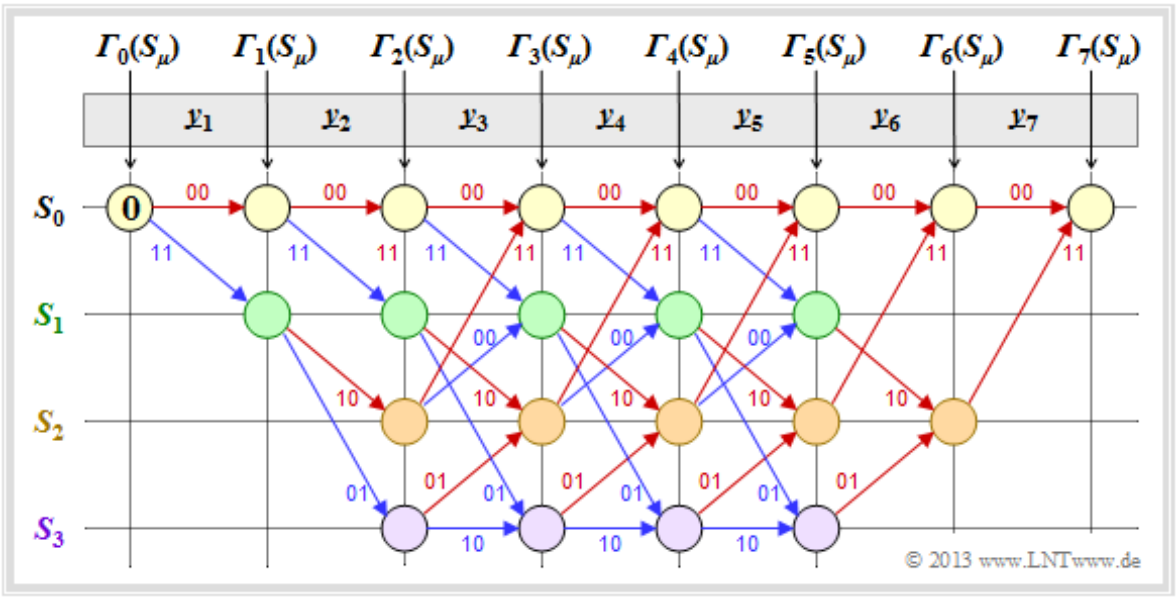
Zusammenfassung: Der Viterbi–Algorithmus sucht bei einem digitalen Kanalmodell (zum Beispiel BSC) von allen möglichen Codesequenzen $\underline{x}'$ diejenige Sequenz $\underline{z}$ mit der minimalen Hamming–Distanz $d_H(\underline{x}', \underline{y})$ zur Empfangssequenz $\underline{y}$:

$$\underline{z} = \arg \min_{\underline{x}' \in \mathcal{C}} d_H(\underline{x}', \underline{y}) = \arg \max_{\underline{x}' \in \mathcal{C}} \Pr(\underline{y} \mid \underline{x}').$$

Das bedeutet auch: Der Viterbi–Algorithmus erfüllt das **Maximum–Likelihood–Kriterium.**

Vorbemerkungen zu den nachfolgenden Decodierbeispielen (1)

In den Beispielen dieses Kapitels wird stets von unserem Standard–Faltungscodierer der Rate $R = 1/2$, mit dem Gedächtnis $m = 2$ sowie der Übertragungsfunktionsmatrix $\mathbf{G}(D) = (1 + D + D^2,\ 1 + D^2)$ ausgegangen. Die Länge der Informationssequenz sei $L = 5$ und unter Berücksichtigung der Terminierung $L' = 7$. Die betrachteten Codesequenzen $\underline{x}$ und auch die Empfangssequenzen $\underline{y}$ setzen sich somit jeweils aus 14 Bit zusammen. Durch Aufteilung entsprechend $\underline{y} = (\underline{y}_1, \underline{y}_2, \ \text{...} \ , \underline{y}_7)$ ergeben sich die Bitpaare $\underline{y}_i \in \{00, 01, 10, 11\}$.



Die Viterbi–Decodierung erfolgt mit Hilfe des dargestellten Trellisdiagramms. Ein roter Pfeil steht für die Hypothese $u_i = 0$, ein blauer für $u_i = 1$. Um Verwechslungen zu vermeiden, versehen wir hypothetische Größen mit Apostroph. Auch dann, wenn wir sicher wissen, dass das Informationsbit $u_i = 1$ ist, gilt $u_i' \in \{0, 1\}$. Neben den Pfeilen steht die jeweilige hypothetische Codesequenz $\underline{x}_i' \in \{00, 01, 10, 11\}$.

Wir gehen auf dieser und den nachfolgenden Seiten davon aus, dass die Viterbi–Decodierung auf der **Hamming–Distanz** $d_{\rm H}(\underline{x}_i', \underline{y}_i)$ zwischen dem Empfangswort $\underline{y}_i$ und den vier möglichen Codeworten $x_i' \in \{00, 01, 10, 11\}$ basiert. Dann gehen wir wie folgt vor:

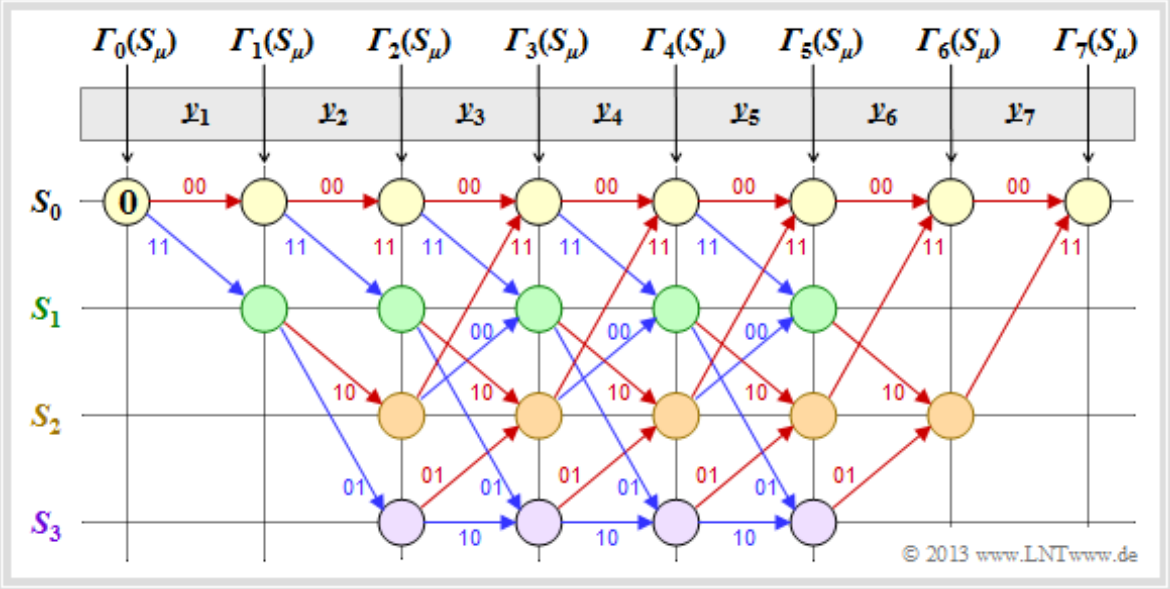
- In den noch leeren Kreisen werden die Fehlergrößen $\Gamma_i(S_\mu)$ der Zustände S_μ $(0 \le \mu \le 3)$ zu den Zeitpunkten i eingetragen. Der Startwert ist stets $\Gamma_0(S_0) = 0$.
- Die Fehlergrößen für $i = 1$ und $i = 2$ ergeben sich zu

$$\Gamma_1(S_0) = d_{\rm H}\big((00)\,,\,\underline{y}_1\big)\,, \qquad \Gamma_1(S_1) = d_{\rm H}\big((11)\,,\,\underline{y}_1\big)\,,$$
$$\Gamma_2(S_0) = \Gamma_1(S_0) + d_{\rm H}\big((00)\,,\,\underline{y}_2\big)\,, \qquad \Gamma_2(S_1) = \Gamma_1(S_0) + d_{\rm H}\big((11)\,,\,\underline{y}_2\big)\,,$$
$$\Gamma_2(S_2) = \Gamma_1(S_1) + d_{\rm H}\big((10)\,,\,\underline{y}_2\big)\,, \qquad \Gamma_2(S_3) = \Gamma_1(S_1) + d_{\rm H}\big((01)\,,\,\underline{y}_2\big)\,.$$

Die Beschreibung wird auf der nächsten Seite fortgesetzt.

Vorbemerkungen zu den nachfolgenden Decodierbeispielen (2)

Fortsetzung der allgemeinen Viterbi–Beschreibung:



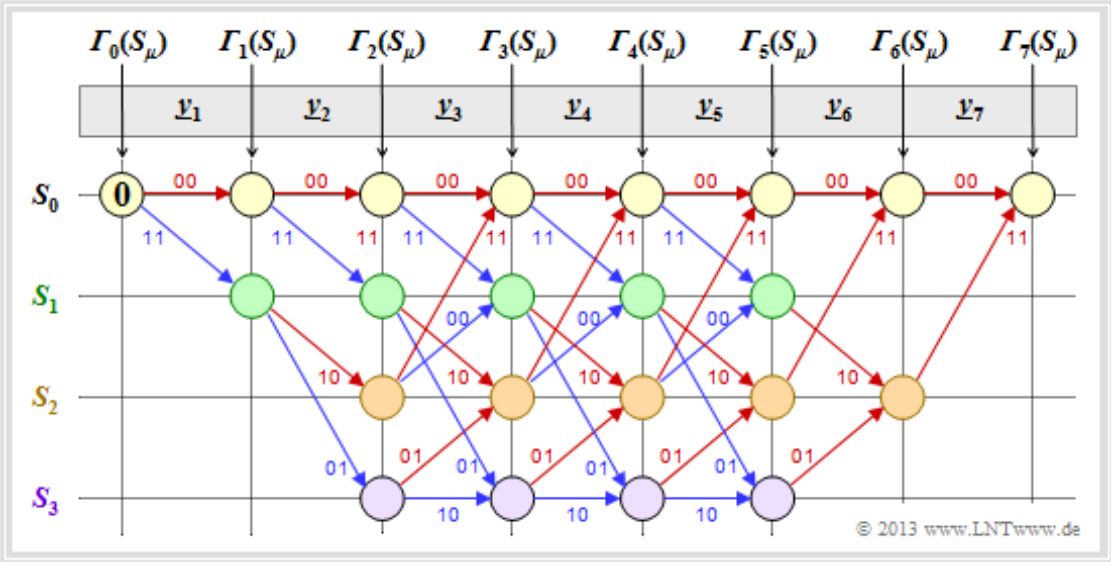
- Ab dem Zeitpunkt $i = 3$ hat das Trellisdiagramm seine Grundform erreicht, und zur Berechnung aller $\Gamma_i(S_\mu)$ muss jeweils das Minimum zwischen zwei Summen ermittelt werden:

$$\Gamma_i(S_0) = \text{Min}\left[\Gamma_{i-1}(S_0) + d_\text{H}\big((00)\,,\,\underline{y}_i\big)\,,\;\; \Gamma_{i-1}(S_2) + d_\text{H}\big((11)\,,\,\underline{y}_i\big)\right]\,,$$
$$\Gamma_i(S_1) = \text{Min}\left[\Gamma_{i-1}(S_0) + d_\text{H}\big((11)\,,\,\underline{y}_i\big)\,,\;\; \Gamma_{i-1}(S_2) + d_\text{H}\big((00)\,,\,\underline{y}_i\big)\right]\,,$$
$$\Gamma_i(S_2) = \text{Min}\left[\Gamma_{i-1}(S_1) + d_\text{H}\big((10)\,,\,\underline{y}_i\big)\,,\;\; \Gamma_{i-1}(S_3) + d_\text{H}\big((01)\,,\,\underline{y}_i\big)\right]\,,$$
$$\Gamma_i(S_3) = \text{Min}\left[\Gamma_{i-1}(S_1) + d_\text{H}\big((01)\,,\,\underline{y}_i\big)\,,\;\; \Gamma_{i-1}(S_3) + d_\text{H}\big((10)\,,\,\underline{y}_i\big)\right]\,.$$

- Von den zwei an einem Knoten $\Gamma_i(S_\mu)$ ankommenden Zweigen wird der schlechtere (der zu einem größeren $\Gamma_i(S_\mu)$ geführt hätte) eliminiert. Zu jedem Knoten führt dann nur noch ein einziger Zweig.
- Sind alle Fehlergrößen bis einschließlich $i = 7$ ermittelt, so kann der Viterbi–Algotithmus mit der Suche das zusammenhängenden Pfades vom Ende des Trellis $\Rightarrow \Gamma_7(S_0)$ bis zum Anfang $\Rightarrow \Gamma_0(S_0)$ abgeschlossen werden.
- Durch diesen Pfad liegen dann die am wahrscheinlichsten erscheinende Codesequenz $\underline{z}$ und die wahrscheinlichste Informationssequenz $\underline{v}$ fest. Nicht für alle Empfangssequenzen $\underline{y}$ gilt aber $\underline{z} = \underline{x}$ und $\underline{v} = \underline{u}$. Das heißt: Bei zu vielen Übertragungsfehlern versagt auch der Viterbi–Algorithmus.

Decodierbeispiel für den fehlerfreien Fall (1)

Zunächst gehen wir von der Empfangssequenz $\underline{y} = (11, 01, 01, 11, 11, 10, 11)$ aus, die hier – wegen der Codewortlänge $n = 2$ – bereits in Bitpaare $\underline{y}_1, \ldots, \underline{y}_7$ unterteilt ist.



