Einführung in die Informatik I

Winter 2005/2006

4 Codierung

4 Codierung ...

Motivation: Ziele dieses Abschnittes

- Zahlensysteme für den Rechner
- Codierung von Zahlen & Zeichen
- Zahlentypen

Herausforderungen:

- ungewohnte Denkweise im Umgang mit Zahlen & Zeichen
- Darstellung von Zahlen mit eingeschränkter Genauigkeit
- wichtig: Daten $\neq$ Information

Literatur:

- [Ernst] Auszüge aus Kapitel 1 & 3
- `http://www.computerbase.de/lexikon/IEEE_754`
- `http://www.h-schmidt.net/FloatApplet/IEEE754de.html`

Additionssystem

- Symbole (***Ziffern***) für ausgesuchte Zahlen
- allgemeine Zahlen werden als Summe von Ziffern dargestellt
- Zahlen können sehr lang werden
- **Beispiel: Römisch-etruskische Zahlen**
  - Ziffern: $I = 1, V = 5, X = 10, L = 50, C = 100, \ldots$
  - Ziffern werden mit abnehmender Wertigkeit notiert
  - Keine Ziffer mehr als drei Mal hintereinander
  - Eine kleinere Ziffer, die vor einer größeren steht, wird von dieser abgezogen
  - Beispiele:
    $3 = III, 14 = XIV, 149 = CIL, 205 = CCV$

Stellenwertsystem

- beliebig große Zahlen mit festem Ziffernsatz darstellbar
- Wichtige Beispiele: Dezimalsystem („tägliches Leben“), Dualsystem, Hexadezimalsystem, Oktalsystem (Rechner), 12er-System (Uhrzeit)
- Besitzt eine Basis (Dezimal: 10, Dual: 2, Hexadezimal: 16, Oktal: 8)
- Wertigkeit einer Ziffer steigt (gewöhnlich) von rechts nach links
- Wertigkeit ergibt sich aus inkrementellen Multiplikationen der Basis
- Ziffer multipiziert mit Wertigkeit ergibt „Beitrag“ zur Gesamtzahl

Beispiel: Dezimalsystem

Ziffern: $0, 1, 2, 3, 4, 5, 6, 7, 8, 9$

10^4	10^3	10^2	10^1	10^0
5	6	3	8	6

$$56386_{10} = 10^4 \cdot 5 + 10^3 \cdot 6 + 10^2 \cdot 3 + 10^1 \cdot 8 + 10^0 \cdot 6$$

Beispiel: Binärsystem

Ziffern: $0, 1$

2^4	2^3	2^2	2^1	2^0
1	0	1	1	0

$$10110_2 = 2^4 \cdot 1 + 2^3 \cdot 0 + 2^2 \cdot 1 + 2^1 \cdot 1 + 2^0 \cdot 0 = 22_{10}$$

Beispiel: Oktalsystem

Ziffern: $0, 1, 3, 4, 5, 6, 7$

8^4	8^3	8^2	8^1	8^0
7	3	2	6	7

$$73267_8 = 8^4 \cdot 7 + 8^3 \cdot 3 + 8^2 \cdot 2 + 8^1 \cdot 6 + 8^0 \cdot 7 = 30391_{10}$$

Beispiel: Hexadezimalsystem

Ziffern: $0, \ldots, 9, A(10), B(11), C(12), D(13), E(14), F(15)$

16^4	16^3	16^2	16^1	16^0
A	9	5	F	F

$$A95FF_{16} = 16^4 \cdot A + 16^3 \cdot 9 + 16^2 \cdot 5 + 16^1 \cdot F + 16^0 \cdot F = 693759_{10}$$

Motivation: Wozu andere Stellenwertsysteme?

