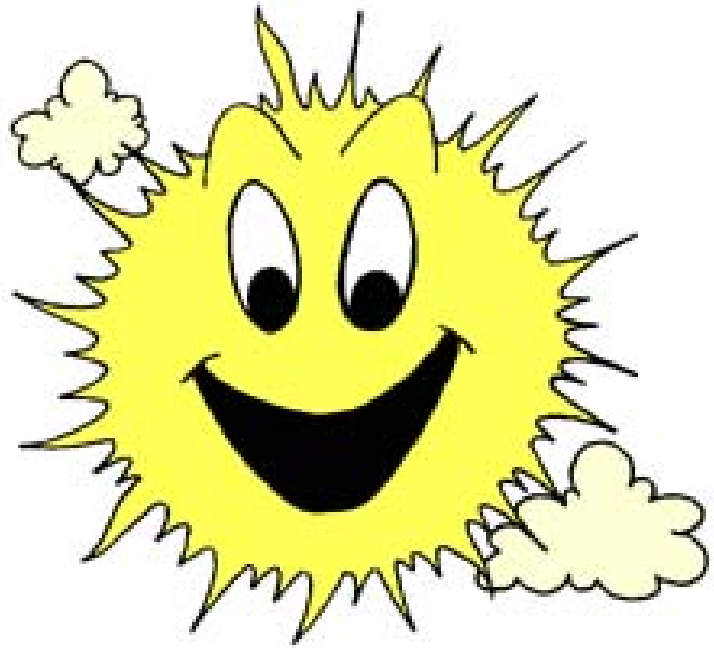
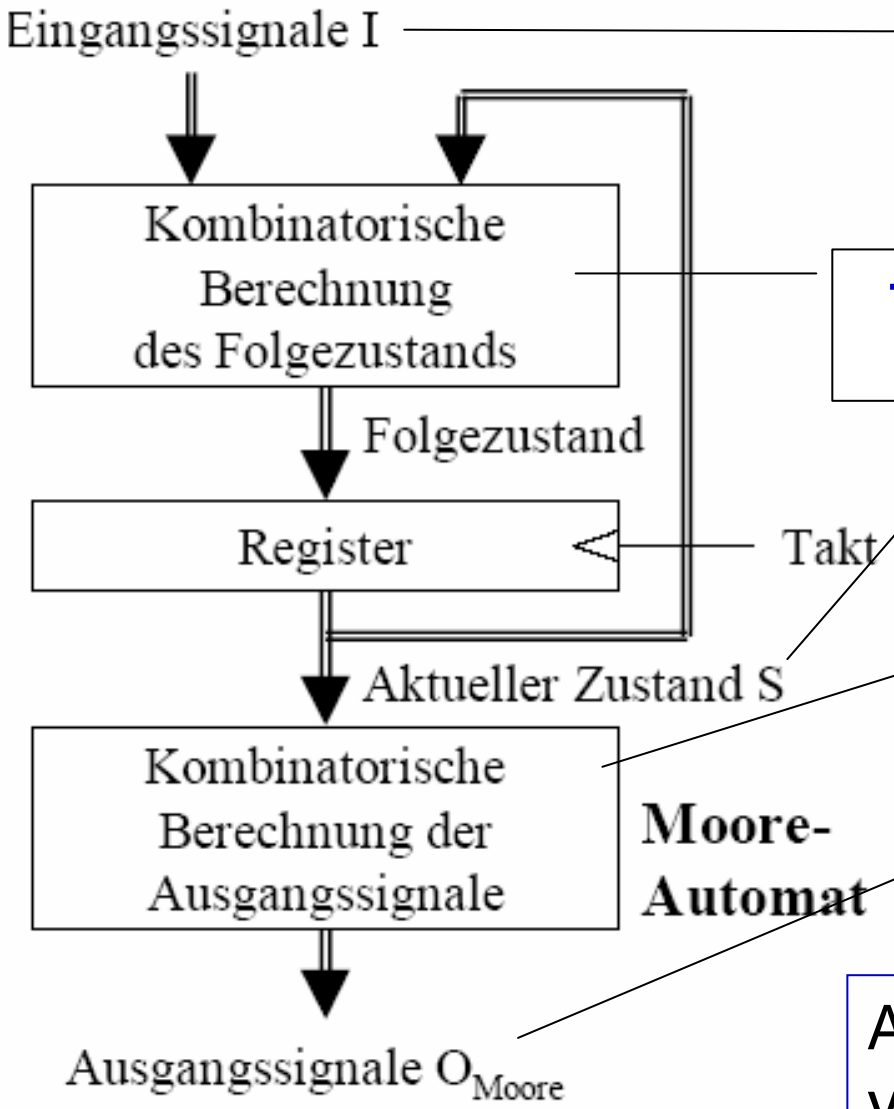
Wiederholung der 9. Vorlesung



19.2 Formale Beschreibung Automaten

- Ein endlicher Automat ist ein Fünftupel
  A = (X, Y, S, f, g) .
- **X** ist ein endliches nichtleeres **Eingabe**alphabet.
- **Y** ist ein endliches nichtleeres **Ausgabe**alphabet.
- **S** ist eine endliche nichtleere Menge von **Zustände**n.
- **f**: **Zustands**(überführungs)**funktion**
- **g**: **Ausgabefunktion**

19.3 Moore-Automaten



X: Eingabealphabet

f: Zustands-/Übergangsfunktion
f(X,S) → S

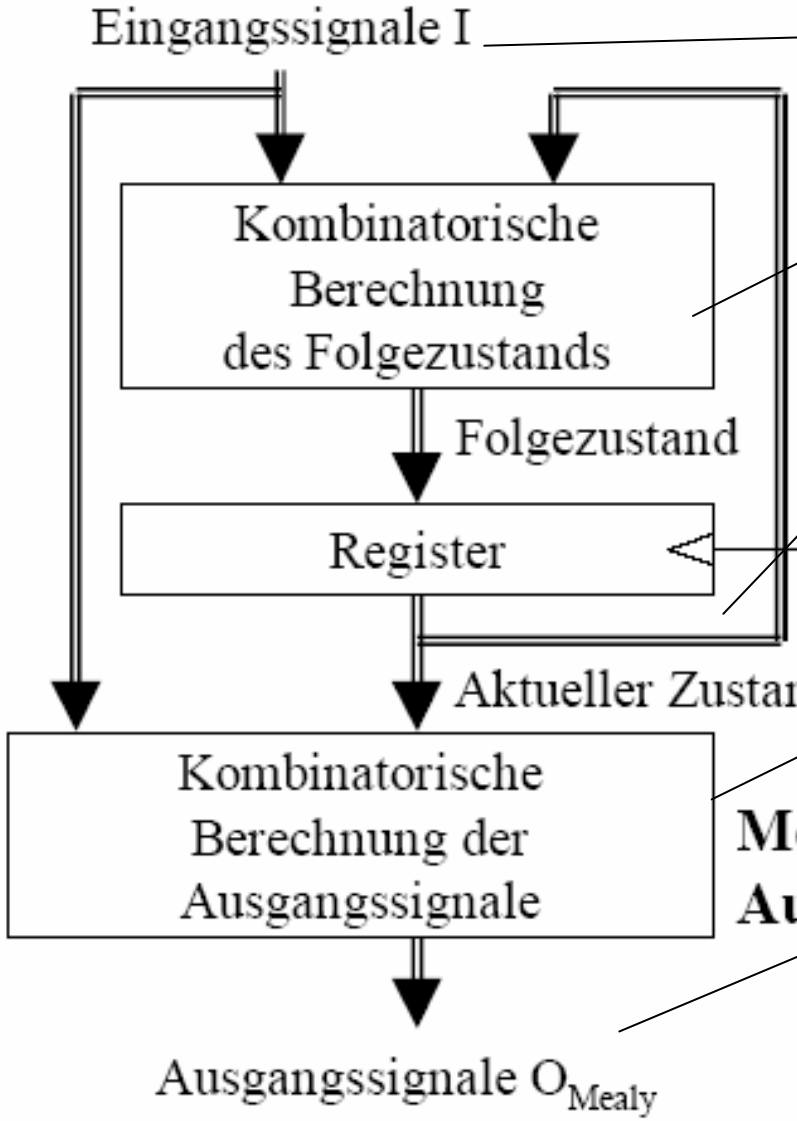
S: Menge von Zuständen

g: Ausgabefunktion
g(S) → Y

Y: Ausgabealphabet

Ausgangssignale sind nur vom Zustand abhängig.