Die eingetragenen Zahlenwerte und die verschiedenen Stricharten werden im folgenden Text erklärt:

- Ausgehend vom Initialwert $\Gamma_0(S_0) = 0$ kommt man mit $\underline{y}_1 = (11)$ durch Addition der Hamming-Distanzen $d_{\rm H}\,((00), \underline{y}_1) = 2$ bzw. $d_{\rm H}\,((11), \underline{y}_1) = 0$ zu den Fehlergrößen $\Gamma_1(S_0) = 2$, $\Gamma_1(S_1) = 0$.

- Im zweiten Decodierschritt gibt es Fehlergrößen für alle vier Zustände: Mit $\underline{y}_2 = (01)$ erhält man:

$$\begin{aligned}\Gamma_2(S_0) &= \Gamma_1(S_0) + d_{\rm H}\big((00)\,,\,(01)\big) = 2+1 = 3,\\ \Gamma_2(S_1) &= \Gamma_1(S_0) + d_{\rm H}\big((11)\,,\,(01)\big) = 2+1 = 3,\\ \Gamma_2(S_2) &= \Gamma_1(S_1) + d_{\rm H}\big((10)\,,\,(01)\big) = 0+2 = 2,\\ \Gamma_2(S_3) &= \Gamma_1(S_1) + d_{\rm H}\big((01)\,,\,(01)\big) = 0+0 = 0.\end{aligned}$$

- In allen weiteren Decodierschritten müssen jeweils zwei Werte verglichen werden, wobei dem Knoten $\Gamma_i(S_\mu)$ stets der kleinere Wert zugewiesen wird. Beispielsweise gilt für $i = 3$ mit $\underline{y}_3 = (01)$:

$$\begin{aligned}\Gamma_3(S_0) &= \min\big[\Gamma_2(S_0) + d_{\rm H}\big((00)\,,\,(01)\big)\,,\;\Gamma_2(S_2) + d_{\rm H}\big((11)\,,\,(01)\big)\big] =\\ &= \min\big[3+1\,,\,2+1\big] = 3,\\ \Gamma_3(S_3) &= \min\big[\Gamma_2(S_1) + d_{\rm H}\big((01)\,,\,(01)\big)\,,\;\Gamma_2(S_3) + d_{\rm H}\big((10)\,,\,(01)\big)\big] =\\ &= \min\big[3+0\,,\,0+2\big] = 2.\end{aligned}$$

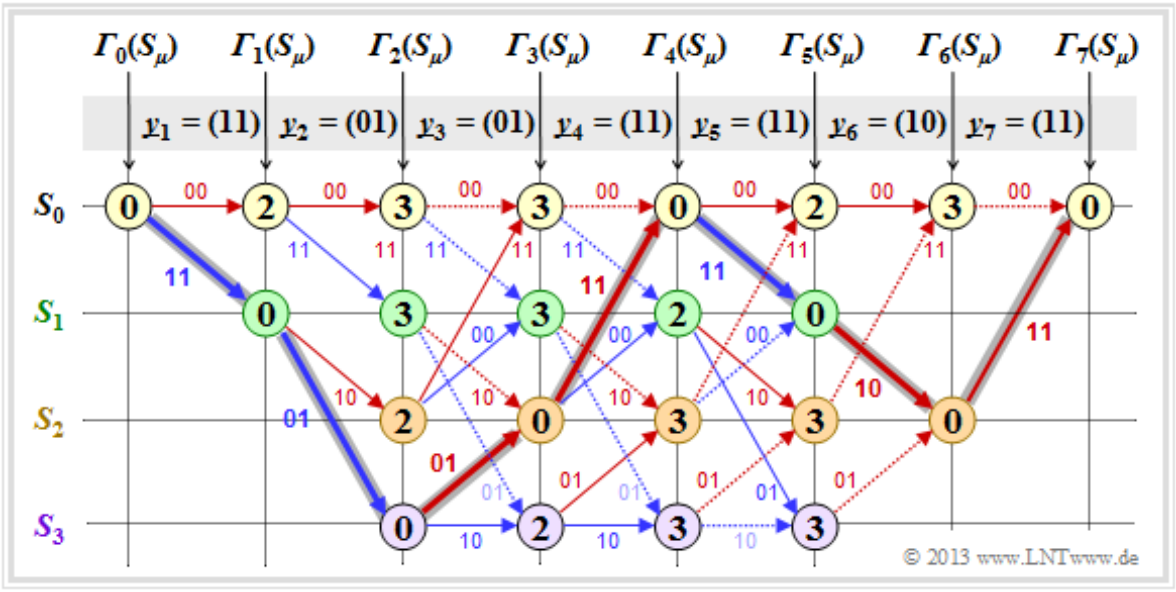
- Ab $i = 6$ wird im betrachteten Beispiel die Terminierung des Faltungscodes wirksam. Hier sind nur noch zwei Vergleiche zur Bestimmung von $\Gamma_6(S_0)$ und $\Gamma_6(S_2)$ anzustellen, und für $i = 7$ nur noch ein Vergleich mit dem Endergebnis $\Gamma_7(S_0)$.

- Von den jeweils zwei an einem Knoten ankommenden Zweigen wird stets nur derjenige bei der abschließenden Pfadsuche herangezogen, der zur minimalen Fehlergröße $\Gamma_i(S_\mu)$ geführt hat. Die „schlechten” Zweige werden verworfen. Sie sind in obiger Grafik jeweils punktiert dargestellt.

Die Beschreibung wird auf der nächsten Seite fortgesetzt.

Decodierbeispiel für den fehlerfreien Fall (2)

Nachdem alle Fehlergrößen $\Gamma_i(S_\mu)$ – in dem vorliegenden Beispiel für $1 \le i \le 7$ und $0 \le \mu \le 3$ – ermittelt wurden, kann der Viterbi–Decoder mit der Pfadsuche beginnen.



Die Pfadsuche läuft wie folgt ab:

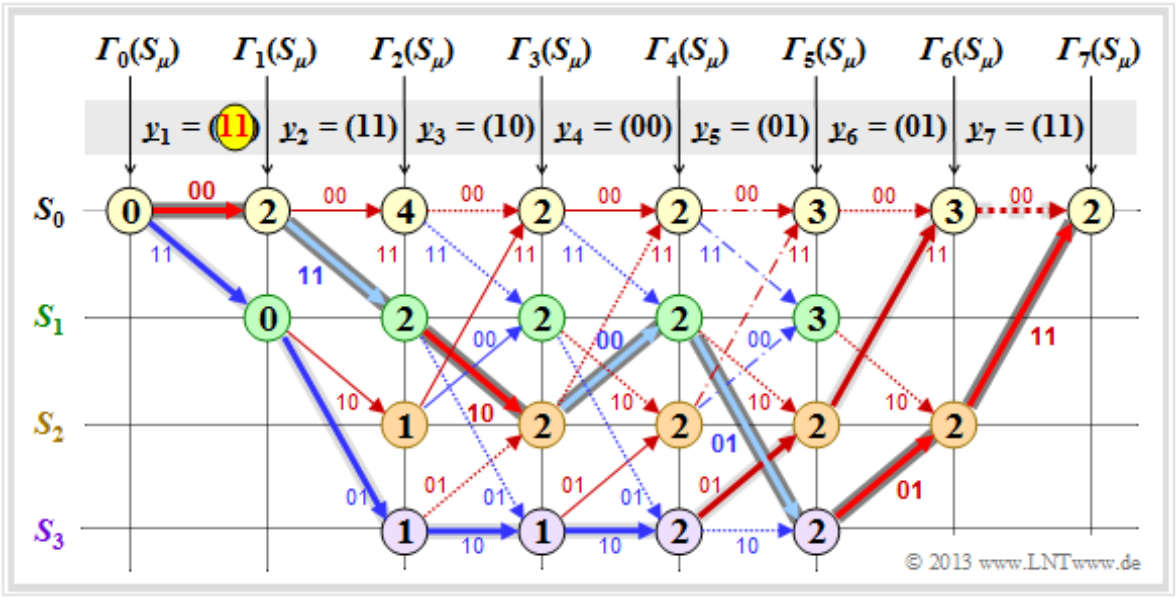
- Ausgehend vom Endwert $\Gamma_7(S_0)$ wird in Rückwärtsrichtung ein zusammenhängender Pfad bis zum Startwert $\Gamma_0(S_0)$ gesucht. Erlaubt sind nur die durchgezogenen Zweige. Punktierte Linien können nicht Teil des ausgewählten Pfades sein.
- Der ausgewählte Pfad ist in der Grafik grau markiert. Er durchläuft von rechts nach links die Zustände $S_0 \leftarrow S_2 \leftarrow S_1 \leftarrow S_0 \leftarrow S_2 \leftarrow S_3 \leftarrow S_1 \leftarrow S_0$. Es gibt keinen zweiten durchgehenden Pfad von $\Gamma_7(S_0)$ zu $\Gamma_0(S_0)$. Das bedeutet: Das Decodierergebnis ist eindeutig.
- Das Ergebnis $\underline{v} = (1, 1, 0, 0, 1, 0, 0)$ des Viterbi–Decoders hinsichtlich der Informationssequenz erhält man, wenn man für den durchgehenden Pfad – nun aber in Vorwärtsrichtung von links nach rechts – die Farben der einzelnen Zweige auswertet (rot entspricht einer 0, blau einer 1).

Aus dem Endwert $\Gamma_7(S_0) = 0$ erkennt man, dass in diesem ersten Beispiel keine Übertragungsfehler vorlagen. Das Decodierergebnis $\underline{z}$ stimmt also mit dem Empfangsvektor $\underline{y} = (11, 01, 01, 11, 11, 10, 11)$ und der tatsächlichen Codesequenz $\underline{x}$ überein. Außerdem ist $\underline{v}$ nicht nur die nach dem ML–Kriterium wahrscheinlichste Informationssequenz $\underline{u}$, sondern es gilt auch hier die Identität: $\underline{v} = \underline{u}$.

Anmerkung: Bei der beschriebenen Decodierung wurde von der bereits in der Überschrift enthaltenen Information „Fehlerfreier Fall" natürlich kein Gebrauch gemacht wurde.

Decodierbeispiele für den fehlerbehafteten Fall (1)

Es folgen zwei weitere Beispiele zur Viterbi–Decodierung. Die Berechnung der Fehlergrößen $\Gamma_i(S_\mu)$ geschieht wie auf **Seite 2** beschrieben und auf der letzten Seite für den fehlerfreien Fall demonstriert.



Als Resümee dieses ersten Beispiels mit obigem Trellis ist festzuhalten:

- Auch hier lässt sich ein eindeutiger Pfad (dunkelgraue Markierung) zurückverfolgen, der zu den folgenden Ergebnissen führt (erkennbar an den Beschriftungen bzw. den Farben dieses Pfades):

$$\underline{z} = (00, 11, 10, 00, 01, 01, 11),$$
$$\underline{v} = (0, 1, 0, 1, 1, 0, 0).$$

- Der Vergleich der am wahrscheinlichsten gesendeten Codesequenz $\underline{z}$ mit dem Empfangsvektor $\underline{y}$ zeigt, dass hier zwei Bitfehler (am Beginn) vorlagen. Da aber der verwendete Faltungscode die **freie Distanz $d_F = 5$** aufweist, führen zwei Fehler noch nicht zu einem falschen Decodierergebnis.

- Es gibt andere Pfade wie zum Beispiel den heller markierten Pfad ($S_0 \rightarrow S_1 \rightarrow S_3 \rightarrow S_3 \rightarrow S_3 \rightarrow S_2 \rightarrow S_0 \rightarrow S_0$), die zunächst als vielversprechend erscheinen. Erst im letzten Decodierschritt ($i = 7$) kann dieser hellgraue Pfad endgültig verworfen werden.

- Dieses Beispiel zeigt, dass eine zu frühe Entscheidung oft zu einem Decodierfehler führt, und man erkennt auch die Zweckmäßigkeit der Terminierung: Bei endgültiger Entscheidung zum Zeitpunkt $i = 5$ (dem Ende der eigentlichen Informationssequenz) wären die Sequenzen (0, 1, 0, 1, 1) und (1, 1, 1, 1, 0) noch als gleichwahrscheinlich angesehen worden.

Anmerkung: Bei der Berechnung von $\Gamma_5(S_0) = 3$ und $\Gamma_5(S_1) = 3$ führen hier jeweils die beiden Vergleichszweige zur gleichen minimalen Fehlergröße. In der Grafik sind diese beiden Sonderfälle durch Strichpunktierung markiert.