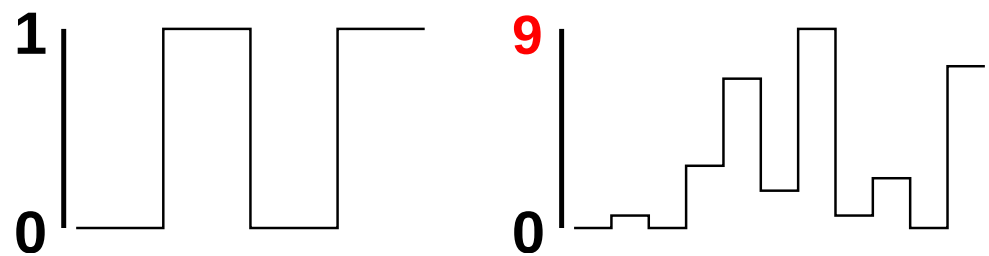
Dezimalsystem ungeeignet für elektronische Rechner

- Nur zwei einfache elektrische Zustände: Strom fließt oder Strom fließt nicht
- Bauteile für zwei Zustände sind einfach, günstig und vor allem platzsparend realisierbar
- Bezug zu Aussagenlogik: $0 =$ false, $1 =$ true
  $\Rightarrow$ Operationen wie $<, \leq, >, \geq, \neq, \ldots$ sinnvoll umsetzbar

Theoretisch: Mehr Zustände unterscheidbar

$\Rightarrow$ Bereits kleine Stromschwankungen führen zu falschen Ergebnissen

- 2 Zustände: 50% Stromschwankung $\rightarrow$ Fehler
- 10 Zustände: 5% Stromschwankung $\rightarrow$ Fehler



Oktal- und Hexadezimalsystem: Warum sind diese von Bedeutung?

- Basen 8 und 16 sind 2er-Potenzen
  $\Rightarrow$ Umrechnungen zwischen diesen Systemen besonders einfach
- Darstellungen von Binärdateien im Oktal- oder Hexadezimalsystem sind deutlich kompakter & lesbarer

Beispiel: Vergleichen Sie folgende Zahlenpaare

$$11101011110110011101110111111010111_2$$
$$11101011110110011101111011111010111_2$$

$$727547357727_8$$
$$727547367727_8$$

$$EBD9DDFD7_{16}$$
$$EBD9DEFD7_{16}$$

Umrechnung: Dezimal → Binär

Vorgehen: Division mit Rest (Modulo-Rechnung)

Beispiel:

$$22_{10} = ?_2$$

$$\begin{aligned} 22 &: 2 = 11 \text{ Rest } 0 \\ 11 &: 2 = 5 \text{ Rest } 1 \\ 5 &: 2 = 2 \text{ Rest } 1 \\ 2 &: 2 = 1 \text{ Rest } 0 \\ 1 &: 2 = 0 \text{ Rest } 1 \end{aligned}$$

$$\Rightarrow 22_{10} = 10110_2$$

Umrechnung: Hexadezimal ↔ Binär

Vorgehen: Einfaches Aneinanderreihen bzw. Blockbildung von Binärzahlen

Beispiel:

$$A95FF_{16} = ?_2$$

$$\begin{aligned} A_{16} &= 1010_2 \\ 9_{16} &= 1001_2 \\ 5_{16} &= 0101_2 \\ F_{16} &= 1111_2 \\ F_{16} &= 1111_2 \end{aligned}$$

$$\Rightarrow A95FF_{16} = \underbrace{1010}_{A}\ \underbrace{1001}_{9}\ \underbrace{0101}_{5}\ \underbrace{1111}_{F}\ \underbrace{1111}_{F}{}_2$$

ASCII: American Standard Code for Information Interchange

Ziel:

- Standardisierter Zeichensatz zum Austausch von Daten
- Zuordnung von Zeichen der Schriftsprache zu Binärzahlen zuordnen
  $\Rightarrow$ Nur so können Computer Zeichen nutzen!

Ansatz:

- Basiert auf lateinischem Alphabet und arabischen Zahlen
- mind. 62 Standard-Zeichen (0-9, a-z, A-Z) plus Sonderzeichen
  $\Rightarrow$ mind. 7 bit notwendig (128 Zeichen)

Alternative Zeichensätze, z.B.