19.4 Mealy-Automaten



X: Eingabealphabet

f: Zustands-/Übergangsfunktion
f(X,S) → S

S: Menge von Zuständen

g: Ausgabefunktion
g(**X**,S) → Y

Y: Ausgabealphabet

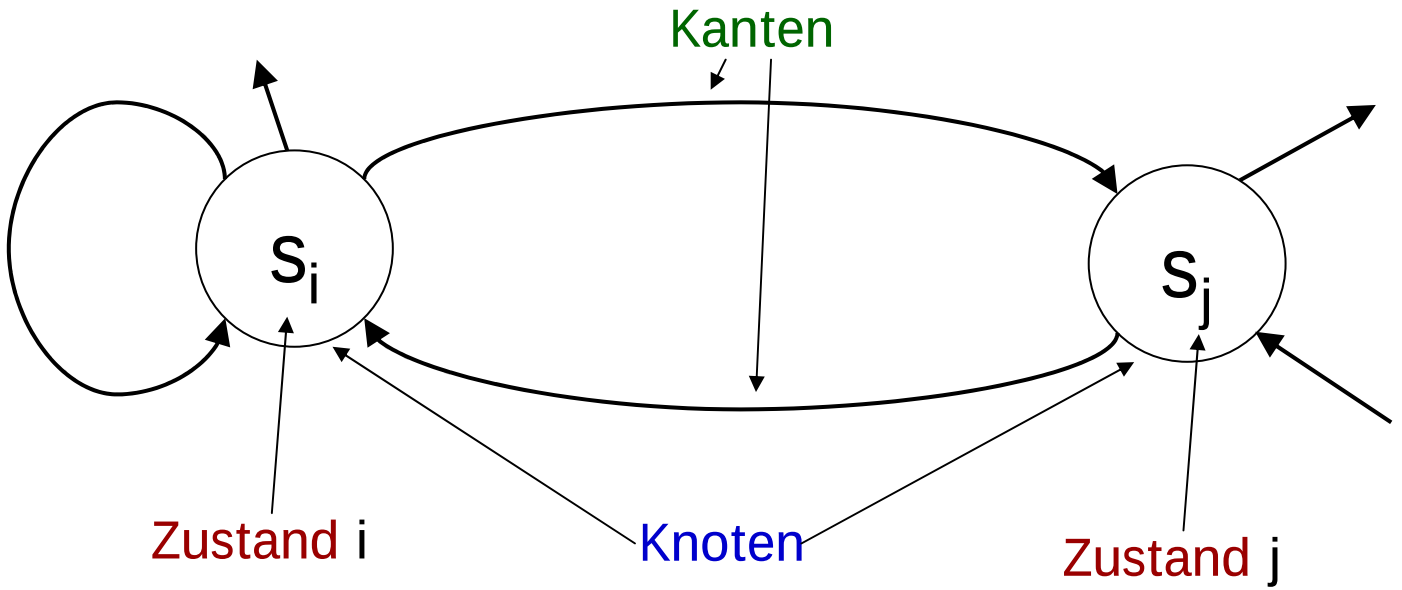
Ausgangssignal wechselt schon bei Änderung des Eingangssignal.

19.6 Beschreibung endlicher Automaten

Endliche Automaten, Schaltwerke können beschrieben werden durch:

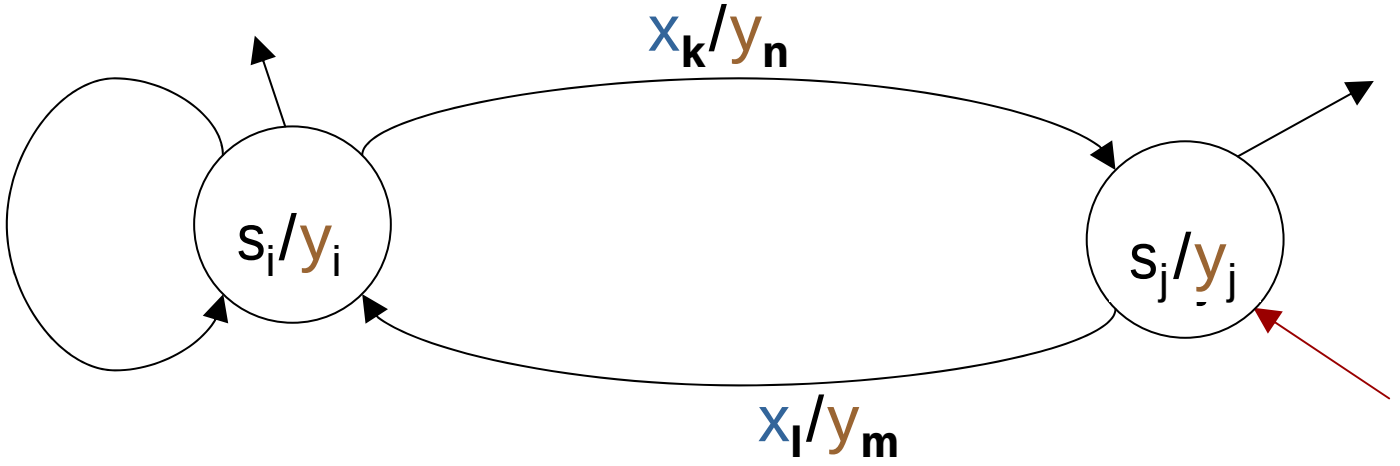
- Zustandsfolgetabellen
- KV-Diagramme
- Schaltfunktionen oder Vektorfunktionen (enthalten die Ausgangs- und Übergangs-Funktionen)
- Zustandsgraphen

19.6 Zustandsgraphen



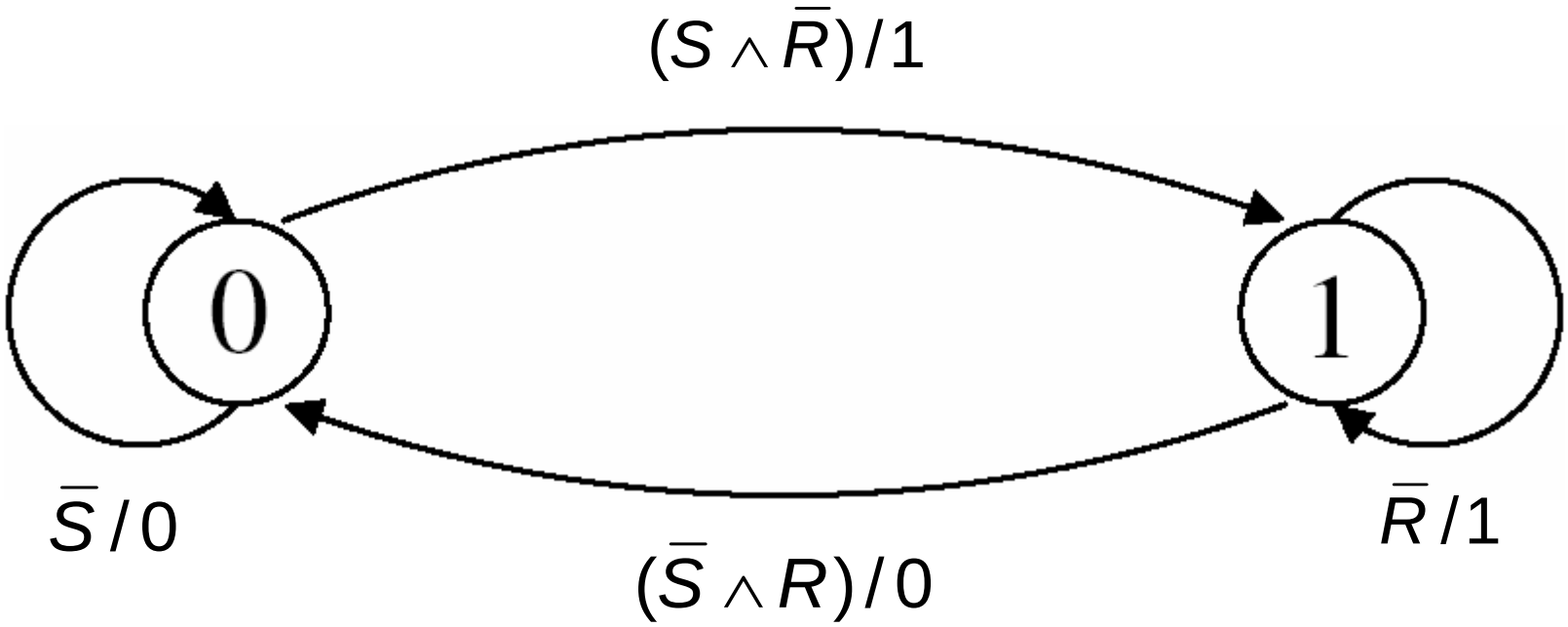
- Ein Zustandsgraph besteht aus Knoten und gerichteten Kanten. Die Knoten (Kreise) beschreiben die Zustände. Die Kanten stellen in Abhängigkeit vom anliegenden Eingabezeichen die Übergänge zwischen dem gegenwärtigen und dem nächsten Zustand her (nächste Folie).
- Zustandsgraph und Automaten sind äquivalent.