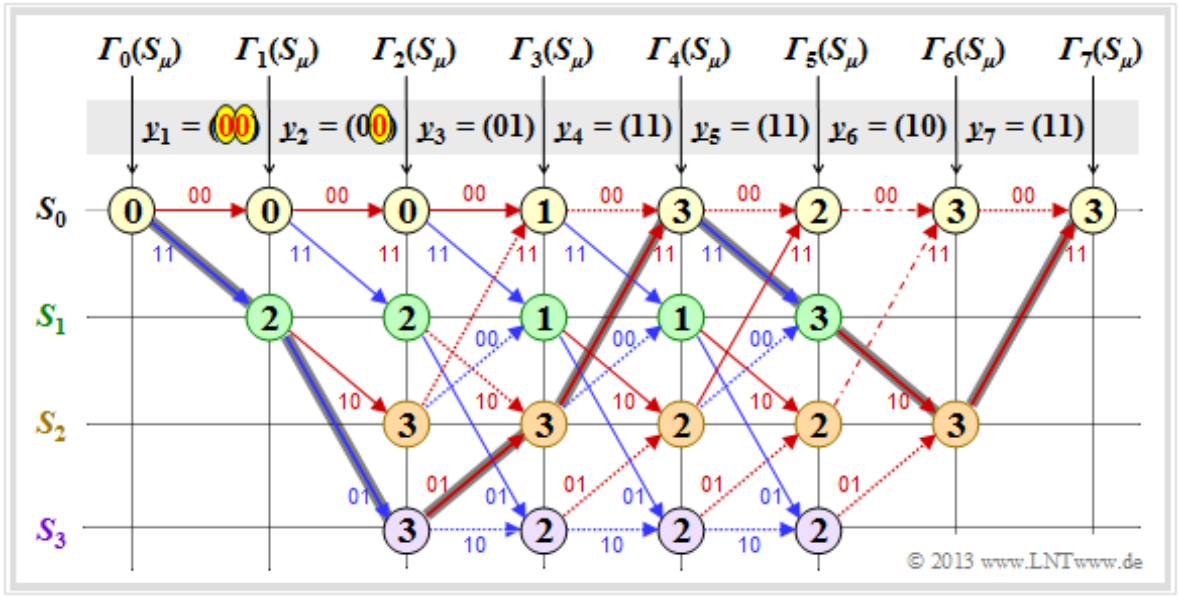
In diesem Beispiel hat dieser Sonderfall keine Auswirkung auf die Pfadsuche. Der Algorithmus erwartet trotzdem stets eine Entscheidung zwischen zwei konkurrierenden Zweigen. In der Praxis hilft man sich dadurch, dass man bei Gleichheit einen der beiden Pfade zufällig auswählt.

Decodierbeispiele für den fehlerbehafteten Fall (2)

Im letzten Beispiel gehen wir stets von folgenden Voraussetzungen bezüglich Quelle und Coder aus:

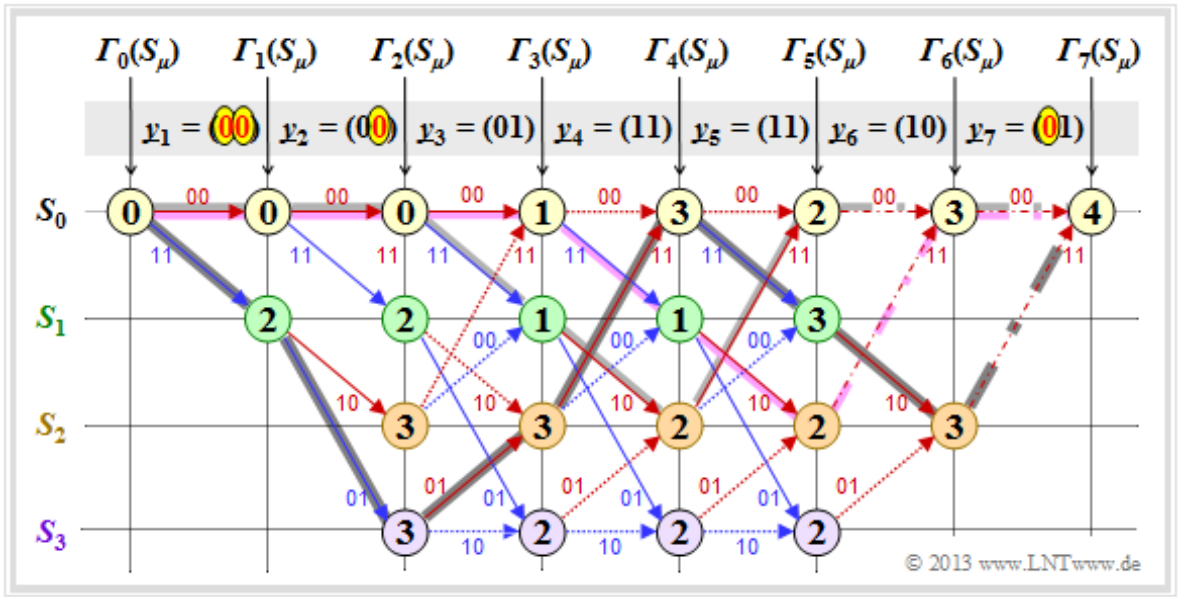
$$\underline{u} = (1, 1, 0, 0, 1, 0, 0) \quad \Rightarrow \quad \underline{x} = (11, 01, 01, 11, 11, 10, 11)\,.$$

Aus der ersten der beiden Grafiken erkennt man, dass sich hier der Decoder trotz dreier Bitfehler für den richtigen Pfad (dunkle Hinterlegung) entscheidet. Es gibt also nicht immer eine Fehlentscheidung, wenn mehr als $d_F/2$ Bitfehler aufgetreten sind.



In der unteren Grafik ist noch ein vierter Bitfehler in Form von $\underline{y}_7 = (01)$ hinzugefügt:

- Nun führen beide Zweige im Schritt $i = 7$ zur minimalen Fehlergröße $\Gamma_7(S_0) = 4$, erkennbar an den strichpunktierten Übergängen. Entscheidet man sich im dann erforderlichen Losverfahren für den dunkel hinterlegten Pfad, so wird auch bei vier Bitfehlern noch die richtige Entscheidung getroffen.
- Andernfalls kommt es zu einer Fehlentscheidung. Je nachdem, wie das Auswürfeln im Schritt $i = 6$ zwischen den beiden strichpunktierten Konkurrenten ausgeht, entscheidet man sich entweder für den violetten oder den hellgrauen Pfad. Mit dem richtigen Pfad haben beide wenig gemein.



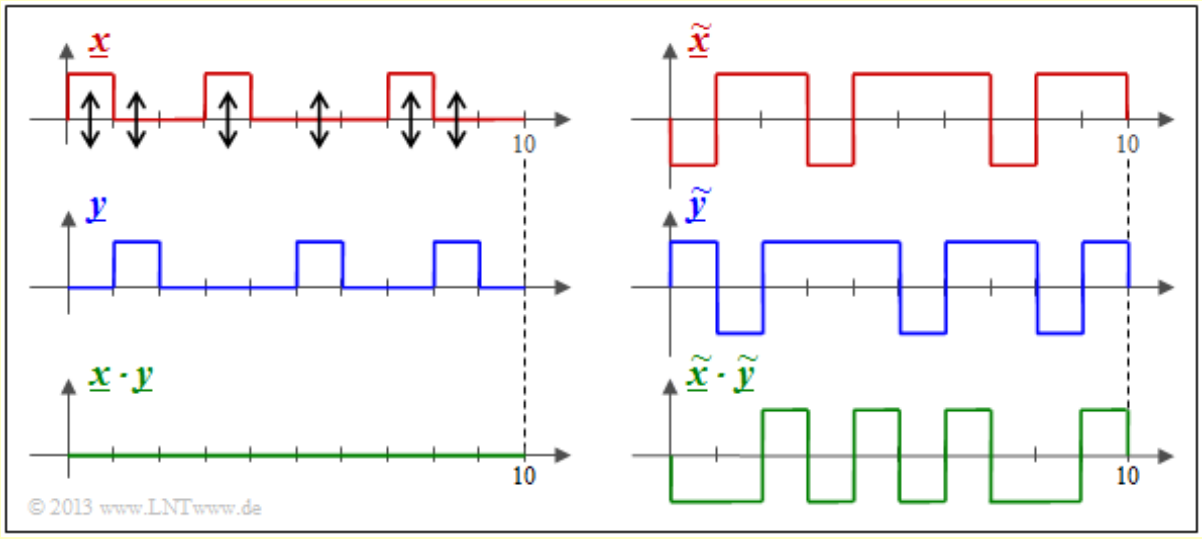
Zusammenhang zwischen Hamming–Distanz und Korrelation

Insbesondere beim **BSC–Modell** – aber auch bei jedem anderen Binärkanal ⇒ Eingang $x_i \in \{0, 1\}$, Ausgang $y_i \in \{0, 1\}$ wie zum Beispiel dem **Gilbert–Elliott–Modell** – liefert die Hamming–Distanz $d_H(\underline{x}, \underline{y})$ genau die gleiche Information über die Ähnlichkeit der Eingangsfolge $\underline{x}$ und der Ausgangsfolge $\underline{y}$ wie das **innere Produkt.** Nimmt man an, dass die beiden Folgen in bipolarer Darstellung vorliegen (gekennzeichnet durch Tilden) und dass die Folgenlänge jeweils L ist, so gilt für das innere Produkt:

$$<\underline{\tilde{x}}, \underline{\tilde{y}}> = \sum_{i=1}^{L} \tilde{x}_i \cdot \tilde{y}_i \quad \text{mit} \quad \tilde{x}_i = 1 - 2 \cdot x_i\,, \quad \tilde{y}_i = 1 - 2 \cdot y_i\,, \quad \tilde{x}_i, \tilde{y}_i \in \{-1, +1\}\,.$$

Wir bezeichnen dieses innere Produkt manchmal auch als „Korrelationswert". Die Anführungszeichen sollen darauf hinweisen, dass der Wertebereich eines **Korrelationskoeffizienten** eigentlich ±1 ist.

Beispiel: Wir betrachten hier zwei Binärfolgen der Länge $L = 10$:



- Links dargestellt sind die unipolaren Folgen $\underline{x}$ und $\underline{y}$ sowie das Produkt $\underline{x} \cdot \underline{y}$. Man erkennt die Hamming–Distanz $d_H(\underline{x}, \underline{y}) = 6$ ⇒ sechs Bitfehler an den Pfeilpositionen. Das innere Produkt $\langle \underline{x} \cdot \underline{y} \rangle = 0$ hat hier keine Aussagekraft. Zum Beispiel ist $\langle \underline{0} \cdot \underline{y} \rangle$ unabhängig von $\underline{y}$ stets Null.
- Die Hamming–Distanz $d_H = 6$ erkennt man auch aus der bipolaren (antipodalen) Darstellung der rechten Grafik. Die „Korrelationswert" hat aber nun den richtigen Wert $4 \cdot (+1) + 6 \cdot (–1) = –2$. Es gilt der deterministische Zusammenhang zwischen den beiden Größen mit der Folgenlänge L:

$$<\underline{\tilde{x}}, \underline{\tilde{y}}> = L - 2 \cdot d_H(\underline{\tilde{x}}, \underline{\tilde{y}})\,.$$

Interpretieren wir nun diese Gleichung für einige Sonderfälle:

- Identische Folgen: Die Hamming–Distanz ist gleich 0 und der „Korrelationswert" gleich L.
- Invertierte: Folgen: Die Hamming–Distanz ist gleich L und der „Korrelationswert" gleich $–L$.
- Unkorrelierte Folgen: Die Hamming–Distanz ist gleich $L/2$, der „Korrelationswert" gleich 0.

Viterbi–Algorithmus, basierend auf Korrelation und Metriken (1)

Mit den Erkenntnissen der letzten Seite lässt sich der Viterbi–Algorithmus auch wie folgt charakterisieren. Der Algorithmus sucht von allen möglichen Codesequenzen $\underline{x}' \in C$ die Sequenz $\underline{z}$ mit der maximalen Korrelation zur Empfangssequenz $\underline{y}$:

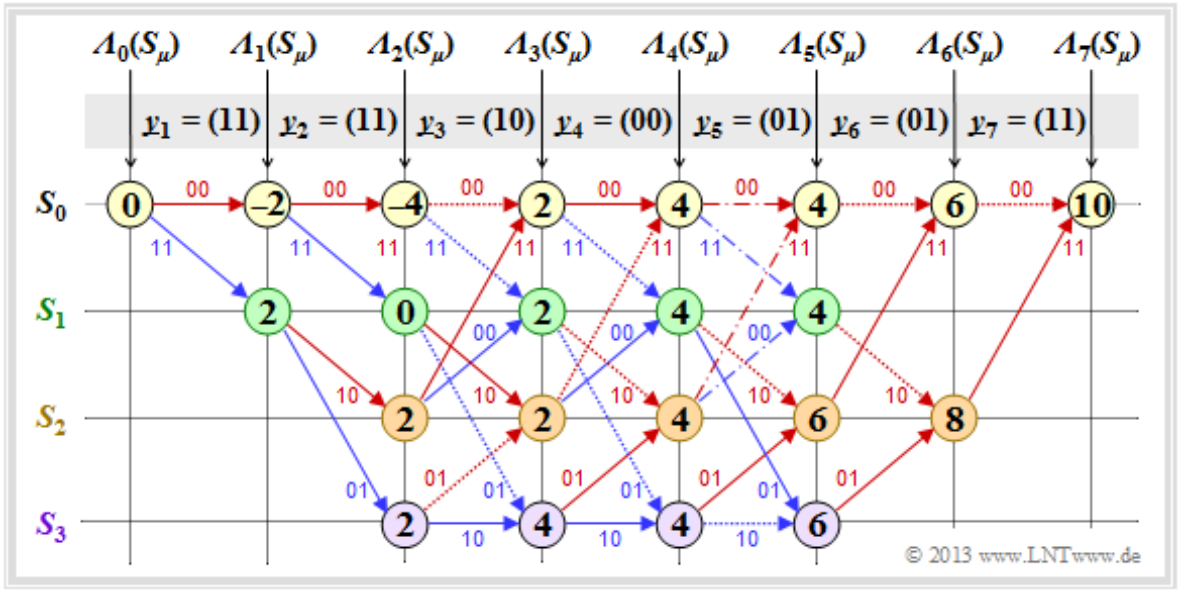
$$\underline{z} = \arg \max_{\underline{x}' \in C} \langle \underline{\tilde{x}}', \underline{\tilde{y}} \rangle \quad \text{mit} \quad \underline{\tilde{x}}' = 1 - 2 \cdot \underline{x}', \ \underline{\tilde{y}} = 1 - 2 \cdot \underline{y}.$$

$\langle \ ... \ \rangle$ bezeichnet einen „Korrelationswert" entsprechend den Aussagen auf der letzten Seite. Die Tilden weisen wieder auf die bipolare (antipodale) Darstellung hin.