- ISCII (Indisch) oder
- ISO8859 (15 verschiedenen Codierungen zur Abdeckung aller europäischen Zeichen und Thai)

ASCII (Forts.)

NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
SP	!	"	#	$	%	&	’	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	¡	=	¿	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{	—	}	~	DEL

Tabelle 1: ASCII Tabelle

- Erste 32 Codes: Steuerzeichen wie Leerzeichen (SP) oder Glocke (BEL)
- Viele der Steuerzeichen sind nicht-druckbar und historisch bedingt (z.B. Steuerung von Druckern)

Unicode und UTF

Unicode: Legt einheitliche Codes für alle sinntragenden Zeichen aller Kulturen fest

UTF (Unicode Transformation Format):

- bildet Unicode auf Codes unterschiedlicher Länge ab
- häufige Zeichen erhalten kurze Codes
- ASCII-Zeichen in UTF-8: Führende $\mathbf{0}$ und 7 bit umfassen ASCII-Zeichen
- Beispiel:

$$A = 1000001_{ASCII} = 01000001_{UTF8}$$

Wozu beschäftigen wir uns mit Information?

Grundsätzlich:

- Bei der Übertragung und Speicherung von Daten ist Effizienz gefragt
- Einige Daten werden häufiger benötigt als andere (z.B. die Buchstaben „e“ und „s“ in deutschen Texten)
  ⇒ Daraus lassen sich Vorteile bei einer Übertragung ziehen

Fragestellung: Wie lassen sich Daten kompakt darstellen (***komprimieren***)?

Komprimierungsarten:

- ***Verlustfrei***: Ursprüngliche Daten lassen sich vollständig wieder herstellen (wichtig z.B. bei Text)
- ***Verlustbehaftet***: Ursprüngliche Daten lassen sich nur unvollständig wieder herstellen (z.B. bei Bildern)

Allgemeiner Informationsbegriff

- Von verschiedenen Wissenschaften (Informatik, Informationstheorie, Semiotik,. . .) unterschiedlich definiert
- Aktuell wird versucht einen interdisziplinär einheitlichen Informationsbegriff zu finden
- Allgemein: Potenziell oder tatsächlich vorhandenes nutzbares oder genutztes Wissen
- Wesentlich sind die Wiedererkennbarkeit und der Neuigkeitsgehalt
- Ist für einen Betrachter in einem bestimmten ***Kontext*** von Bedeutung und verändert dessen inneren Zustand (beim Menschen: dessen ***Wissen***)
- Vgl. Datum: Quasi „Einheit“ von Information, jedoch selbst keine Information (Beispiel: Zeichen)

Wichtige Begriffe

Alphabet:

- Besteht aus einer ***abzählbaren Menge von Zeichen*** . . .
- . . . und einer ***Ordnungsrelation***
- Üblicherweise endlicher Zeichenvorrat

Beispiel:

- $\{a, b, c, \ldots z\}$ (Menge aller Kleinbuchstaben in alphabetischer Ordnung)
- $\{0, 1, 2, \ldots 9\}$ (Menge der ganzen Zahlen 0 bis 9 mit der „$\leq$“-Relation)

Nachricht: Besteht aus Sequenz beliebiger Länge von Elementen von A

Nachrichtenraum A^*: Menge aller Nachrichten über dem Alphabet A

Länge einer Nachricht $X \in A^*$: $|X|$

Nachricht fester Länge n: $A^n = \{X \in A^* \ : \ |X| = n\}$

Shannon'scher Informationsbegriff (Claude Shannon, 1950)

Ausgangspunkt: Eine Nachricht $X = x_1 x_2 x_3 \ldots x_k \in A^*$

Statistischer Informationsgehalt $I(a)$ eines Zeichens: Mindestanzahl der zur Identifizierung des Zeichens $a \in A$ in der Nachricht nötigen Bits