19.6 Zustandsgraphen

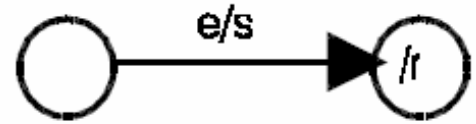


- Pfeil auf ersten Zustand ist Anfangszustand
- Vor dem '/' steht die Eingangsbedingung, unter der der Ausgangszustand verlassen wird, hinter dem '/' steht das Ausgangssignal
  → Mealy-Automat
- Im Knoten vor dem '/' steht wie bisher der Zustand, dahinter das Ausgangssignal
  → Moore-Automat

19.6 Zustandsgraphen (Beispiel FF)

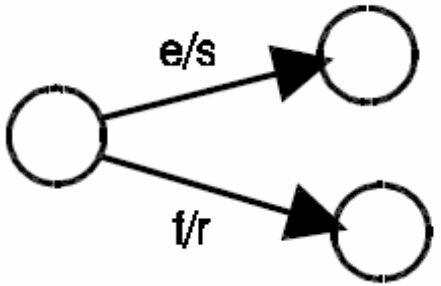


19.6 Zustandsgraphen



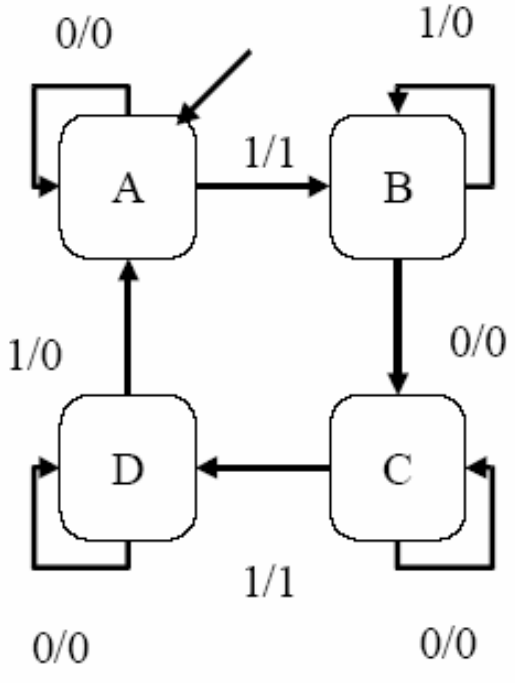
Zustandsübergang

e auslösendes Ereignis
/ s und /r erzeugte Ereignisse



- Statecharts beschreiben endliche Automaten.
- Statecharts sind äquivalent zu endlichen Automaten.
- Ein Zustand im Chart ist elementar oder selbst wieder ein Statechart. Statecharts sind also hierarchisch und parallel zerlegbar.
- Das Zusammenfassen von Zuständen erzeugt Superstates

19.7 Beispiel Mealy-Automat

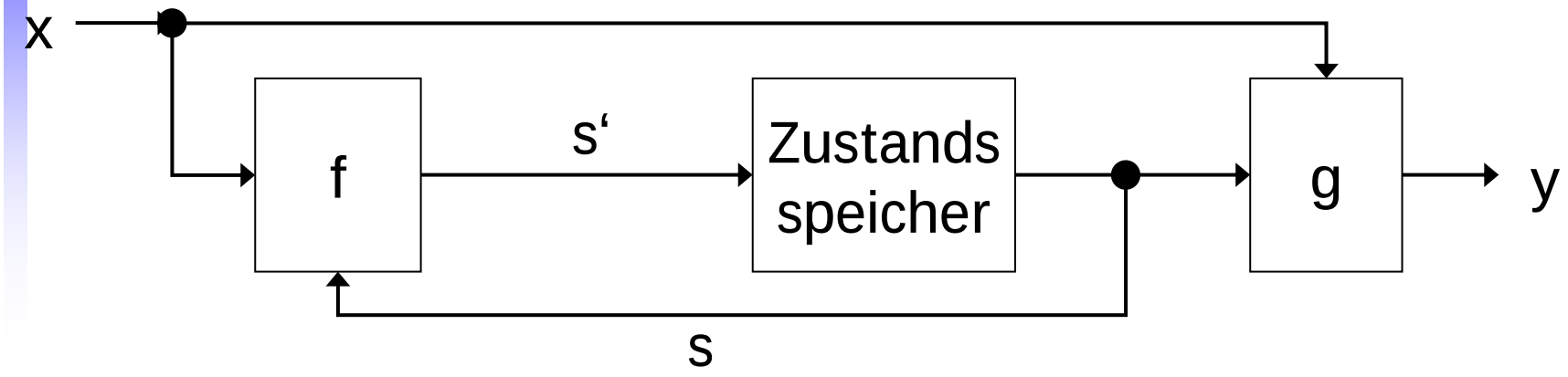


- Beschreiben Sie den Automaten mit einem 5-Tupel

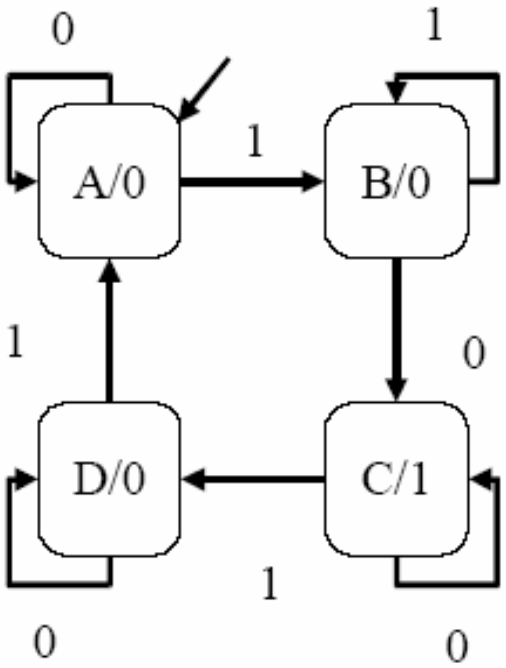
A = ({0,1}, {0,1}, {A,B,C,D}, f, g);
mit f: f(X,S) ; g: g(X,S)

- Skizzieren Sie den Automaten

ergänzen Sie ...



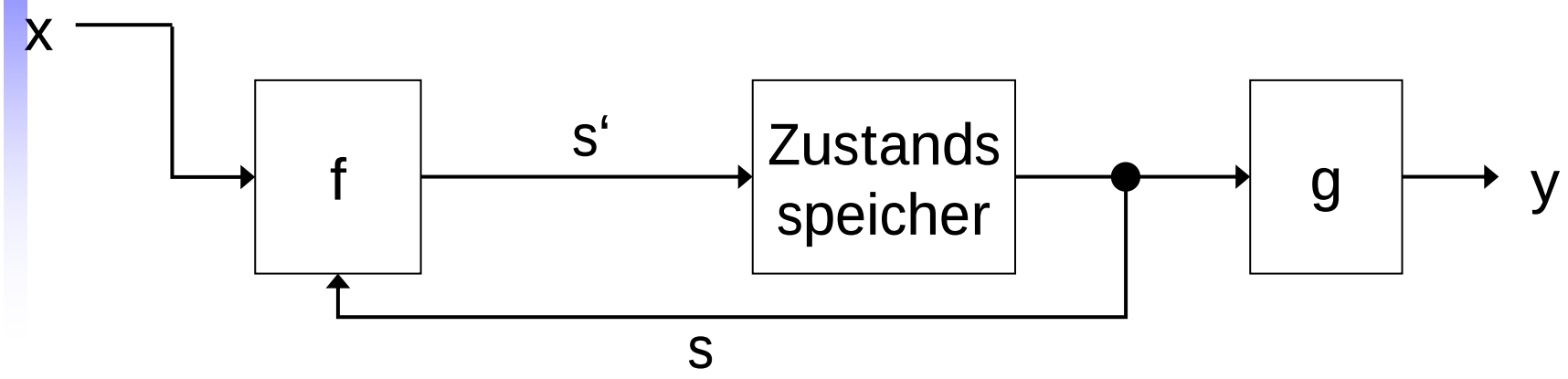
19.7 Beispiel Moore-Automat



- Beschreiben Sie den Automaten mit einem 5-Tupel

A = ({0,1}, {0,1}, {A,B,C,D}, f, g);
mit f: f(X,S) ; g: g(S)

- Skizzieren Sie den Automaten



19.8 Zustandsfolgetabelle

Eingangsvariable		Ausgangsvariable	
$s_0, \ldots, s_n$	$x_0, \ldots, x_i$	$s'_0, \ldots, s'_n$	$y_0, \ldots, y_m$

Die Zustandsfolgetabelle enthält:

- die Komponenten des Eingangsvektor X und die Komponenten des Zustandsvektors $S(t_n)$
- Als Ausgangsvariable die Komponenten des Ausgangsvektors Y und die Komponenten des Folgezustandsvektors $S(t_{n+1})$