Die Grafik zeigt die diesbezügliche Trellisauswertung. Zugrunde liegt

- der gleiche Faltungscode mit $R = 1/2$, $m = 2$ und $\mathbf{G}(D) = (1 + D + D^2, 1 + D^2)$, und
- der gleiche Empfangsvektor $\underline{y} = (11, 11, 10, 00, 01, 01, 11)$

wie für das **Trellisdiagramm** auf Seite 3a dieses Kapitels, basierend auf der minimalen Hamming–Distanz und den Fehlergrößen $\Gamma_i(S_\mu)$. Beide Darstellungen ähneln sich sehr.



Ebenso wie die Suche nach der Sequenz mit der minimalen Hamming–Distanz geschieht auch die *Suche nach dem maximalen Korrelationswert* schrittweise:

- Die Knoten bezeichnet man hier als die Metriken $\Lambda_i(S_\mu)$. Die englische Bezeichnung hierfür ist *Cumulative Metric*, während *Branch Metric* den Metrikzuwachs angibt.
- Der Endwert $\Lambda_7(S_0) = 10$ gibt den „Korrelationswert" zwischen der ausgewählten Folge $\underline{z}$ und dem Empfangsvektor $\underline{y}$ an. Im fehlerfreien Fall ergäbe sich $\Lambda_7(S_0) = 14$.

Die Trellisbeschreibung wird auf der nächsten Seite fortgesetzt.

Viterbi–Algorithmus, basierend auf Korrelation und Metriken (2)

Die nachfolgende Beschreibung bezieht sich auf die Trellisauswertung entsprechend der letzten Seite, basierend auf **„Korrelationswerten" und den Metriken $\Lambda_i(S_\mu)$.** Zum Vergleich zeigt die Grafik auf Seite 3a Trellisauswertung, die auf **Hamming–Distanzen und den Fehlergrößen $\Gamma_i(S_\mu)$** basieren.

- Die Metriken zum Zeitpunkt $i = 1$ ergeben sich mit $\underline{y}_1 = (11)$ zu

$$\Lambda_1(S_0) = <(00), (11)> = <(+1, +1), (-1, -1)> = -2,$$
$$\Lambda_1(S_1) = <(11), (11)> = <(-1, -1), (-1, -1)> = +2.$$

- Entsprechend gilt zum Zeitpunkt $i = 2$ mit $\underline{y}_2 = (11)$:

$$\Lambda_2(S_0) = \Lambda_1(S_0) + <(00), (11)> = -2 - 2 = -4,$$
$$\Lambda_2(S_1) = \Lambda_1(S_0) + <(11), (11)> = -2 + 2 = 0,$$
$$\Lambda_2(S_2) = \Lambda_1(S_1) + <(10), (11)> = 2 + 0 = 2,$$
$$\Lambda_2(S_3) = \Lambda_1(S_1) + <(01), (11)> = 2 + 0 = 2.$$

- Ab dem Zeitpunkt $i = 3$ muss eine Entscheidung zwischen zwei Metriken getroffen werden. Beispielsweise erhält man mit $\underline{y}_3 = (10)$ für die oberste und die unterste Metrik im Trellis:

$$\Lambda_3(S_0) = \max\left[\Lambda_2(S_0) + <(00), (11)>, \Lambda_2(S_1) + <(00), (11)>\right] =$$
$$= \max\left[-4 + 0, 2 + 0\right] = 2,$$
$$\Lambda_3(S_3) = \max\left[\Lambda_2(S_1) + <(01), (10)>, \Lambda_2(S_3) + <(10), (10)>\right] =$$
$$= \max\left[0 + 0, 2 + 2\right] = 4.$$

Vergleicht man die zu minimierenden Fehlergrößen $\Gamma_i(S_\mu)$ mit den zu maximierenden Metriken $\Lambda_i(S_\mu)$, so erkennt man den folgenden deterministischen Zusammenhang:

$$\Lambda_i(S_\mu) = 2 \cdot \left[i - \Gamma_i(S_\mu)\right].$$

Die Auswahl der zu den einzelnen Decodierschritten überlebenden Zweige ist bei beiden Verfahren identisch, und auch die Pfadsuche liefert das gleiche Ergebnis.

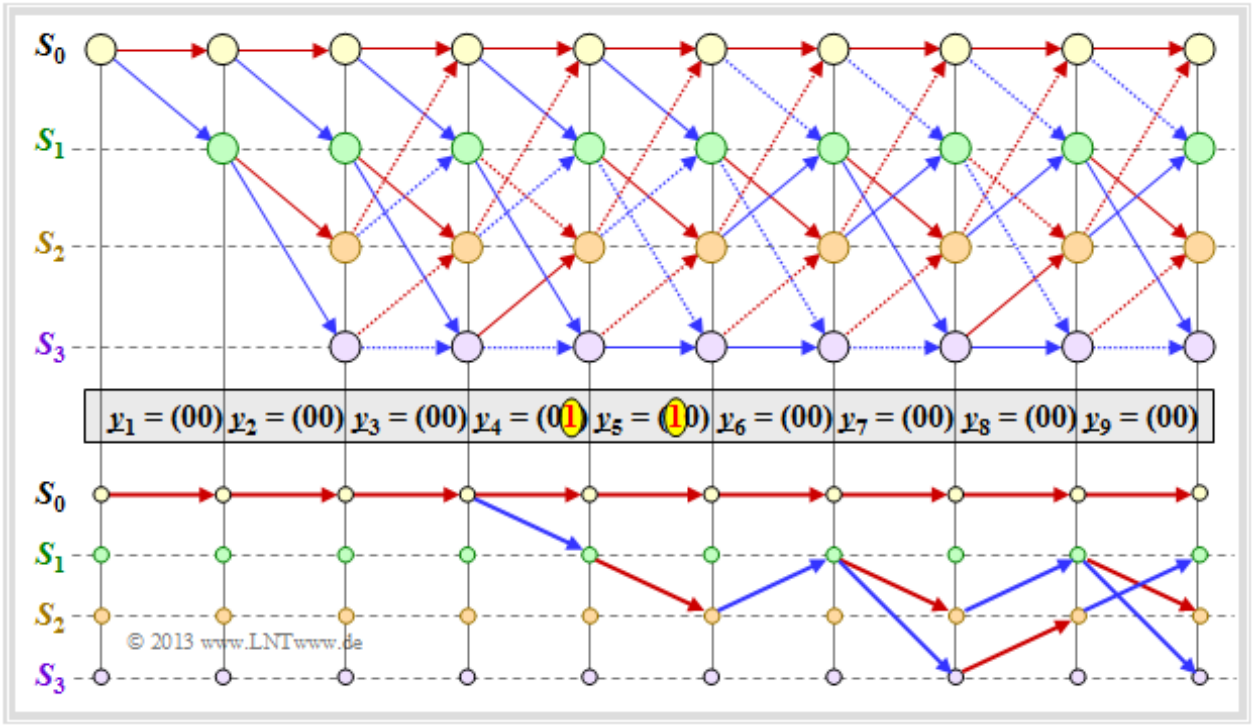
Zusammenfassung:

- Beim Binärkanal – zum Beispiel dem BSC–Modell – führen die beiden beschriebenen Viterbi–Varianten *Fehlergrößenminimierung* und *Metrikmaximierung* zum gleichen Ergebnis.
- Beim AWGN–Kanal ist die Fehlergrößenminimierung nicht anwendbar, da keine Hamming–Distanz zwischen dem binären Eingang $\underline{x}$ und dem analogen Ausgang $\underline{y}$ angegeben werden kann.
- Die Metrikmaximierung ist beim AWGN–Kanal vielmehr identisch mit der Minimierung der **Euklidischen Distanz** – siehe **Aufgabe Z3.10.**
- Ein weiterer Vorteil der Metrikmaximierung ist, dass eine Zuverlässigkeitsinformation über die Empfangswerte $\underline{y}$ in einfacher Weise berücksichtigt werden kann.

Viterbi–Entscheidung bei nicht–terminierten Faltungscodes (1)

Bisher wurde stets ein terminierter Faltungscode der Länge $L' = L + m$ betrachtet, und das Ergebnis des Viterbi–Decoders war der durchgehende Trellispfad vom Startzeitpunkt ($i = 0$) bis zum Ende ($i = L'$).

Bei nicht–terminierten Faltungscodes ($L' \to \infty$) ist diese Entscheidungsstrategie nicht anwendbar. Hier muss der Algorithmus abgewandelt werden, um in endlicher Zeit eine bestmögliche Schätzung (gemäß Maximum–Likelihood) der einlaufenden Bits der Codesequenz liefern zu können.



Die obere Grafik zeigt ein beispielhaftes Trellis für

- unseren Standard–Codierer $\Rightarrow$ $R = 1/2$, $m = 2$, $G(D) = (1 + D + D^2, 1 + D^2)$,
- die Nullfolge $\Rightarrow$ $\underline{u} = \underline{0} = (0, 0, 0, ...)$ $\Rightarrow$ $x = \underline{0} = (00, 00, 00, ...)$,
- jeweils einen Übertragungsfehler bei $i = 4$ und $i = 5$.

Anhand der Stricharten erkennt man erlaubte (durchgezogene) und verbotene (punktierte) Pfeile in rot ($u_i = 0$) und blau ($u_i = 1$). Punktierte Linien haben einen Vergleich gegen einen Konkurrenten verloren und können nicht Teil des ausgewählten Pfades sein.

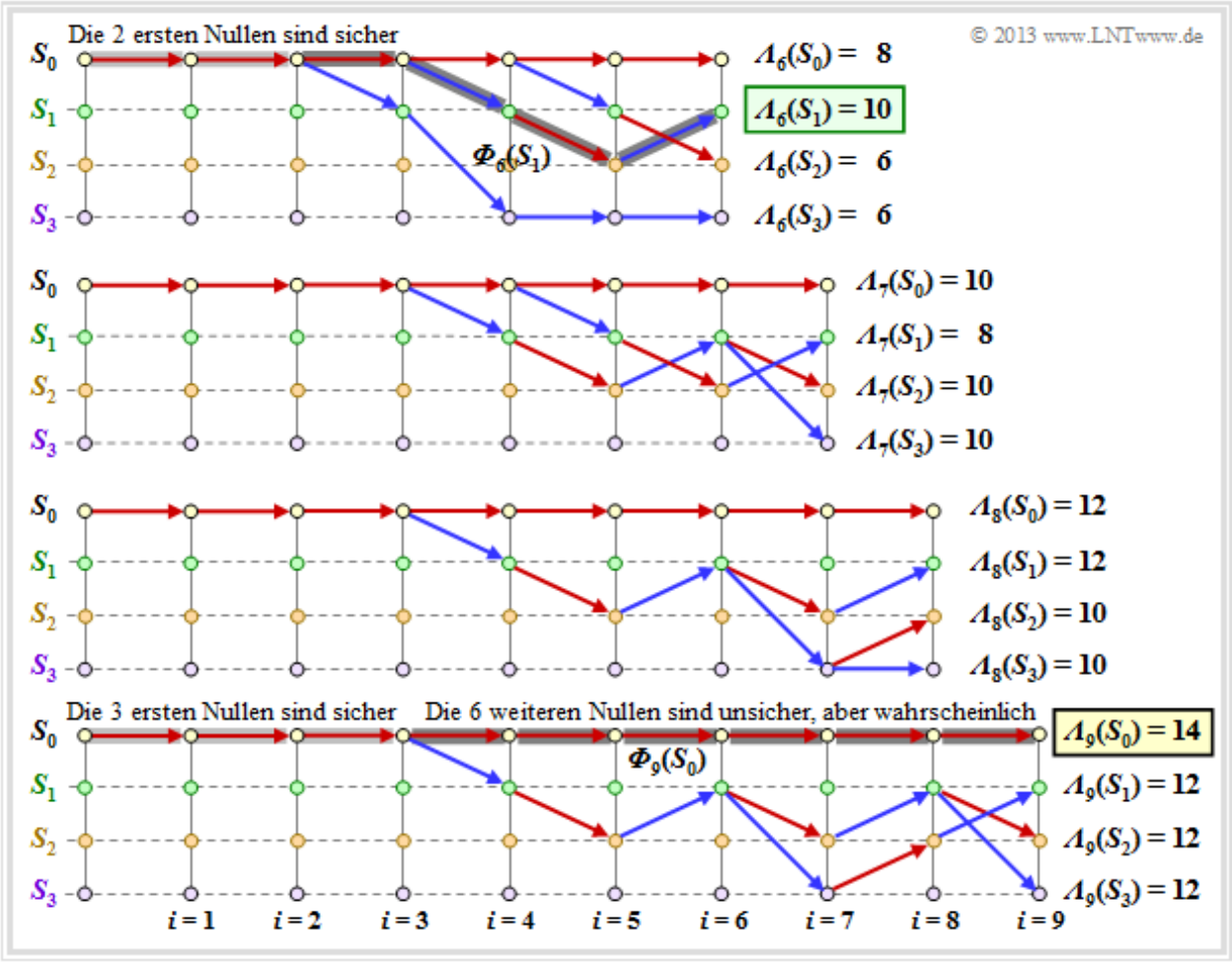
Die untere Grafik zeigt die 2^m überlebenden Pfade $\Phi_i\ (S_\mu)$ für den Zeitpunkt $i = 9$. Man findet diese Pfade am einfachsten von rechts nach links. Die folgende Angabe zeigt die durchlaufenen Zustände S_μ in Vorwärtsrichtung:

$$\Phi_9(S_0): S_0 \to S_0 \to S_0 \to S_0 \to S_0 \to S_0 \to S_0 \to S_0 \to S_0,$$
$$\Phi_9(S_1): S_0 \to S_0 \to S_0 \to S_0 \to S_1 \to S_2 \to S_1 \to S_3 \to S_2 \to S_1,$$
$$\Phi_9(S_2): S_0 \to S_0 \to S_0 \to S_0 \to S_1 \to S_2 \to S_1 \to S_2 \to S_1 \to S_2,$$
$$\Phi_9(S_3): S_0 \to S_0 \to S_0 \to S_0 \to S_1 \to S_2 \to S_1 \to S_2 \to S_1 \to S_3.$$

Die Beschreibung wird auf der nächsten Seite fortgesetzt.