Auftrittswahrscheinlichkeit von $a \in A$ in X:

$$w_X(a) = \frac{\text{Häufigkeit von } a \text{ in } X}{\text{Länge von } X}$$

Konkrete Funktion für Informationsgehalt:

$$I(a) = \log_2 \frac{1}{w_X(a)} = -\log_2 w_X(a)$$

Eigenschaften des Informationsgehalts nach Shannon:

1. Je kleiner $w_X(a)$, desto größer $I(a)$
2. Für ein Zeichen mit $w_X(a) = 1$ gilt: $I(a) = 0$

Entropie

Ausgangspunkt: Eine Nachricht $X = x_1 x_2 x_3 \ldots x_k \in A^*$

Bekannt: Statistischer Informationsgehalt $I(a)$ aller Zeichen von A

Mittlerer Informationsgehalt (Entropie) der Nachricht X:

$$H(X) = \sum_{a \in A} \frac{1}{\log_2(w_X(a))} \cdot w_X(a) = -\sum_{a \in A} \log_2(w_X(a)) \cdot w_X(a)$$

H **entspricht der theoretischen Untergrenze** von Bit pro Zeichen zur Codierung der Nachricht X

Beachte: Nicht die Abfolge sondern nur die Häufigkeit der Zeichen wird verwendet

$\Rightarrow$ „geschickte“ Codierung kann mit weniger Bits auskommen

Beispiel:

- Nachricht: $AABBDCDDAD$
- Alphabet: $\{A, B, C, D\}$
- Absolute Häufigkeiten: $\{3, 2, 1, 4\}$
- Relative Häufigkeiten: $\{\frac{3}{10}, \frac{2}{10}, \frac{1}{10}, \frac{4}{10}\}$
- $H = -(\log_2(\frac{3}{10}) \cdot \frac{3}{10} + \log_2(\frac{2}{10}) \cdot \frac{2}{10} + \log_2(\frac{1}{10}) \cdot \frac{1}{10} + \log_2(\frac{4}{10}) \cdot \frac{4}{10}) = 1.85$

$\Rightarrow$ Zur Codierung jedes Zeichens werden 1.85 Bit benötigt

$\Rightarrow$ Zur Codierung der Nachricht werden 18.5 Bit benötigt

- weitere Beispiele:

Rel. Häufigkeiten: $\{1, 0, 0, 0\}$ $\quad H = -(\log_2(1) \cdot 1 + 3 \cdot \log_2(0) \cdot 0) = 0$

Rel. Häufigkeiten: $\{\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\}$ $\quad H = -(4 \cdot \underbrace{\log_2(\frac{1}{4})}_{=-2} \cdot \frac{1}{4}) = 2$

4.3.2 Runlength-Encoding

Ziel: Sehr einfacher, verlustfreier Kompressionsalgorithmus

Ansatz: Verkürzte Sequenz für wiederholte Zeichen durch Speicherung ihrer Anzahl

Beispiel:
- Alphabet $A = \{S, W\}$
  - Vor der Anwendung: $SSSSSSSSSSWSSSSSSSSSSWW$
  - Nach RLE: $10SW10S2W$

Probleme: Codierung der Wiederholzahl, wenn Zahlen selbst im Alphabet enthalten

Kontrollzeichen (Escape-Sequenz): Verwendung eines Sondersymbols (Escape-Zeichen), z.B. '\';

Beispiel:
- '\7' steht für Wiederholzahl 7
- '\\' steht für Zeichen '\'

Huffman-Codierung (David Huffman, 1952)

Ziel: Effiziente Codierung von Nachrichten unter Berücksichtigung der Auftrittswahrscheinlichkeiten der Zeichen

Idee: Verwendung von kurzen Codes für häufiger auftretende Zeichen

$\Rightarrow$ Zeichen-Codes mit variabler Länge & abhängig von Nachricht

Herausforderung: Verhinderung von Mehrdeutigkeiten; Beispiel:

- 3 Zeichen (A, B, C) mit steigender Wahrscheinlichkeit $(w(A) < w(B) < w(C))$
- Ungeschickte Codierung: $A = 10, B = 01, C = 0$
- Nun kann die Zeichenfolge 10010 entweder ABC oder ACA bedeuten