10.1 Wiederh.: Schaltnetze und Schaltwerke

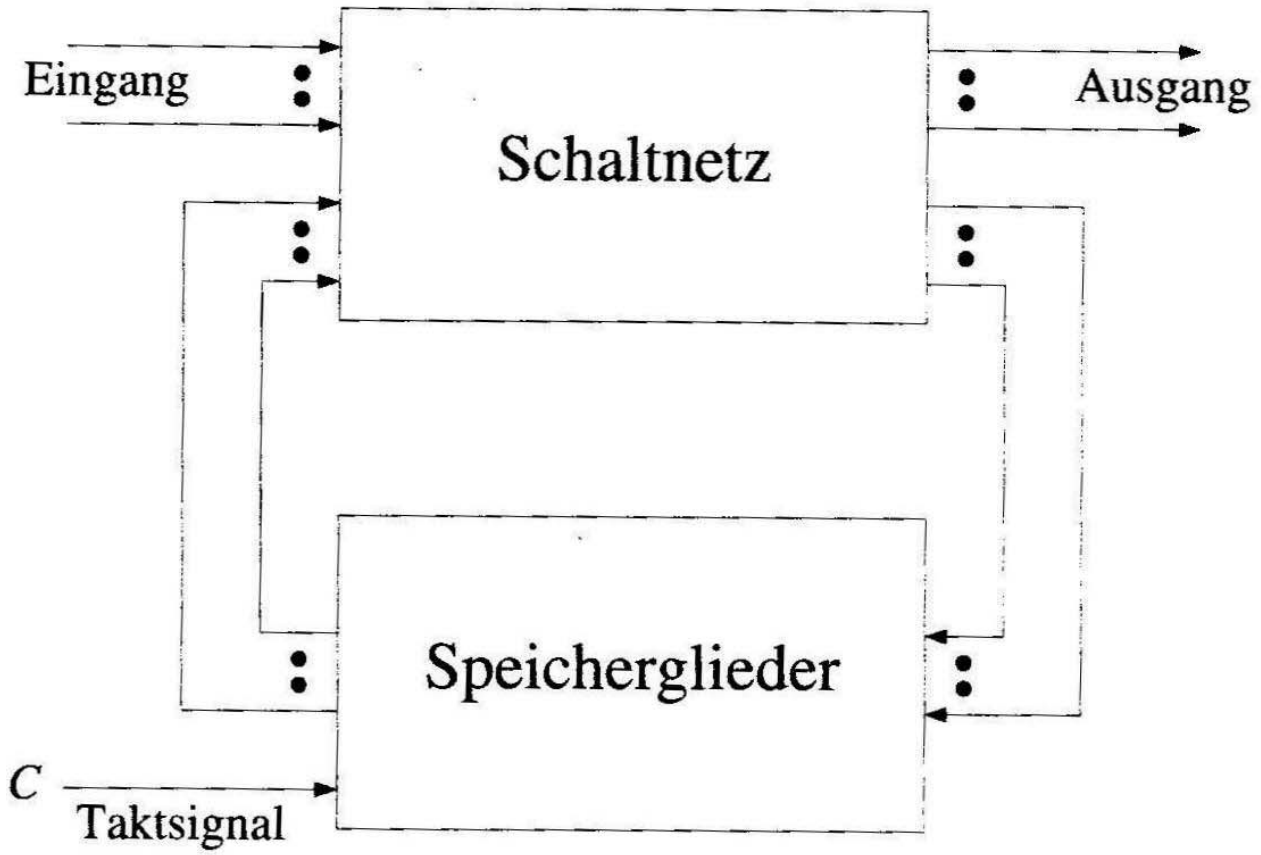
- Bei einem Schaltnetz hängt der Ausgangszustand nur vom aktuellen Eingangszustand ab:

$$A = f(E)$$

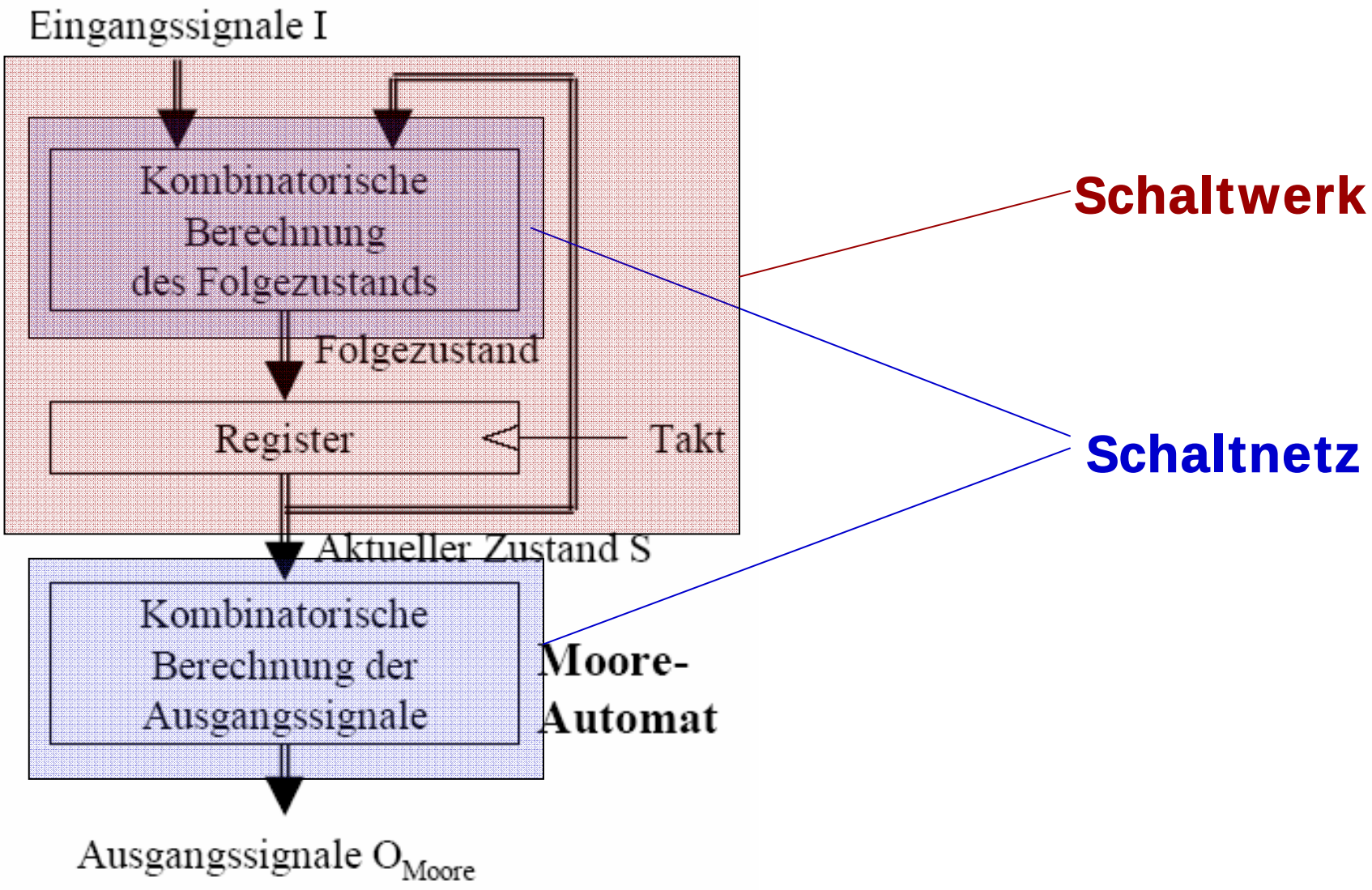
- Bei einem Schaltwerk kann der Ausgangszustand von allen bisherigen Eingangszuständen abhängen:

$$A = f(\ E(t_n),\ E(t_{n-1}), \ldots E(t_0)\ )$$

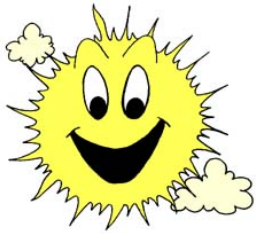
Wiederholung Schaltnetze - Schaltwerke



Zum Praktikum 6



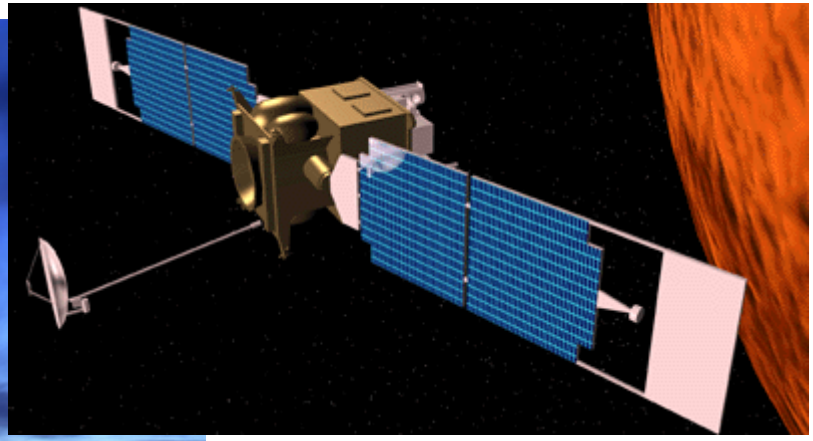
Ende der Wiederholung



17. Codierverfahren und Codesicherung

- Einführung
- Stellendistanz und Hammingdistanz
- Huffman-Codierung
- Fehlererkennung und Korrektur

17.1. Warum Kodierungstheorie?



Anforderung an Kodierung:

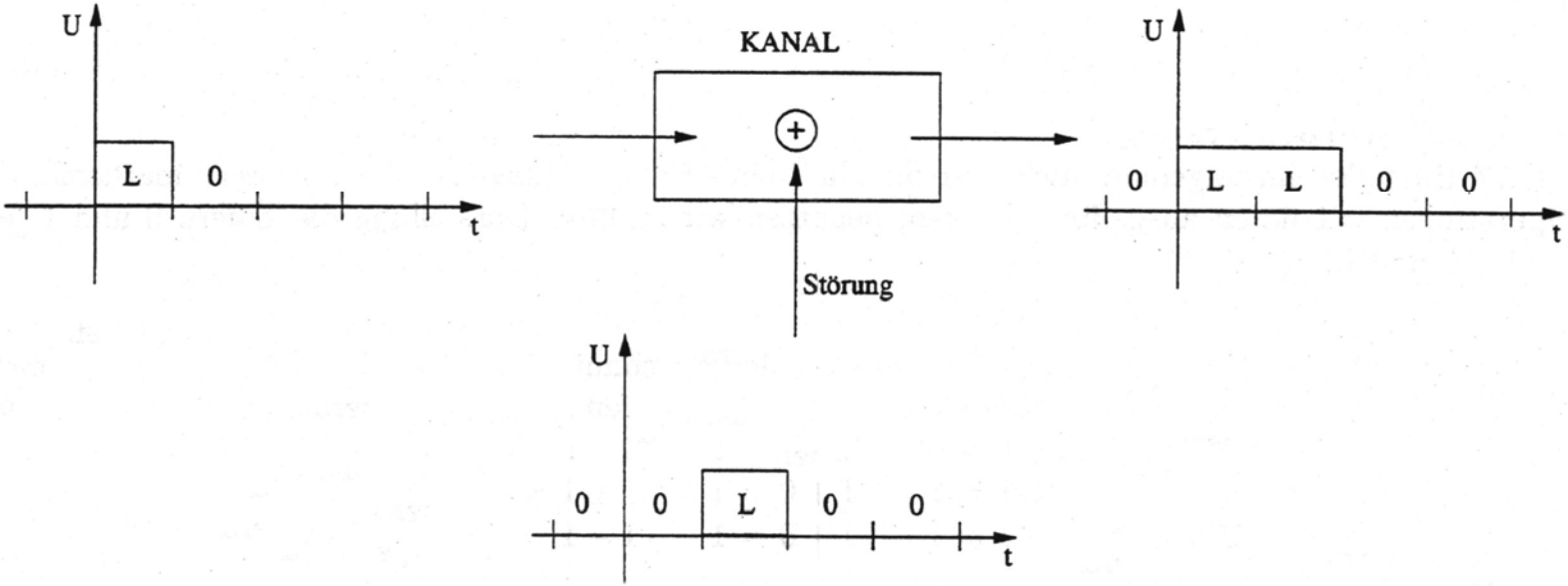
- Möglichst sparsame Darstellung zur Speicherung ohne Verlust
- Sichere Übertragung von Informationen auf gestörten Kanälen

➔ Unabdingbares Instrument in der modernen Kommunikationstechnik

17.1 Begriffe

- Code:
  - Die darzustellende Information in einer bestimmten Repräsentation
  - Zeichenfolge
- Kodierung:
  - Übergang von einem Repräsentationssystem (mit gewissen Eigenschaften) in ein anderes (mit anderen Eigenschaften)

17.1. Kanal mit Fehlereinstreuung



Der Empfänger kann „echte“ Information und eingestreuten Fehler nicht unterscheiden

Ziel:

- möglichst geringe Redundanz
- möglicht sichere Übertragung und Speicherung

Informationsgehalt, Entropie, Redundanz

- **Entscheidungsgehalt H_0** einer Nachricht:
  $H_0(n) = ld(n)$; Maßeinheit ist das bit (binary digit)
- Wenn Informationsquelle n Nachrichten liefern kann, dann sind mindestens ld(n) Fragen nötig um die Information einer Nachricht zu ermitteln.
- Den **mittlere Informationsgehalt** einer Informationsquelle nennt man **Entropie H**

$$H = \sum_i I(N_i)\, p_i$$

$$= -\sum_i p_i\, ld(p_i)\ ;\ i = 1, 2, ..., n$$

wenn ferner gilt $\sum_i p_i = p_1 + p_2 + ... + p_n = 1$

- **Redundanz**: **R = |H(n) - H0(n)|**

17.2 Codesicherung

- Störung: Bei der Übertragung oder Speicherung kippen einzelne Bits.
- Wenn alle Codeworte als Nutzworte verwendet werden, führt jede Störung zu einem neuen Nutzwort. Der Code ist nicht redundant.
- Es werden Redundanzen eingebaut, die eine Erkennung und eventuell Korrektur der Bitfehler gestatten.
- Beispielsweise kann die Codierung neben Nutzworten auch Pseudoworte enthalten, deren Empfang eine Störung signalisiert.

17.2 Einfache Verfahren

- Sender überträgt die Nachricht zweimal. Empfänger kann beide Sendungen vergleichen.
- Sender wartet bis der Empfänger die gesendete Nachricht zurück sendet. Der Sender kann die gesendete und die empfangene Nachricht vergleichen (Handshake-Verfahren).
- Beide Verfahren haben den Nachteil, dass die Datenmenge verdoppelt wird und keine automatische Korrektur möglich ist.
- Was passiert, wenn nur eine Übertragungsrichtung gestört ist?

17.2 Einfache Fehlerkorrektur

- Jedes einzelne Bit wird dreifach wiederholt.
- Bei jeder empfangenen Triade wird an Hand des majoritären Wertes entschieden.
- Einzelne Bitfehler können so korrigiert werden.

17.2 Beispiel

- Mögliche Fehler bei der Übertragung von **000**:
  - 1 Bit Fehler: 001, 010, 100
  - 2 und 3 Bit fehlerhaft: 110, 101, 011, 111
- Mit $p = 0{,}1$ Wahrscheinlichkeit für einen Bitfehler und q die Wahrscheinlichkeit für ein korrektes Bit:

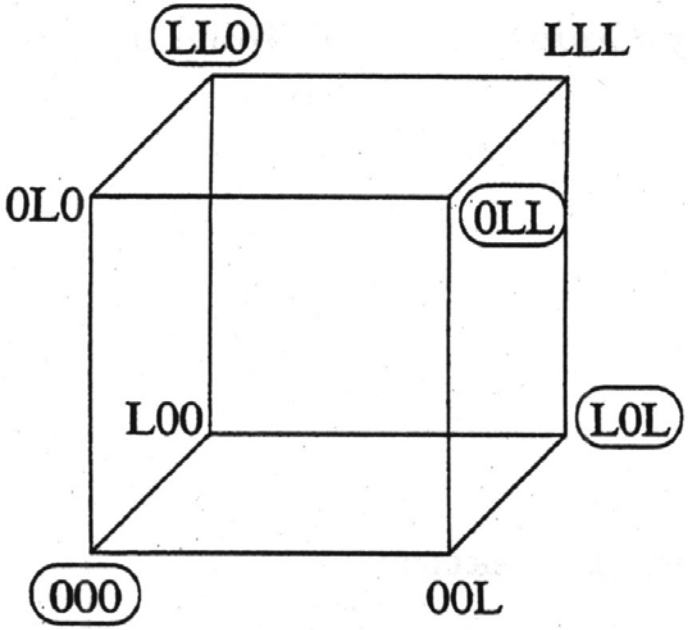
Wahrscheinlichkeit bezüglich 2- und 3 Bit-Fehler

$$\rightarrow P = p^2q + pqp + qp^2 + p^3$$
$$= 3p^2q + p^3$$
$$= 3(p^2(1-p)) + p^3$$
$$= 3p^2 - 3p^3 + p^3$$
$$= 3p^2 - 2p^3$$
$$= 0{,}03 - 0{,}002 = 0{,}028$$

Wahrscheinlichkeit bezüglich 1 Bit-Fehler
→ **Hörsaalübung**

17.3 Möglichkeiten der Fehlererkennung

- Man nutzt *nicht* den gesamten zur Verfügung stehenden Wortraum für gültige Codewörter
- Man lässt **Abstand** zwischen gültigen Codewörtern
- Einzelne Bitfehler erzeugen illegalen Codewörter



In diesem Beispiel:

- Einzelbitfehler werden erkannt
- Stelle des Fehlers nicht erkennbar
- daher keine Korrektur möglich

= wird benutzt

17.4 Hamming-Distanz

Hamming-Distanz **h** oder Hamming-Abstand ist

- kleinste Anzahl der (Bit-)Stellen, in denen sich zwei gültige Wörter voneinander unterscheiden

$$h : \mathbf{2}^n \times \mathbf{2}^n \rightarrow N \text{ mit } h(\langle a_1 \ldots a_n \rangle, \langle b_1 \ldots b_n \rangle)$$

$$= \sum_{i=1}^{n} d_i, \text{ mit } d_i = \begin{cases} 1 & \text{falls } a_i \neq b_i \\ 0 & \text{sonst} \end{cases}$$

- Abstand zwischen zwei bestimmten gültigen Wörtern bezeichnet man als Stellendistanz

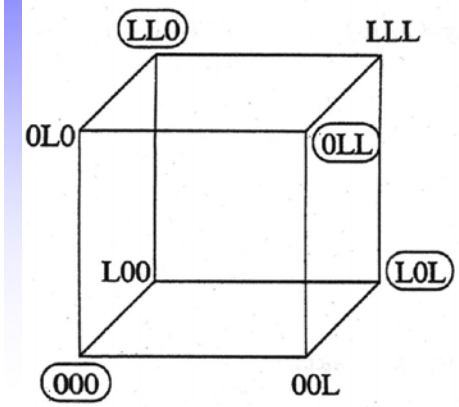
Beispiel:

- h (‘Kanne‘, ‘Tanne‘) = 1
- h (‘11001001‘, ‘01010001‘) = 3

17.4 Hamming-Distanz (andere Definition)

- Entropie: H = ld(N), wobei N die Anzahl der Nutzwörter bezeichnet.
- Mittlere Codierungslänge L
- Redundanz: R = L – H, wobei L die mittlere Länge der Codewörter bezeichnet.