Viterbi–Entscheidung bei nicht–terminierten Faltungscodes (2)

Definition: Der **überlebende Pfad** (englisch: *Survivor*) $\Phi_i(S_\mu)$ ist der durchgehende Pfad vom Start S_0 (bei $i = 0$) zum Knoten S_μ zum Zeitpunkt i. Empfehlenswert ist die Pfadsuche in Rückwärtsrichtung.



Die Grafik zeigt die überlebenden Pfade für die Zeitpunkte $i = 6$ bis $i = 9$. Zusätzlich sind die jeweiligen Metriken $\Lambda_i(S_\mu)$ für alle vier Zustände angegeben.Die Grafik ist wie folgt zu interpretieren:

- Zum Zeitpunkt $i = 9$ kann noch keine endgültige ML–Entscheidung über die ersten neun Bit der Informationssequenz getroffen werden. Allerdings ist bereits sicher, dass die wahrscheinlichste Bitfolge durch einen der Pfade $\Phi_9(S_0)$, ... , $\Phi_9(S_3)$ richtig wiedergegeben wird.
- Da alle vier Pfade bei $i = 3$ zusammenlaufen, ist die Entscheidung „$v_1 = 0, v_2 = 0, v_3 = 0$" die bestmögliche (hellgraue Hinterlegung). Auch zu einem späteren Zeitpunkt würde keine andere Entscheidung getroffen werden. Hinsichtlich der Bits v_4, v_5, ... sollte man sich noch nicht festlegen.
- Müsste man zum Zeitpunkt $i = 9$ eine Zwangsentscheidung treffen, so würde man sich für $\Phi_9(S_0)$ $\Rightarrow$ $\underline{v} = (0, 0, \text{...} , 0)$ entscheiden, da die Metrik $\Lambda_9(S_0) = 14$ größer ist als die Vergleichsmetriken.
- Die Zwangsentscheidung zum Zeitpunkt $i = 9$ führt in diesem Beispiel zum richtigen Ergebnis. Zum Zeitpunkt $i = 6$ wäre ein solcher Zwangsentscheid falsch gewesen $\Rightarrow$ $\underline{v} = (0, 0, 0, 1, 0, 1)$, und zu den Zeitpunten $i = 7$ bzw. $i = 8$ nicht eindeutig.

Weitere Decodierverfahren für Faltungscodes

Wir haben uns bisher in diesem Kapitel nur mit dem Viterbi–Algorithmus beschäftigt, der 1967 von A. J. Viterbi veröffentlicht wurde. Erst 1974 hat G. D. Forney nachgewiesen, dass dieser Algorithmus eine Maximum–Likelihood–Decodierung von Faltungscodes durchführt.

Aber schon in den Jahren zuvor waren viele Wissenschaftler sehr bemüht, effiziente Decodierverfahren für die 1955 erstmals von Peter Elias beschriebenen Faltungscodes bereitzustellen. Zu nennen sind hier unter Anderem – genauere Beschreibungen findet man beispielsweise in **[Bos99].**

- *Sequential Decoding* von J. M. Wozencraft und B. Reifen aus dem Jahre 1961,
- der Vorschlag von R. M. Fano (1963), der als *Fano–Algorithmus* bekannt wurde,
- die Arbeiten von K. Zigangirov (1966) und F. Jelinek (1969), deren Decodierverfahren häufig als *Stack–Algorithmus* bezeichnet wird.

Alle diese Decodierverfahren und auch der Viterbi–Algorithmus in seiner bisher beschriebenen Form liefern „hart" entschiedene Ausgangswerte $\Rightarrow v_i \in \{0, 1\}$. Oftmals wären jedoch Informationen über die Zuverlässigkeit der getroffenen Entscheidungen wünschenswert, insbesondere dann, wenn ein verkettetes Codierschema mit einem äußeren und einem inneren Code vorliegt.

Kennt man die Zuverlässigkeit der vom inneren Decoder entschiedenen Bits zumindest grob, so kann durch diese Information die Bitfehlerwahrscheinlichkeit des äußeren Decoders (signifikant) herabgesetzt werden. Der von J. Hagenauer in **[Hag90]** vorgeschlagene *Soft–Output–Viterbi–Algorithmus* (SOVA) erlaubt es, zusätzlich zu den entschiedenen Symbolen auch jeweils ein Zuverlässigkeitsmaß anzugeben.

Abschließend gehen wir noch etwas genauer auf den *BCJR–Algorithmus* ein, benannt nach dessen Erfinder L. R. Bahl, J. Cocke, F. Jelinek und J. Raviv **[BCJR74].** Während der Viterbi–Algorithmus nur eine Schätzung der Gesamtsequenz vornimmt $\Rightarrow$ **block–wise ML,** schätzt der BCJR–Algorithmus ein einzelnes Symbol (Bit) unter Berücksichtigung der gesamten empfangenen Codesequenz. Es handelt sich hierbei also um eine *symbolweise Maximum–Aposteriori–Decodierung* $\Rightarrow$ **bit–wise MAP.**

Der Unterschied zwischen Viterbi–Algorithmus und BCJR–Algorithmus soll – stark vereinfacht – am Beispiel eines terminierten Faltungscodes dargestellt werden:

- Der **Viterbi–Algorithmus** arbeitet das Trellis nur in einer Richtung – der *Vorwärtsrichtung* – ab und berechnet für jeden Knoten die Metriken $\Lambda_i(S_\mu)$. Nach Erreichen des Endknotens wird der überlebende Pfad gesucht, der die wahrscheinlichste Codesequenz kennzeichnet.
- Beim **BCJR–Algorithmus** wird das Trellis zweimal abgearbeitet, einmal in Vorwärtsrichtung und anschließend in *Rückwärtsrichtung*. Für jeden Knoten sind dann zwei Metriken angebbar, aus denen für jedes Bit die Aposterori–Wahrscheinlichkeit bestimmt werden kann.

Hinweis: Diese Kurzzusammenfassung basiert auf dem Lehrbuch **[Bos98].** Eine etwas ausführlichere Beschreibung des BCJR–Algorithmus' folgt im **Kapitel 4.1.**

Freie Distanz vs. Minimale Distanz

Eine äußerst wichtige Kenngröße hinsichtlich der Fehlerwahrscheinlichkeit eines linearen Blockcodes ist die *minimale Distanz* zwischen zwei Codeworten:

$$d_{\rm min}(\mathcal{C}) = \min_{\substack{\underline{x},\,\underline{x}'\in\mathcal{C}\\ \underline{x}\neq\underline{x}'}} d_{\rm H}(\underline{x},\,\underline{x}') = \min_{\substack{\underline{x}\in\mathcal{C}\\ \underline{x}\neq\underline{0}}} w_{\rm H}(\underline{x}).$$

Der zweite Gleichungsteil ergibt sich aus der Tatsache, dass jeder lineare Code auch das Nullwort ($\underline{0}$) beinhaltet. Zweckmäßigerweise setzt man deshalb $\underline{x}' = \underline{0}$, so dass die **Hamming–Distanz** $d_{\rm H}(\underline{x},\,\underline{0})$ das gleiche Ergebnis liefert wie das Hamming–Gewicht $w_{\rm H}(\underline{x})$.

Beispiel: Die nachfolgende Tabelle zeigt die 16 Codeworte des **(7, 4, 3)–Hamming–Codes.**

0 0 0 0 0 0 0	0 1 0 0 1 1 1	1 0 0 0 1 0 1	1 1 0 0 0 1 0
0 0 0 1 0 1 1	0 1 0 1 1 0 0	1 0 0 1 1 1 0	1 1 0 1 0 0 1
0 0 1 0 1 1 0	0 1 1 0 0 0 1	1 0 1 0 0 1 1	1 1 1 0 1 0 0
0 0 1 1 1 0 1	0 1 1 1 0 1 0	1 0 1 1 0 0 0	1 1 1 1 1 1 1

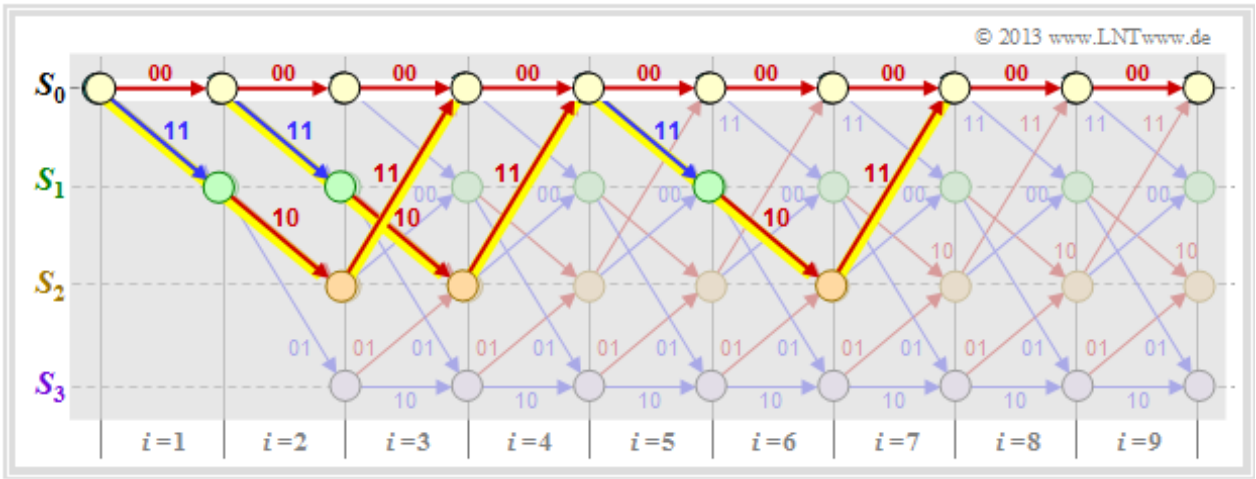
Alle Codeworte außer dem Nullwort ($\underline{0}$) beinhalten mindestens drei Einsen ⇒ $d_{\rm min} = 3$. Es gibt sieben Codeworte mit drei Einsen, sieben mit vier Einsen und je eines ohne Einsen bzw. mit sieben Einsen.

Die **freie Distanz** $d_{\rm F}$ eines Faltungscodes (*Convolution Code* ⇒ CC) unterscheidet sich formelmäßig nicht von der minimalen Distanz eines linearen Blockcodes:

$$d_{\rm F}(\mathcal{CC}) = \min_{\substack{\underline{x},\,\underline{x}'\in\mathcal{CC}\\ \underline{x}\neq\underline{x}'}} d_{\rm H}(\underline{x},\,\underline{x}') = \min_{\substack{\underline{x}\in\mathcal{CC}\\ \underline{x}\neq\underline{0}}} w_{\rm H}(\underline{x}).$$

In der Literatur wird anstelle von $d_{\rm F}$ teilweise auch d_∞ verwendet.

- Wesentlicher Unterschied zur minimalen Distanz ist, dass bei Faltungscodes nicht Informations– und Codeworte zu betrachten sind, sondern Sequenzen mit der Eigenschaft **semi–infinite.**
- Jede Codesequenz $\underline{x}$ beschreibt einen Pfad durch das Trellis. Die freie Distanz ist dabei das kleinstmögliche Hamming–Gewicht eines solchen Pfades (mit Ausnahme des Nullpfades).



Die Grafik zeigt drei der unendlich vielen Pfade mit dem minimalen Hamming–Gewicht $w_{\rm H}(\underline{x}) = d_{\rm F} = 5$.