Lösung: Keine Code für ein Zeichen darf Anfang eines anderen Codes sein

Im Beispiel ist die Codierung für C Anfang der Codierung für B

Alternative Bezeichnung: Entropie-Codierung

4.3.3 Huffman-Codes ...

Algorithmus:

1. Sortiere Zeichen aufsteigend nach ihrer Wahrscheinlichkeit
2. Zusammenfassung der beiden Zeichen mit der geringsten Wahrscheinlichkeit
   - Vergabe von (zusätzlichen) Bits für beide Zeichen
   - Addition der Wahrscheinlichkeiten beider Zeichen
3. Wiederhole die Schritte bis sich nur noch ein Zeichen in der Sortierung befindet

4.3.3 Huffman-Codes ...

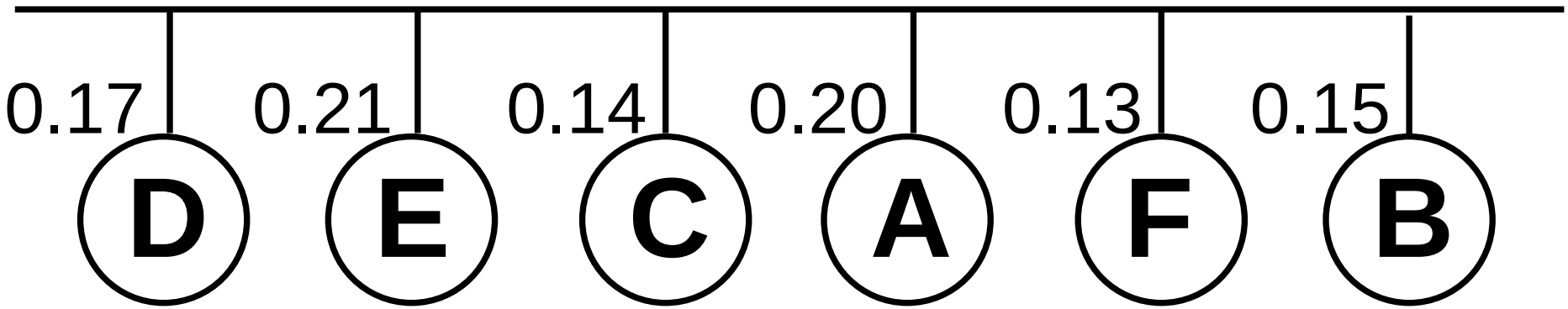
Algorithmus:

1. Sortiere Zeichen aufsteigend nach ihrer Wahrscheinlichkeit
2. Zusammenfassung der beiden Zeichen mit der geringsten Wahrscheinlichkeit
   - ❍ Vergabe von (zusätzlichen) Bits für beide Zeichen
   - ❍ Addition der Wahrscheinlichkeiten beider Zeichen
3. Wiederhole die Schritte bis sich nur noch ein Zeichen in der Sortierung befindet