Beispiel:



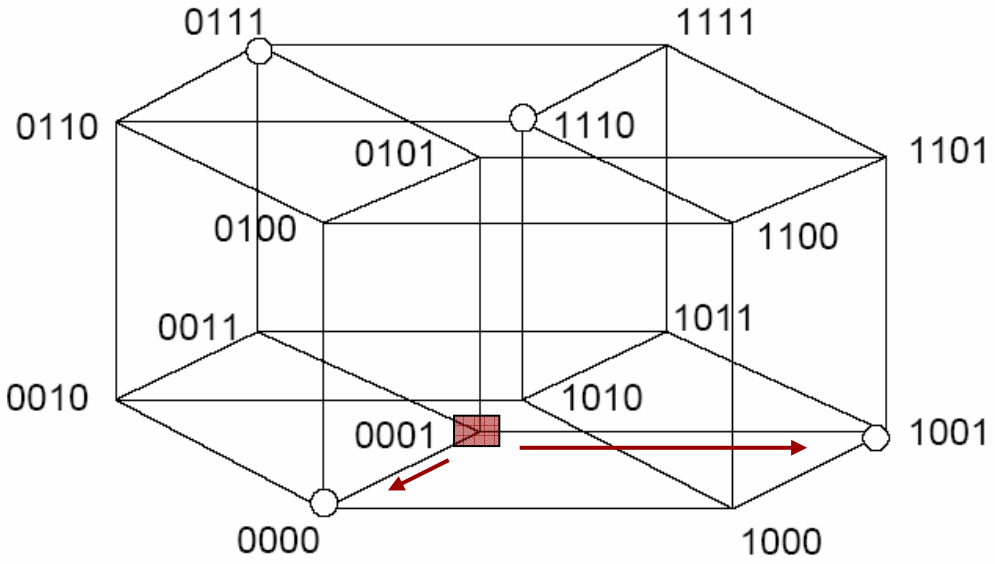
Codierungslänge L = 3

Entropie H = ld(4) = 2

Redundanz R = L – H = 1

Hamming Distanz h = 1

17.4 Hamming-Distanz – Hyperkubus



Kreise sind gültige Worte (4):

Codierungslänge **L** = 4 (bit), Entropie **H** = ld(4) = 2

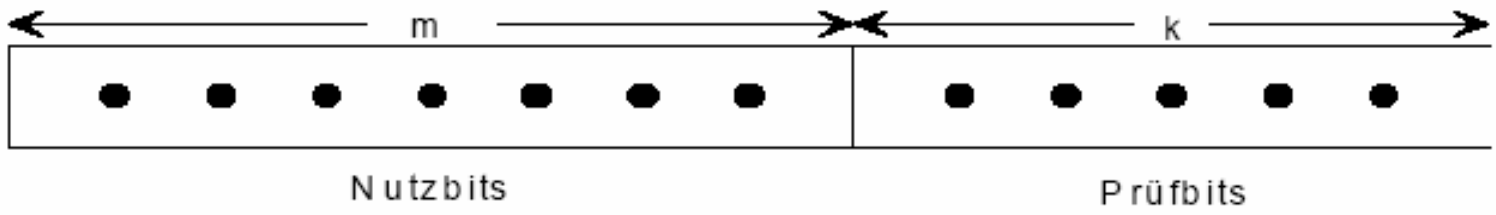
- Redundanz R = L – H = 2
- Eine Hammingdistanz h = 2 kann realisiert werden:
- 1- und 2-Bit-Fehler können erkannt werden.
- Es ist hier nicht ratsam, 1-Bit Fehler zu korrigieren, da dies zu falschen Ergebnissen führen kann.

17.5 Fehlererkennung - Fehlerkorrektur

- Fehlertoleranz basiert auf Redundanz
- Vorrat an möglichen Codewörtern wird nicht vollständig ausgeschöpft:
  - benutzte Codewörter: Nutzwörter
  - nicht benutzte : Pseudowörter
- die redundanten Bits heißen Prüfbits

- Mit s zusätzlichen Bits:
  - Wieviele Fehler können erkannt?
  - Wieviele Fehler können korrigiert werden?

17.5 Fehlererkennung – Fehlerkorrektur (2)



- Code-Redundanz r bezeichnet das Verhältnis **r = k / m**

Es gilt:

- Wenn Codewörter einer Codierung aus *m* Nutzbits und *k* Prüfbits aufgebaut sind, können 1-Bit-Fehler erkannt werden, wenn
  $m + k < 2^k$ gilt.
- Eine Codierung mit einem Hammingabstand *d* erlaubt die Erkennung von *d - 1* Bitfehlern und
  (d-1)/2 Korrekturen, wenn *d* ungerade ist, bzw.
  d/2 – 1 Korrekturen, wenn *d* gerade ist.

ERWEITERN

- Hamming Codes
- Ein- und zweidimensionale Parity Überprüfung

- Siehe Fehlertolerante Codierung (Becker, Drechsel, Molitor – technische Informatik)

17.5 Fehlererkennung – Fehlerkorrektur (3)

- Bedingungen werden beispielsweise durch das Tripletten-Schema erfüllt

m	4	8	11	16	26
k	3	4	4	5	5
r	0,75	0,50	0,36	0,32	0,19

- SEC (single error correction)
- DED (double error detection)
- Beim SEC/DED Code lassen sich Einzel- und Doppelbitfehler erkennen, aber nur Einzelbitfehler sicher korrigieren

17.6 Paritätsbit

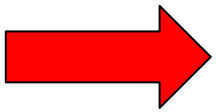
- Die Nutzwörter werden um ein Paritätsbit ergänzt, so dass – in Abhängigkeit vom Verfahren – die Codewörter immer eine gerade oder ungerade Anzahl von 1-Bits enthalten.
  - gerade Parität: Anzahl der 1-bits im erweiterten Codewort ist gerade
  - ungerade Parität: Anzahl der 1-bits im erweiterten Codewort ist ungerade
- Der Einsatz von Paritätsbits wird auch als Zeichen- oder Querparität bezeichnet.
- Eingesetzt wird das Paritätsbit bei der seriellen Übertragung, wobei noch gerade oder ungerade Parität festgesetzt wird.

17.7 Blocksicherung

Daten werden für die Übertragung zu einem Block zusammengefasst.

Beispiel:
ungerade Zahl an 1-Bits für Zeilen und Spalten

Codewort	D3	D2	D1	D0	PB
CW1	0	1	0	1	**1**
CW2	1	1	0	0	**1**
CW3	1	0	0	0	**0**
CW4	1	1	**1**	1	**0**
CW5	0	0	0	0	**1**
Prüfwort	**0**	**0**	**1**	**1**	**0**



Einzelne Bitfehler können detektiert und korrigiert werden.

17.7 Blocksicherung

Codewort	D3	D2	D1	D0	PB
CW1	0	1	0	1	**1**
CW2	1	1	0	0	**1**
CW3	1	0	**1**	0	**0**
CW4	1	1	**1**	1	**0**
CW5	0	0	0	0	**1**
Prüfwort	**0**	**0**	**1**	**1**	**0**

Codewort	D3	D2	D1	D0	PB
CW1	0	1	0	1	**1**
CW2	1	1	0	0	**1**
CW3	1	**1**	0	0	**0**
CW4	1	1	**1**	1	**0**
CW5	0	0	0	0	**1**
Prüfwort	**0**	**0**	**1**	**1**	**0**

Zwei Bitfehler können i.a. nur detektiert und nicht sicher korrigiert wer-den.

17.8 Zyklischer Redundanzcode (CRC)

- Zyklischer Redundanzcode, Polynomcode

- Für jede Nachricht N wir eine „Prüfsumme“ P berechnet und mit N zu N^* zusammengefasst.
- P wird mit einem Generatorpolynom G der Ordnung g berechnet.
- N wird mit g 0-Bits zu N* ergänzt.
- N* wird durch G geteilt.
- Der Rest der Division wird von N* subtrahiert

17.8 Zyklischer Redundanzcode (CRC) (2)

- N^* ist nun durch G ohne Rest teilbar.
- Auf der Empfängerseite kann dies über-prüft werden, da auch hier G bekannt ist.
- Mit der Entfernung von P erhält der Empfänger die ursprüngliche Nachricht N.
- Mit dem CRC-Verfahren können Fehler detektiert aber nicht korrigiert werden.
- Wird das Verfahren mit Paritätsbits er-gänzt kann auch eine Korrektur durch-geführt werden. der Division wird von N* subtrahiert

17.8 Zyklischer Redundanzcode (CRC) (3)

- Berechnung erfolgt ohne Berücksichtigung von Überträgen (algebraische Feldtheorie Modulo-2).
- Berechnung kann einfach in Hardware realisiert werden.
- Generatorpolynome werden so konstruiert, dass bestimmte Fehlerklassen detektiert werden können.

Bezeichnung	Polynom	Anwendung
CRC-12	$x^{12}+x^{11}+x^3+x^2+x^1+1$	6 Bit Werte
CRC-16	$x^{16}+x^{15}+x^2+1$	8 Bit Werte
CRC-CCITT	$x^{16}+x^{12}+x^5+1$	8 Bit Werte
CRC-32	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x^1+1$	Internet

17.9 Huffman-Codierung

- Situation: Die Zeichen eines Zeichenvorrats werden mit unterschiedlicher Wahrscheinlichkeit verwendet.
- Gesucht: Codierung des Zeichenvorrats, die eine minimale mittlere Länge der Codewörter liefert.
- Ziel: Minimierung der zu übertragenden bzw. zu speichernden Datenmenge.