Pfadgewichtsfunktion (1)

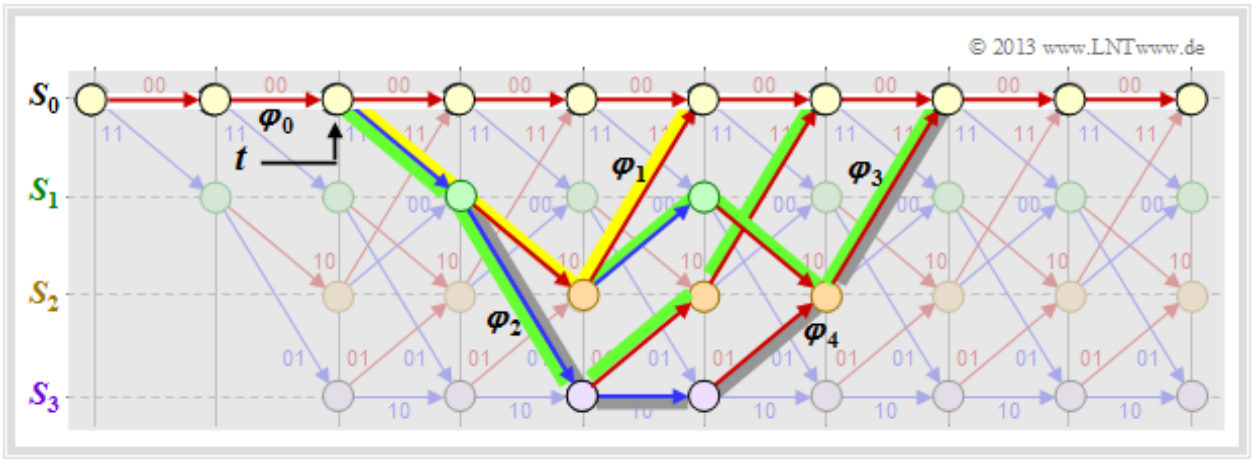
Für jeden linearen Blockcode lässt sich wegen der endlichen Anzahl an Codeworten $\underline{x}$ in einfacher Weise eine Gewichtsfunktion angeben. Für das Beispiel auf der **letzten Seite** lautet diese:

$$W(X) = 1 + 7 \cdot X^3 + 7 \cdot X^4 + X^7 .$$

Bei einem (nicht terminierten) Faltungscode kann keine solche Gewichtsfunktion angegegeben werden, da es unendlich viele, unendlich lange Codesequenzen $\underline{x}$ gibt, und damit auch unendlich viele Trellispfade. Um dieses Problem in den Griff zu bekommen, gehen wir nun von folgenden Voraussetzungen aus:

- Als Bezugsgröße für das Trellisdiagramm wählen wir stets den Pfad der Codesequenz $\underline{x} = \underline{0}$ und nennen diesen den *Nullpfad* φ_0.
- Desweiteren betrachten wir nur noch solche Pfade $\varphi_j \in \Phi$, die alle zu einer vorgegebenen Zeit t vom Nullpfad abweichen und irgendwann wieder zu diesem zurückkehren.

Obwohl nur ein Bruchteil aller Trellispfade zu dieser Menge Φ gehören, beinhaltet $\Phi = \{\varphi_1, \varphi_2, \varphi_3, ...\}$ noch immer eine unbegrenzte Menge an Pfaden. φ_0 gehört nicht zu dieser Menge.



Im obigen Trellis sind einige Pfade $\varphi_j \in \Phi$ eingezeichnet:

- Der gelbe Pfad φ_1 gehört zur Sequenz $\underline{x}_1 = (11, 10, 11)$ mit dem Hamming–Gewicht $w_{\rm H}(\underline{x}_1) = 5$. Damit ist auch das Pfadgewicht $w(\varphi_1) = 5$. Aufgrund der Festlegung des Abzweigzeitpunktes t hat nur noch dieser einzige Pfad φ_1 die freie Distanz $d_{\rm F} = 5$ zum Nullpfad $\Rightarrow\ A_5 = 1$.
- Für die beiden grünen Pfade mit den korrespondierenden Sequenzen $\underline{x}_2 = (11, 01, 01, 11)$ bzw. $\underline{x}_3 = (11, 10, 00, 10, 11)$ gilt $w(\varphi_2) = w(\varphi_3) = 6$. Kein anderer Pfad weist das Pfadgewicht 6 auf. Wir berücksichtigen diese Tatsache durch den Koeffizienten $A_6 = 2$.
- Eingezeichnet ist auch der graue Pfad φ_4, assoziiert mit der Sequenz $\underline{x}_4 = (11, 01, 10, 01, 11)\ \Rightarrow$ $w(\varphi_4) = 7$. Auch die Sequenzen $\underline{x}_5 = (11, 01, 01, 00, 10, 11)$, $\underline{x}_6 = (11, 10, 00, 01, 01, 11)$ und $\underline{x}_7 = (11, 10, 00, 10, 00, 10, 11)$ weisen jeweils das gleiche Pfadgewicht 7 auf $\Rightarrow\ A_7 = 4$.

Damit lautet die Pfadgewichtsfunktion (englisch: *Path Weight Enumerator Function*, PWEF):

$$T(X) = A_5 \cdot X^5 + A_6 \cdot X^6 + A_7 \cdot X^7 + \ ... \ = X^5 + 2 \cdot X^6 + 4 \cdot X^7 + \ ... \ .$$

Die Definition dieser Funktion $T(X)$ wird auf der nächsten Seite nachgeliefert.

Pfadgewichtsfunktion (2)

Definition: Für die **Pfadgewichtsfunktion** (englisch: *Path Weight Enumerator Function*, PWEF) eines Faltungscodes gilt:

$$T(X) = \sum_{\varphi_j \in \Phi} X^{w(\varphi_j)} = \sum_{w=d_{\rm F}}^{\infty} A_w \cdot X^w.$$

- Φ bezeichnet die Menge aller Pfade an, die den Nullpfad φ_0 genau zum festgelegten Zeitpunkt t verlassen und (irgendwann) später zu diesem zurückkehren.
- Gemäß der zweiten Gleichung sind die Summanden nach ihren Pfadgewichten w geordnet, wobei A_w die Anzahl der Pfade mit Pfadgewicht w bezeichnet. Die Summe beginnt mit $w = d_{\rm F}$.
- Das Pfadgewicht $w(\varphi_j)$ ist gleich dem Hamming–Gewicht (also der Anzahl der Einsen) der zum Pfad φ_j assoziierten Codesequenz $\underline{x}_j$:

$$w(\varphi_j) = w_{\rm H}(\underline{x}_j).$$

Hinweis: Die für die linearen Blockcodes definierte Gewichtsfunktion $W(X)$ und die hier definierte Pfadgewichtsfunktion $T(X)$ weisen viele Gemeinsamkeiten auf, sie sind jedoch nicht identisch.

Betrachten wir nochmals die Gewichtsfunktion

$$W(X) = 1 + 7 \cdot X^3 + 7 \cdot X^4 + X^7$$

des (7, 4, 3)–Hamming–Codes und die Pfadgewichtsfunktion

$$T(X) = X^5 + 2 \cdot X^6 + 4 \cdot X^7 + 8 \cdot X^8 + \ldots$$

unseres Standard–Faltungscodierers, so fällt die „1" in der ersten Gleichung auf. Das heißt: Bei den linearen Blockcodes wird das Bezugs–Codewort $\underline{x}_i = \underline{0}$ mitgezählt, wohingegen die Nullcodesequenz $\underline{x}_i = \underline{0}$ bzw. der Nullpfad φ_0 bei den Faltungscodes ausgeschlossen wird. Nach Ansicht der Autoren hätte man auch $W(X)$ ohne die „1" definieren können. Damit wäre unter anderem vermieden worden, dass sich die Bhattacharyya–Schranke für lineare Blockcodes und für Faltungscodes durch „–1" unterscheiden, wie aus den folgenden Gleichungen hervorgeht:

Bhattacharyya–Schranke für die linearen Blockcodes:

$$\Pr(\text{Blockfehler}) \le W(X = \beta) - 1,$$

Bhattacharyya–Schranke für die Faltungscodes:

$$\Pr(\text{Burstfehler}) \le T(X = \beta),$$

Die Pfadgewichtsfunktion $T(X)$ liefert nur Informationen hinsichtlich der Gewichte der Codesequenz $\underline{x}$. Mehr Informationen erhält man, wenn zusätzlich auch die Gewichte der Informationssequenz $\underline{u}$ erfasst werden. Man benötigt dann zwei Formalparameter X und U, wie aus der Definition auf der folgenden Seite hervorgeht.

Erweiterte Pfadgewichtsfunktion

Definition: Die **erweiterte Pfadgewichtsfunktion** (englisch: *Enhanced Path Weight Enumerator Function*, EPWEF) lautet:

$$T_{\rm enh}(X,U) = \sum_{\varphi_j \in \Phi} X^{w(\varphi_j)} \cdot U^{u(\varphi_j)} = \sum_{w} \sum_{u} A_{w,\,u} \cdot X^w \cdot U^u .$$

Es gelten alle Angaben der Definition von $T(X)$ auf der letzten Seite. Zusätzlich ist zu beachten:

- Das Pfadeingangsgewicht $u(\varphi_j)$ ist gleich dem Hamming–Gewicht der zum Pfad φ_j assoziierten Informationssequenz $\underline{u}_j$. Es wird als Potenz des Formalparameters U ausgedrückt.
- Der Koeffizient $A_{w,\,u}$ bezeichnet die Anzahl der Pfade φ_j mit dem Pfadausgangsgewicht $w(\varphi_j)$ und dem Pfadeingangsgewicht $u(\varphi_j)$. Als Laufvariable für den zweiten Anteil wird u verwendet.
- Setzt man in der erweiterten Pfadgewichtsfunktion den Formalparameter $U = 1$, so ergibt sich die ursprüngliche Gewichtsfunktion $T(X)$ gemäß der Definition auf der letzten Seite.

Bei vielen (und allen relevanten) Faltungscodes lässt sich obere Gleichung noch vereinfachen:

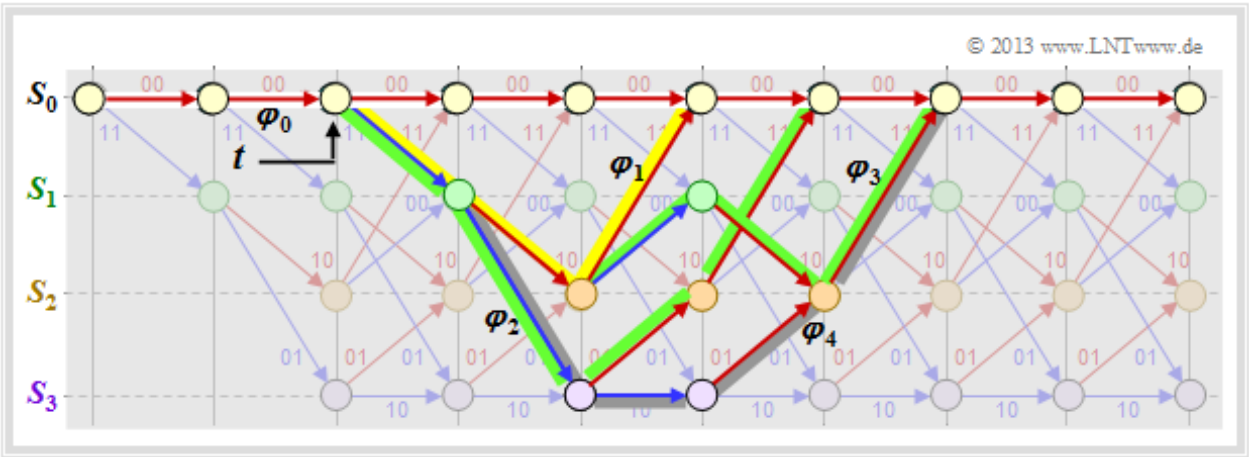
$$T_{\rm enh}(X,U) = \sum_{w=d_{\rm F}}^{\infty} A_w \cdot X^w \cdot U^u .$$

Die erweiterte Pfadgewichtsfunktion unseres Standardcodieres lautet somit:

$$T_{\rm enh}(X,U) = U \cdot X^5 + 2 \cdot U^2 \cdot X^6 + 4 \cdot U^3 \cdot X^7 + \ldots .$$

Vergleicht man dieses Ergebnis mit dem unten dargestellten Trellis, so erkennt man:

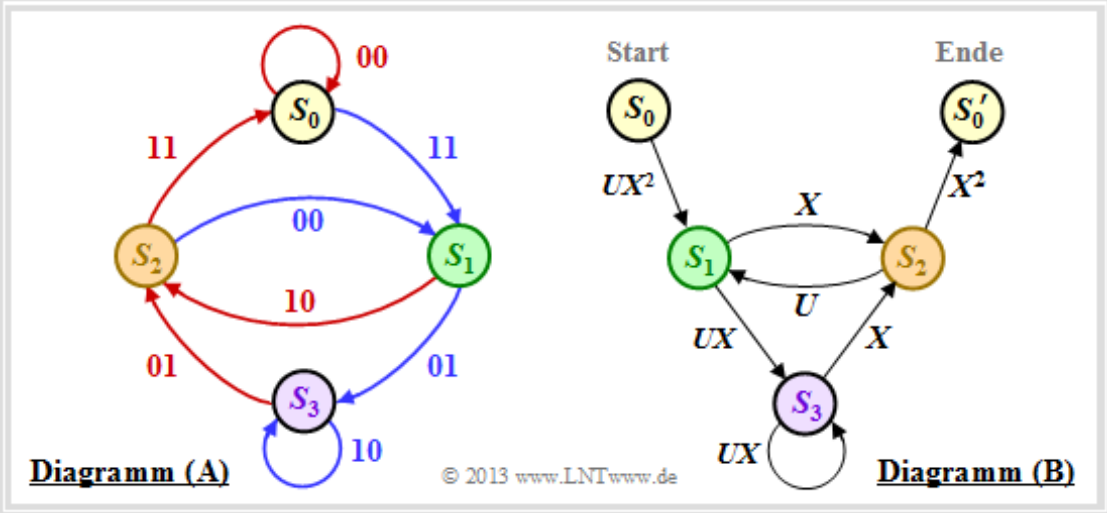
- Der gelb hinterlegte Pfad – gekennzeichnet durch X^5 – setzt sich aus einem blauen Pfeil ($u_i = 1$) und zwei roten Pfeilen ($u_i = 0$) zusammen. Somit wird aus X^5 der erweiterte Term UX^5.
- Die Sequenzen der beiden grünen Pfade sind $\underline{u}_2 = (1, 1, 0, 0)$ ⇒ $\underline{x}_2 = (11, 01, 01, 11)$ sowie $\underline{u}_3 = (1, 0, 1, 0, 0)$ ⇒ $\underline{x}_3 = (11, 10, 00, 10, 11)$. Daraus ergibt sich der zweite Term $2 \cdot U^2X^6$.
- Der graue Pfad (und die drei nicht gezeichneten Pfade) ergeben zusammen den Beitrag $4 \cdot U^3X^7$. Jeder dieser Pfade beinhaltet drei blaue Pfeile ⇒ drei Einsen in jeder Informationssequenz.