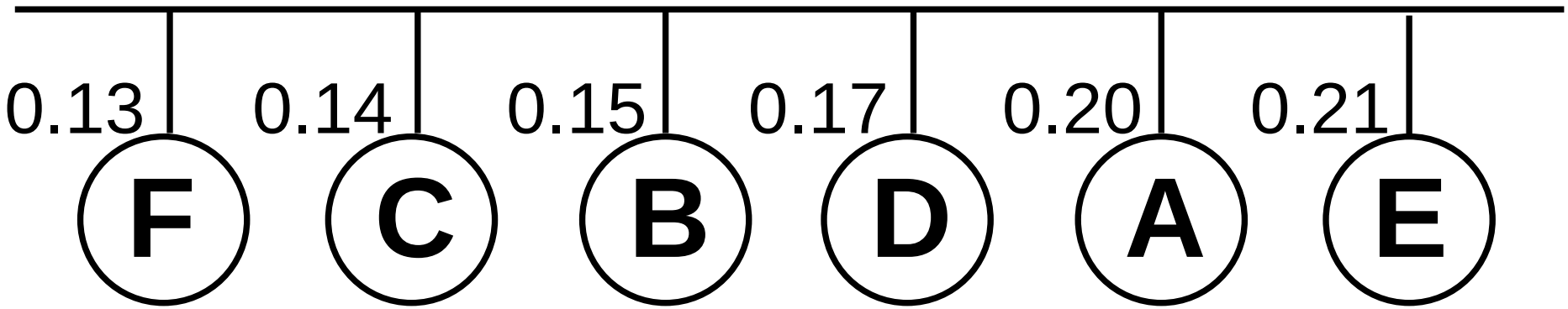
Beispiel: Nachricht „HAAAALLO“

Z	w	C	Z	w	C	Z	w	C	Z	W	C
H	1/8		H	2/8	0	H		00	H		000
A	4/8		O		1	O	4/8	01	O	8/8	001
L	2/8		A	4/8		L		1	L		01
O	1/8		L	2/8		A	4/8		A		1