17.9 Huffman-Codierung (2)

- Huffman Baum (H-Baum) liefert eine Lösung für die Minimierung der mittleren Länge der Codierung der Codewörter.
- Der H-Baum ist ein Binärbaum, der iterativ erstellt werden kann.
- Die Blätter des Baums stellen die Code-wörter dar.
- Die Codierung ergibt sich aus der Kantenbeschriftung von der Wurzel zu den Blättern

17.9 mittlere Länge für die Codierung

$$L = \sum_{i=1}^{n} p_i l_i$$

L: mittlere Länge für Codierung eines Zeichens

n: Anzahl der Zeichen des Zeichenvorrats

p_i: Wahrscheinlichkeit für das Auftreten des Zeichens i

l_i: Länge der Codierung für Zeichen i

17.9 Konstruktion des Huffman-Baumes

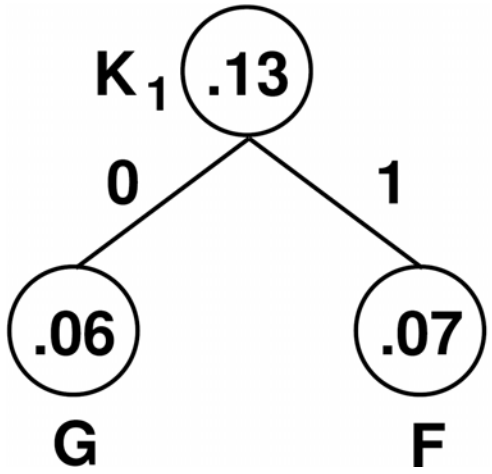
1. Schritt: Suche zwei Zeichen x_i und x_j der Informationsquelle mit kleinsten Wahrscheinlichkeiten
2. Schritt: Bilde einen Knoten K_{ij} des Codebaums; ordne ihm die Wahrscheinlichkeit $p(K_{ij}) = p(x_i) + p(x_j)$ zu. Verbinde K_{ij} mit x_i und x_j.
3. Schritt: Entferne x_i und x_j aus der Informationsquelle und füge ihr statt dessen K_{ij} hinzu.
4. Schritt: Gehe zu Schritt 1, falls die Informationsquelle noch mehr als ein Zeichen enthält.
5. Schritt: Füge (letztes) Zeichen als Wurzel zum Codebaum. Beschrifte die Kanten wie oben erläutert.

17.9 Beispiel H-Baumes

x_i	A	B	C	D	E	F	G
p_i	.25	.21	.18	.14	.09	.07	.06

Auswahl: F (.07), G (.06)

$K_1 = .13$

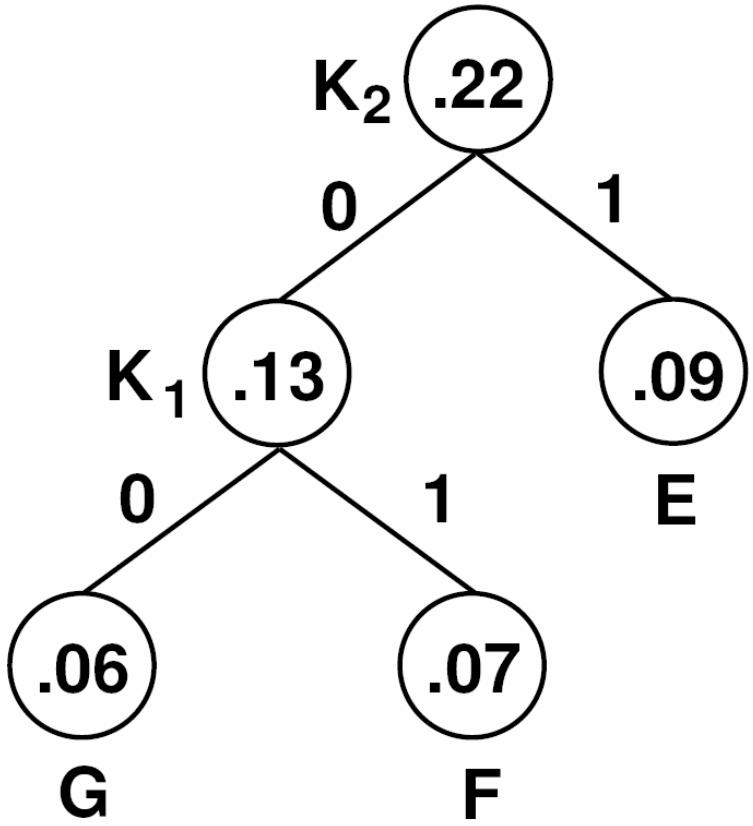


17.9 Beispiel H-Baumes (2)

x_i	A	B	C	D	K_1	E
p_i	.25	.21	.18	.14	.13	.09

Auswahl: E, K_1

K_2 = .22

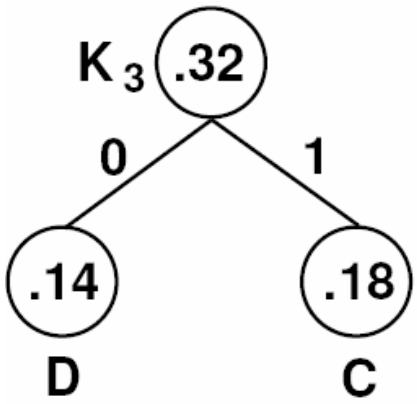


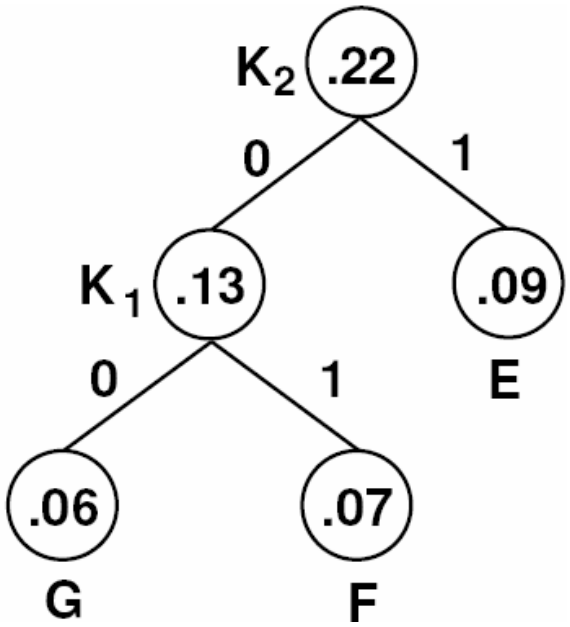
17.9 Beispiel H-Baumes (3)

x_i	A	K_2	B	C	D
p_i	.25	.22	.21	.18	.14

Auswahl: C, D

K_3 = .32



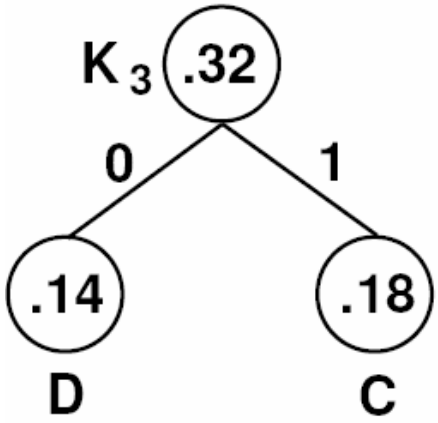


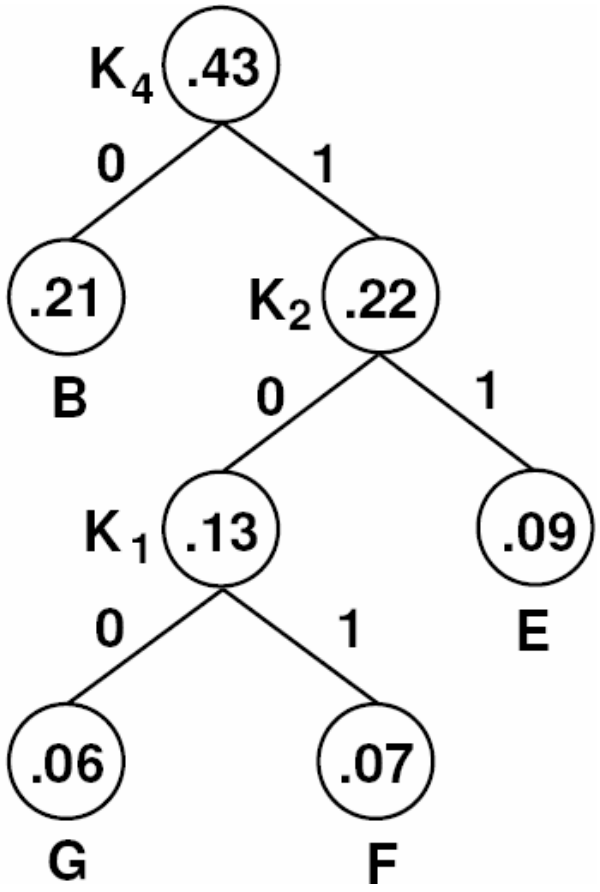
17.9 Beispiel H-Baumes (4)