Pfadgewichtsfunktion aus Zustandsübergangsdiagramm (1)

Es gibt eine elegante Methode, um die Pfadgewichtsfunktion $T(X)$ und deren Erweiterung direkt aus dem Zustandsübergangsdiagramm zu bestimmen. Dies soll hier und auf den folgenden Seiten am Beispiel unseres **Standardcodes** demonstriert werden.

Zunächst muss dazu das Zustandsübergangsdiagramm umgezeichnet werden. Die Grafik zeigt dieses links in der bisherigen Form als Diagramm (A), während rechts das neue Diagramm (B) angegeben ist.



Man erkennt:

- Der Zustand S_0 wird aufgespalten in den Startzustand S_0 und den Endzustand S_0'. Damit lassen sich alle Pfade des Trellisdiagramms, die im Zustand S_0 beginnen und irgendwann zu diesem zurückkehren, auch im rechten Graphen (B) nachvollziehen. Ausgeschlossen sind dagegen direkte Übergänge von S_0 nach S_0' und damit auch der Nullpfad (Dauer–S_0).
- Im Diagramm (A) sind die Übergänge anhand der Farben Rot (für $u_i = 0$) und Blau (für $u_i = 1$) unterscheidbar, und die Codeworte $\underline{x}_i \in \{00, 01, 10, 11\}$ sind an den Übergängen vermerkt. Im neuen Diagramm (B) werden (00) durch $X^0 = 1$ und (11) durch X^2 ausgedrückt. Die Codeworte (01) und (10) sind nun nicht mehr unterscheidbar, sondern werden einheitlich mit X bezeichnet.
- Anders formuliert: Das Codewort $\underline{x}_i$ wird nun als X^w dargestellt, wobei X eine dem Ausgang (der Codesequenz) zugeordnete Dummy–Variable ist und $w = w_{\rm H}(\underline{x}_i)$ das Hamming–Gewicht des Codewortes $\underline{x}_i$ angibt. Bei einem Rate–1/2–Code ist der Exponent w entweder 0, 1 oder 2.
- Ebenfalls verzichtet wird im Diagramm (B) auf die Farbcodierung. Das Informationsbit $u_i = 1$ wird nun durch $U^1 = U$ und das Informationsbit $u_i = 0$ durch $U^0 = 1$ gekennzeichnet. Die Dummy–Variable U ist also der Eingangssequenz $\underline{u}$ zugeordnet.

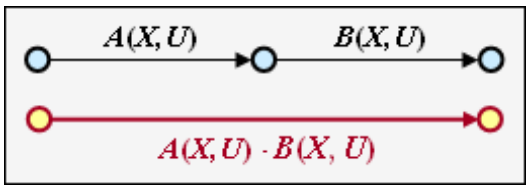
Die Beschreibung wird auf den nächsten Seiten fortgesetzt.

Pfadgewichtsfunktion aus Zustandsübergangsdiagramm (2)

Ziel unserer Berechnungen wird es sein, den (beliebig komplizierten) Weg von S_0 nach S_0' durch die erweiterte Pfadgewichtsfunktion $T_{\rm enh}(X, U)$ zu charakterisieren. Dazu benötigen wir Regeln, um den Graphen schrittweise vereinfachen zu können.

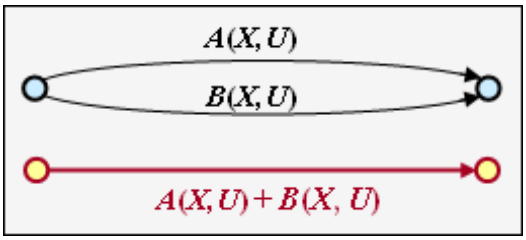
Serielle Übergänge

Zwei serielle Verbindungen – gekennzeichnet durch $A(X, U)$ und $B(X, U)$ – können durch eine einzige Verbindung mit dem Produkt dieser Bewertungen ersetzt werden.



Parallele Übergänge

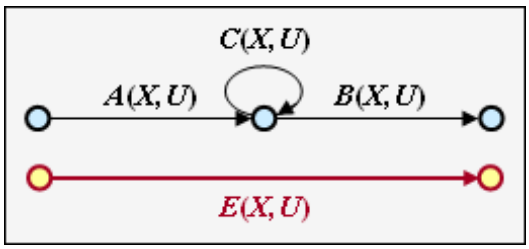
Zwei parallele Verbindungen werden durch die Summe ihrer Bewertungsfunktionen zusammengefasst.



Ring

Die nebenstehende Konstellation kann durch eine einzige Verbindung ersetzt werden, wobei für die Ersetzung gilt:

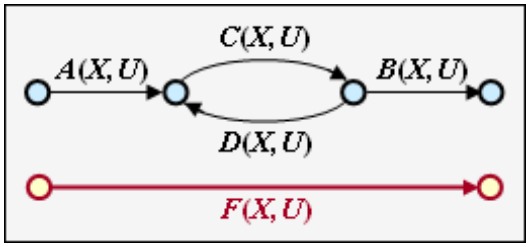
$$E(X,U) = \frac{A(X,U) \cdot B(X,U)}{1 - C(X,U)}.$$



Rückkopplung

Durch die Rückkopplung können sich hier zwei Zustände beliebig oft abwechseln. Für diese Konstellation gilt:

$$F(X,U) = \frac{A(X,U) \cdot B(X,U) \cdot C(X,U)}{1 - C(X,U) \cdot D(X,U)}.$$



Die hier angegebenen Gleichungen für Ring und Rückkopplung sind in **Aufgabe Z3.12** zu beweisen.

Pfadgewichtsfunktion aus Zustandsübergangsdiagramm (3)

Die auf der letzten Seite genannten Regeln sollen nun auf unser Standardbeispiel angewendet werden. In der unteren Grafik sehen Sie links das modifizierte Zustandsübergangsdiagramm (B).

- Zunächst ersetzen wir den rot hinterlegten Umweg von S_1 nach S_2 über S_3 im Diagramm (B) durch die im Diagramm (C) eingezeichnete rote Verbindung. Es handelt sich nach der Klassifizierung auf der letzten Seite um einen „Ring" mit den Beschriftungen $A = C = U \cdot X$ und $B = X$, und wir erhalten die *erste Reduktionsfunktion*:

$$T_1(X, U) = \frac{U \cdot X^2}{1 - U \cdot X}.$$

- Nun fassen wir die parallelen Verbindungen entsprechend der blauen Hinterlegung im Diagramm (C) zusammen und ersetzen diese durch die blaue Verbindung im Diagramm (D). Die *zweite Reduktionsfunktion* lautet somit:

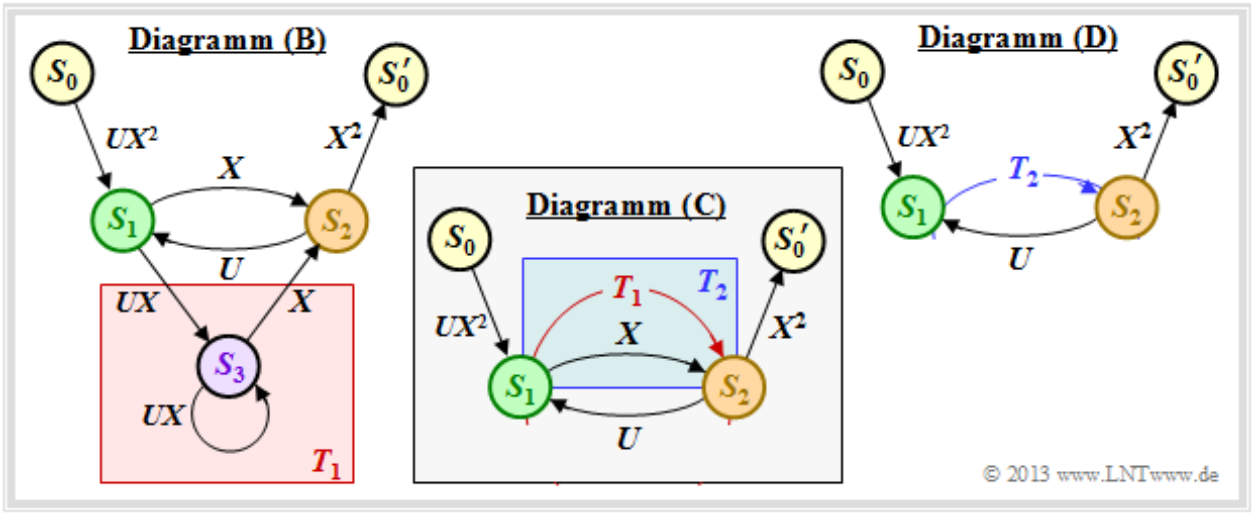
$$T_2(X, U) = T_1(X, U) + X = \frac{UX^2 + X \cdot (1 - UX)}{1 - UX} = \frac{X}{1 - UX}.$$

- Der gesamte Graph (D) kann somit durch eine einzige Verbindung von S_0 nach S_0' ersetzt werden. Nach der Rückkopplungsregel erhält man für die *erweiterte Pfadgewichtsfunktion*:

$$T_{\rm enh}(X, U) = \frac{(UX^2) \cdot X^2 \cdot \frac{X}{1-UX}}{1 - U \cdot \frac{X}{1-UX}} = \frac{UX^5}{1 - UX - UX} = \frac{UX^5}{1 - 2 \cdot UX}.$$

Mit der Reihenentwicklung $1/(1 - x) = 1 + x + x^2 + x^3 + \ldots$ lässt sich hierfür auch schreiben:

$$T_{\rm enh}(X, U) = UX^5 \cdot \left[1 + 2UX + (2UX)^2 + (2UX)^3 + \ldots\right].$$



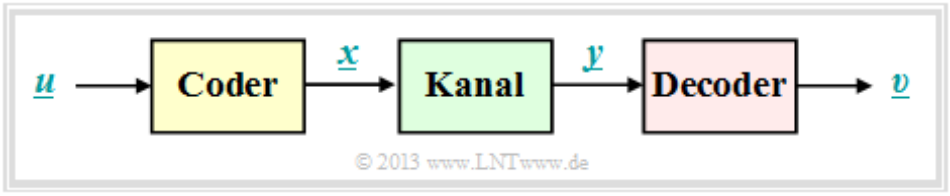
Setzt man die formale Input–Variable $U = 1$, so erhält man die „einfache" Pfadgewichtsfunktion, die allein Aussagen über die Gewichtsverteilung der Ausgangssequenz $\underline{x}$ erlaubt:

$$T(X) = X^5 \cdot \left[1 + 2X + 4X^2 + 8X^3 + \ldots\right].$$

Das gleiche Ergebnis haben wir bereits aus dem Trellisdiagramm auf **Seite 2a** abgelesen. Dort gab es einen grauen Pfad mit Gewicht 5, zwei gelbe Pfade mit Gewicht 6 und vier grüne Pfade mit Gewicht 7.

Burstfehlerwahrscheinlichkeit und Bhattacharyya–Schranke (1)

Das folgende einfache Modell gilt sowohl für lineare Blockcodes als auch für Faltungscodes.



Bei den Blockcodes bezeichnen $\underline{u} = (u_1, ..., u_i, ..., u_k)$ und $\underline{v} = (v_1, ..., v_i, ..., v_k)$ die Informationsblöcke am Eingang und Ausgang des Systems. Damit können folgende Beschreibungsgrößen definiert werden:

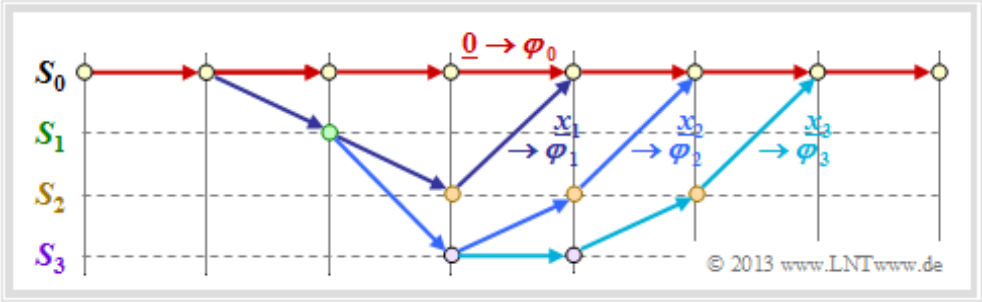
- die **Blockfehlerwahrscheinlichkeit** $\Pr(\underline{v} \neq \underline{u})$,
- die **Bitfehlerwahrscheinlichkeit** $\Pr(v_i \neq u_i)$.

Bei realen Übertragungssystemen ist aufgrund des thermischen Rauschens die Bitfehlerwahrscheinlichkeit stets größer als 0. Weiter gilt:

$$\Pr(\text{Blockfehler}) > \Pr(\text{Bitfehler}) .$$

Hierfür ein einfacher Erklärungsversuch: Entscheidet der Decoder in jedem Block der Länge k Bit genau ein Bit falsch, so beträgt die Bitfehlerwahrscheinlichkeit = $1/k$ und die Blockfehlerwahrscheinlichkeit ist 1.

Bei Faltungscodes ist dagegen die Blockfehlerwahrscheinlichkeit nicht angebbar, da hier $\underline{u} = (u_1, u_2, ...)$ und $\underline{v} = (v_1, v_2, ...)$ Sequenzen darstellen. Selbst der kleinstmögliche Codeparameter $k = 1$ führt hier zur Sequenzlänge $k' \to \infty$, und die Blockfehlerwahrscheinlichkeit ergäbe sich stets zu 1, selbst wenn die Bitfehlerwahrscheinlichkeit extrem klein (aber $\neq 0$) ist.