x_i	K_3	A	K_2	B
p_i	.32	.25	.22	.21

Auswahl: K_2, B

K_4 = .43



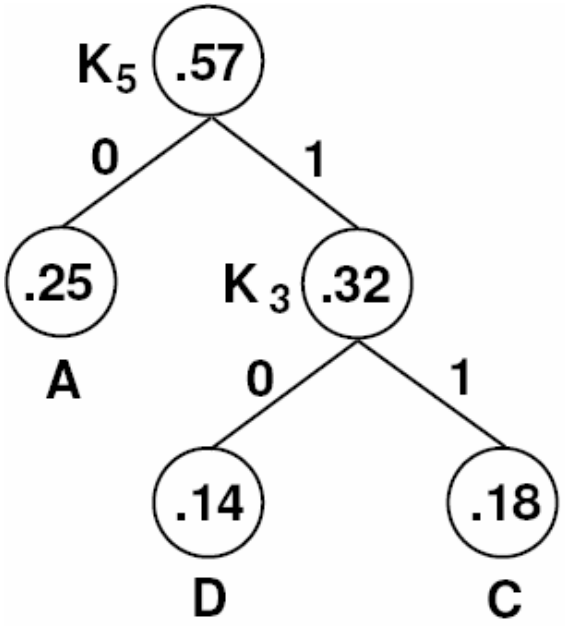


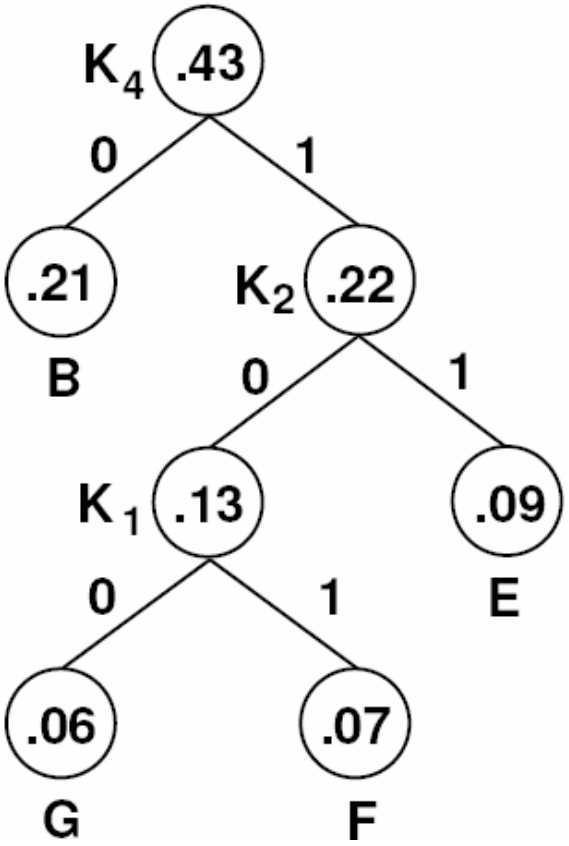
17.9 Beispiel H-Baumes (5)

x_i	K_4	K_3	A
p_i	.43	.32	.25

Auswahl: A, K_3

K_5 = .57



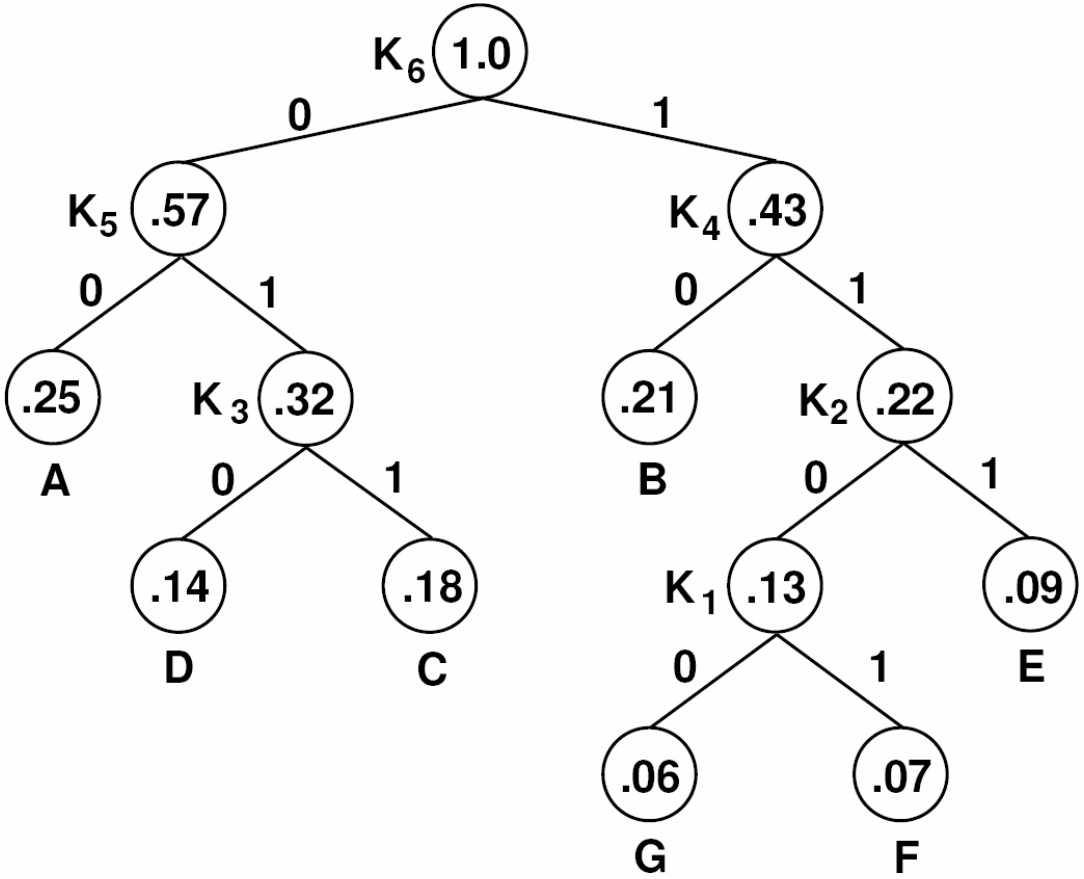


17.9 Beispiel H-Baumes (6)

x_i	K_5	K_4
p_i	.57	.43

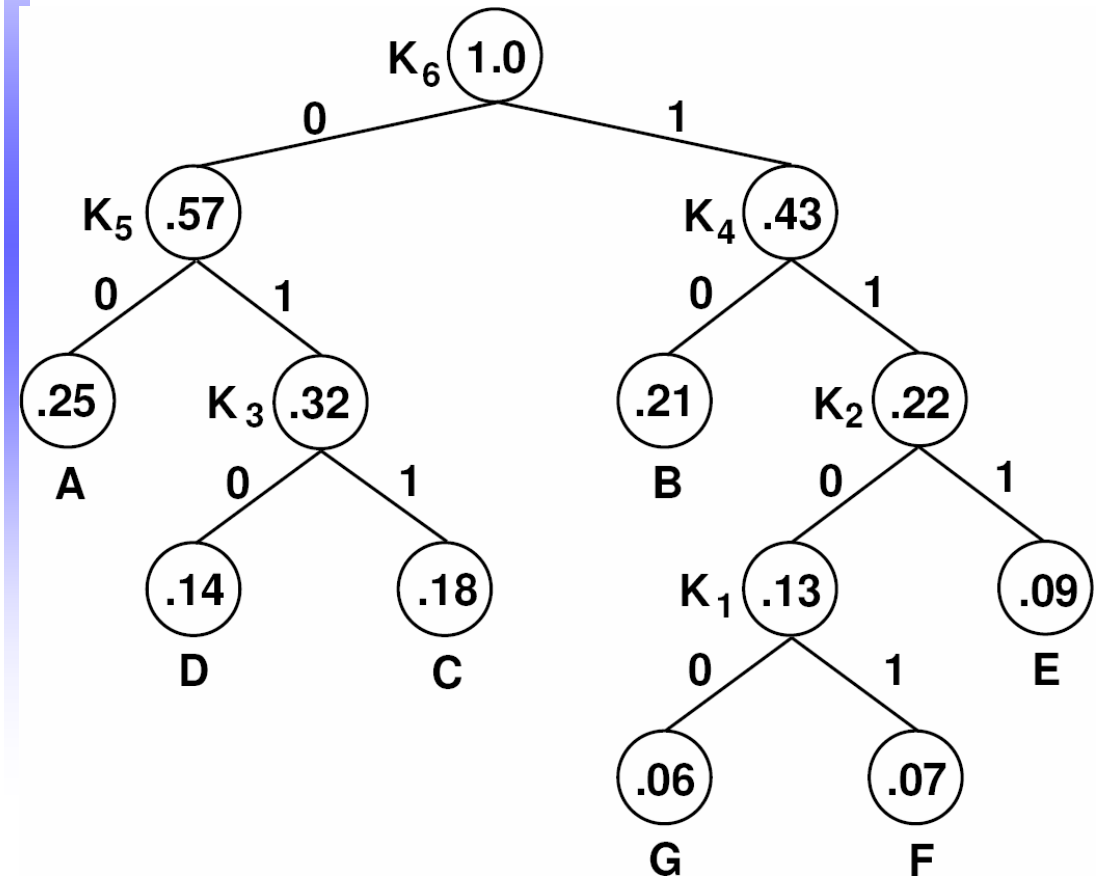
Auswahl: K_4, K_5

$K_6 = 1.0$



17.9 Beispiel H-Baumes (7)

x_i	A	B	C	D	E	F	G
p_i	.25	.21	.18	.14	.09	.07	.06



A: 00

G: 1100

0011011101111 **?**

➡ **A F F E**

Präfixfreier Code

17.9 Fazit

7 Zeichen

➡ L = **3** bit/Zeichen bei Codierung gleicher Länge

Huffman:

x_i	A	B	C	D	E	F	G
p_i	.25	.21	.18	.14	.09	.07	.06
l_i	2	2	3	3	3	4	4

➡ L = **2,67** bit/Zeichen