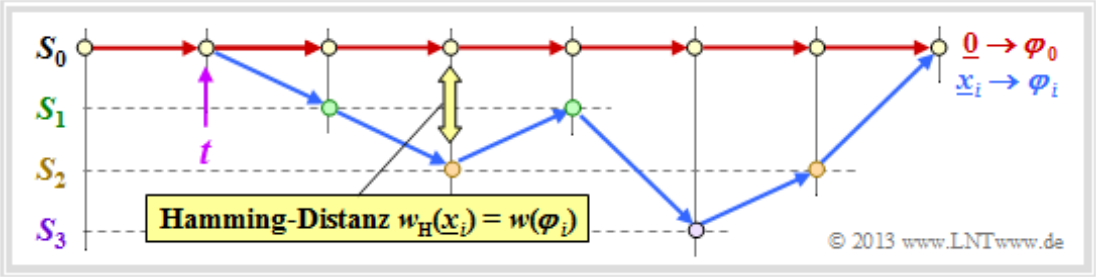
Deshalb definieren wir bei Faltungscodes stattdessen die **Burstfehlerwahrscheinlichkeit**:

$$\Pr(\text{Burstfehler}) = \Pr\{\text{Decoder verlässt zur Zeit } t \text{ den korrekten Pfad}\} .$$

Um für die folgende Herleitung die Schreibweise zu vereinfachen, gehen wir stets von der Nullsequenz ($\underline{0}$) aus, die im gezeichneten Trellis als Nullpfad φ_0 rot dargestellt ist. Alle anderen eingezeichneten Pfade φ_1, φ_2, φ_3, ... (und noch viele mehr) verlassen φ_0 zur Zeit t. Sie alle gehören zur Pfadmenge Φ $\Rightarrow$ „Viterbi–Decoder verlässt den korrekten Pfad zur Zeit t", deren Wahrscheinlichkeit auf der nächsten Seite berechnet werden soll.

Burstfehlerwahrscheinlichkeit und Bhattacharyya–Schranke (2)

Wir gehen wie in **Kapitel 1.6** von der paarweisen Fehlerwahrscheinlichkeit $\Pr[\varphi_0 \to \varphi_i]$ aus, dass vom Decoder anstelle des Pfades φ_0 der Pfad φ_i ausgewählt werden **könnte**. Alle betrachteten Pfade φ_i haben gemein, dass sie den Nullpfad φ_0 zum Zeitpunkt t verlassen; sie gehören alle zur Pfadmenge Φ.



Die gesuchte **Burstfehlerwahrscheinlichkeit** ist gleich der folgenden Vereinigungsmenge:

$$\Pr(\text{Burstfehler}) = \Pr\left([\varphi_0 \mapsto \varphi_1] \cup [\varphi_0 \mapsto \varphi_2] \cup \ldots\right) = \Pr\left(\cup_{\varphi_i \in \Phi}\ [\varphi_0 \mapsto \varphi_i]\right).$$

Eine obere Schranke hierfür bietet die so genannte **Union–Bound** entsprechend Kapitel 1.6:

$$\Pr(\text{Burstfehler}) \le \sum_{\varphi_i \in \Phi}\ \Pr\left[\varphi_0 \mapsto \varphi_i\right] = \Pr(\text{Union Bound}).$$

Die paarweise Fehlerwahrscheinlichkeit kann mit der **Bhattacharyya–Schranke** abgeschätzt werden:

$$\Pr\left[\underline{0} \mapsto \underline{x}_i\right] \le \beta^{w_{\rm H}(x_i)} \quad \Rightarrow \quad \Pr\left[\varphi_0 \mapsto \varphi_i\right] \le \beta^{w(\varphi_i)}.$$

$w_{\rm H}(\underline{x}_i)$ bezeichnet das Hamming–Gewicht der möglichen Codesequenz $\underline{x}_i$, $w(\varphi_i)$ das Pfadgewicht des entsprechenden Pfades $\varphi_i \in \Phi$ und β den so genannten **Bhattacharyya–Kanalparameter.**

Durch Summation über alle Pfade und einen Vergleich mit der (einfachen) **Pfadgewichtsfunktion** $T(X)$ erhalten wir das Ergebnis:

$$\Pr(\text{Burstfehler}) \le T(X = \beta), \quad \text{mit} \quad T(X) = \sum_{\varphi_i \in \Phi}\ X^{w(\varphi_i)}.$$

Beispiel: Für unseren Standardcodierer $\Rightarrow$ $R = 1/2$, $m = 2$, $\mathbf{G}(D) = (1 + D + D^2,\ 1 + D)$ haben wir folgende Pfadgewichtsfunktion erhalten, siehe **Theorieteil, Seite 2a:**

$$T(X) = X^5 + 2 \cdot X^6 + 4 \cdot X^7 + \ldots = X^5 \cdot (1 + 2 \cdot X + 4 \cdot X^2 + \ldots).$$

Mit der Reihenentwicklung $1/(1 - x) = 1 + x + x^2 + x^3 + \ldots$ kann hierfür auch geschrieben werden:

$$T(X) = \frac{X^5}{1 - 2 \cdot X}.$$

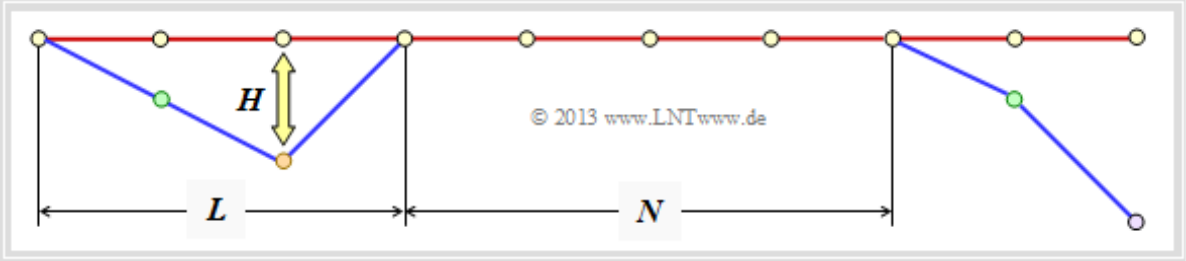
Das BSC–Modell liefert mit der Verfälschungswahrscheinlichkeit ε folgende Bhattacharyya–Schranke:

$$\Pr(\text{Burstfehler}) \le T(X = \beta) = T\left(X = 2 \cdot \sqrt{\varepsilon \cdot (1 - \varepsilon)}\right) = \frac{\left(2 \cdot \sqrt{\varepsilon \cdot (1 - \varepsilon)}\right)^5}{1 - 4 \cdot \sqrt{\varepsilon \cdot (1 - \varepsilon)}}.$$

In **Aufgabe A3.14** soll diese Gleichung numerisch ausgewertet werden.

Bitfehlerwahrscheinlichkeit und Viterbi–Schranke (1)

Abschließend wird eine obere Schranke für die Bitfehlerwahrscheinlichkeit angegeben. Entsprechend der Grafik gehen wir wie **[Liv10]** von folgenden Gegebenheiten aus:



- Gesendet wurde die Nullsequenz $\underline{x} = \underline{0} \Rightarrow$ Pfad φ_0.
- Die Dauer einer Pfadabweichung (englisch: *Error Burst Duration*) wird mit L bezeichnet.
- Den Abstand zweier Bursts (englisch: *Inter–Burst Time*) nennen wir N.
- Das Hamming–Gewicht des Fehlerbündels sei H.

Für einen Rate–1/n–Faltungscode $\Rightarrow k = 1$, also einem Informationsbit pro Takt, lässt sich aus den Erwartungswerten $E[L]$, $E[N]$ und $E[H]$ der oben definierten Zufallsgrößen eine obere Schranke für die Bitfehlerwahrscheinlichkeit angeben:

$$\Pr(\text{Bitfehler}) = \frac{\mathrm{E}[H]}{\mathrm{E}[L] + \mathrm{E}[N]} \le \frac{\mathrm{E}[H]}{\mathrm{E}[N]}.$$

Hierbei ist vorausgesetzt, dass die (mittlere) Dauer eines Fehlerbündels in der Praxis sehr viel kleiner ist als der zu erwartende Abstand zweier Bündel. Weiter kann gezeigt werden, dass die mittlere *Inter–Burst Time* E[N] gleich dem Kehrwert der Burstfehlerwahrscheinlichkeit ist, während der Erwartungswert im Zähler wie folgt abgeschätzt:

$$\mathrm{E}[H] \le \frac{1}{\Pr(\text{Burstfehler})} \cdot \sum_{\varphi_i \in \Phi} u(\varphi_i) \cdot \beta^{w(\varphi_i)}.$$

Bei der Herleitung dieser Schranke in **[Liv10]** werden die *paarweise Fehlerwahrscheinlichkeit* $\Pr[\varphi_0 \to \varphi_i]$ sowie die *Bhattacharyya–Abschätzung* verwendet. Damit erhält man mit

- dem Pfadeingangsgewicht $u(\varphi_i)$,
- dem Pfadausgangsgewicht $w(\varphi_i)$,
- dem Bhattacharyya–Parameter β.

die folgende Abschätzung für die Bitfehlerwahrscheinlichkeit:

$$\Pr(\text{Bitfehler}) \le \sum_{\varphi_i \in \Phi} u(\varphi_i) \cdot \beta^{w(\varphi_i)}.$$

Man nennt diese Abschätzung die **Viterbi–Schranke**.

Bitfehlerwahrscheinlichkeit und Viterbi–Schranke (2)

Wir erinnern uns an die **erweiterte Pfadgewichtsfunktion**

$$T_{\rm enh}(X,U) = \sum_{\varphi_j \in \Phi} X^{w(\varphi_j)} \cdot U^{u(\varphi_j)}.$$

Leitet man diese Funktion nach der Dummy–Eingangsvariablen U ab, so erhält man

$$\frac{\rm d}{{\rm d}U}\ T_{\rm enh}(X,U) = \sum_{\varphi_j \in \Phi} u(\varphi_j) \cdot X^{w(\varphi_j)} \cdot U^{u(\varphi_j)-1}.$$

Schließlich setzen wir noch für die Dummy–Eingangsvariablen $U = 1$:

$$\left[\frac{\rm d}{{\rm d}U}\ T_{\rm enh}(X,U)\right]_{U=1} = \sum_{\varphi_j \in \Phi} u(\varphi_j) \cdot X^{w(\varphi_j)}.$$

Man erkennt den Zusammenhang zum Ergebnis der letzten Seite.

Zusammenfassung: Die **Bitfehlerwahrscheinlichkeit** eines Faltungscodes kann mit der erweiterten Pfadgewichtsfunktion in geschlossener Form abgeschätzt werden:

$${\rm Pr(Bitfehler)} \le {\rm Pr(Viterbi)} = \left[\frac{\rm d}{{\rm d}U}\ T_{\rm enh}(X,U)\right]_{\substack{X=\beta\\U=1}}.$$

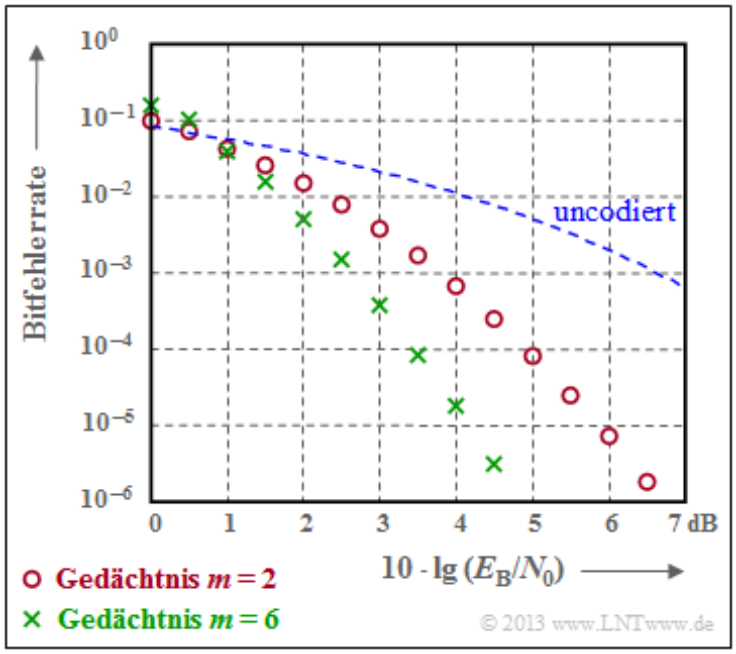
Man spricht von der **Viterbi–Schranke**. Dabei leitet man die erweiterte Pfadgewichtsfunktion nach dem zweiten Parameter U ab und setzt dann $X = \beta$ und $U = 1$.

In **Aufgabe A3.14** werden

- die Viterbi–Schranke und
- die Bhattacharyya–Schranke

für unseren Rate–1/2–Standardcode sowie das **BSC–Modell** numerisch ausgewertet.

- Die roten Kreise kennzeichnen die Bitfehlerrate für den gleichen Code ($m = 2$) beim **AWGN–Kanal.**
- Die grünen Kreuze markieren einen Faltungscode mit $m = 6$, den man oft **Industriestandardcode** nennt.



Die Grafik verdeutlicht die gute Korrekturfähigkeit der Faltungscodes. Insbesondere Codes mit großem Gedächtnis m führen zu großen Gewinnen gegenüber uncodierter Übertragung (gestrichelte Kurve).