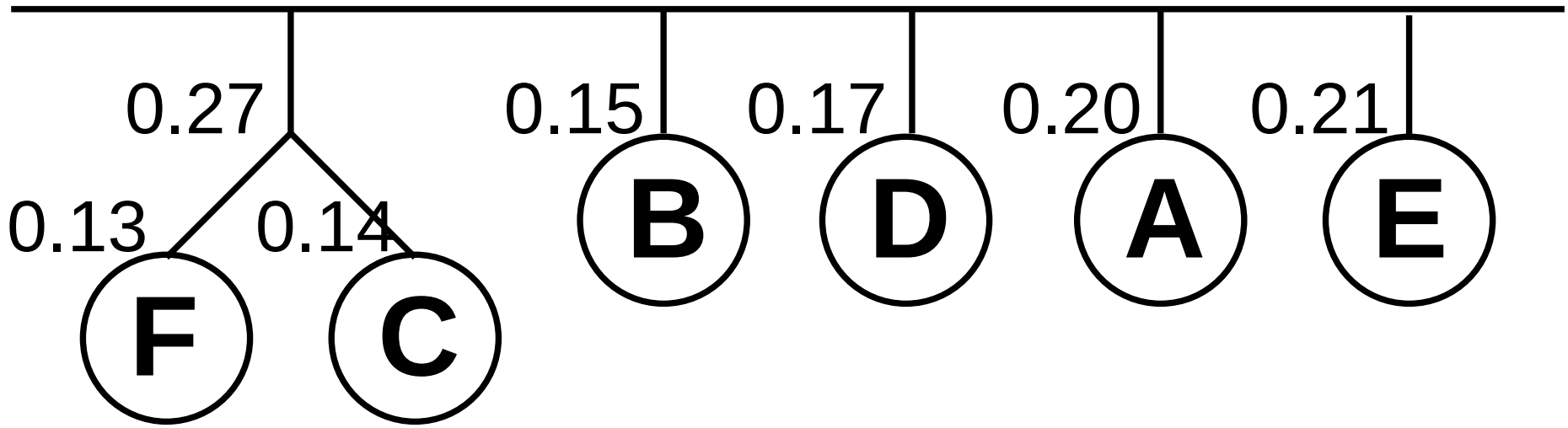
Beispiel: Darstellung als Binärbaum



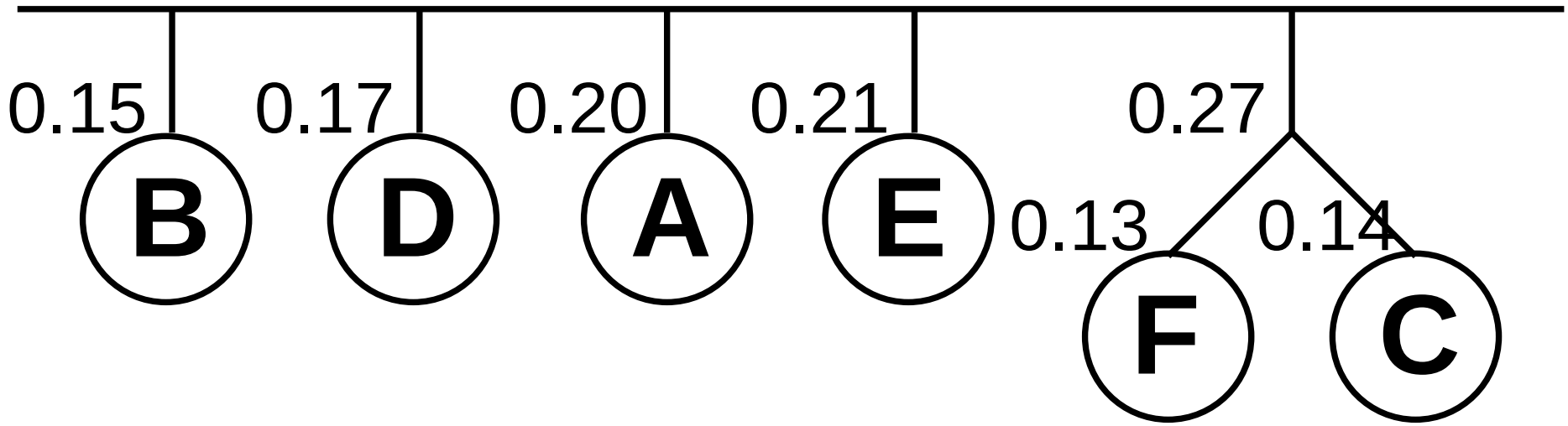
Beispiel: Darstellung als Binärbaum



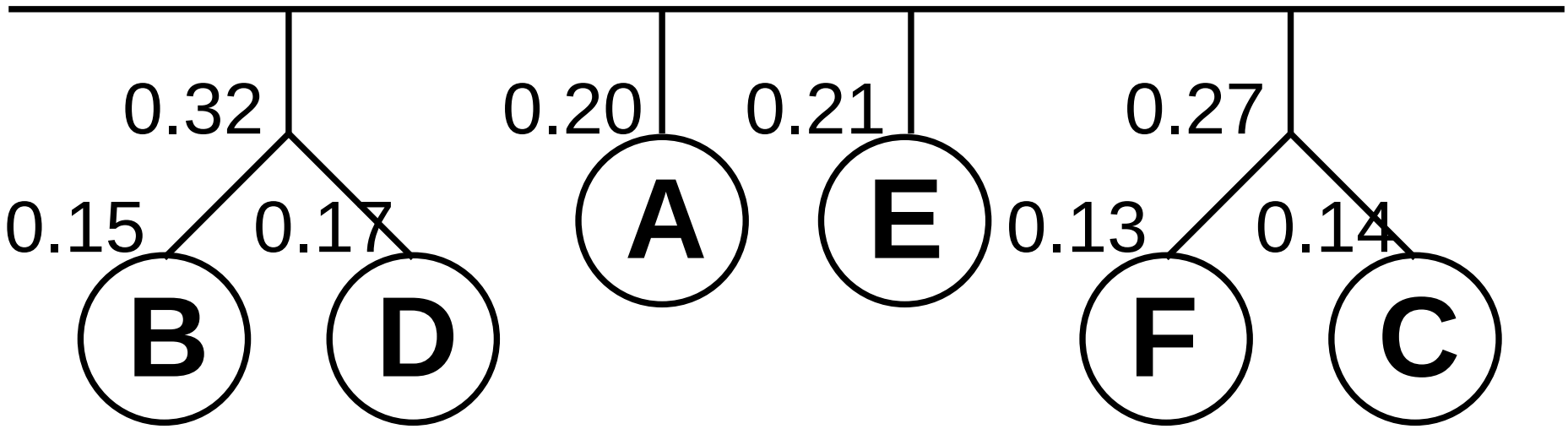
Beispiel: Darstellung als Binärbaum



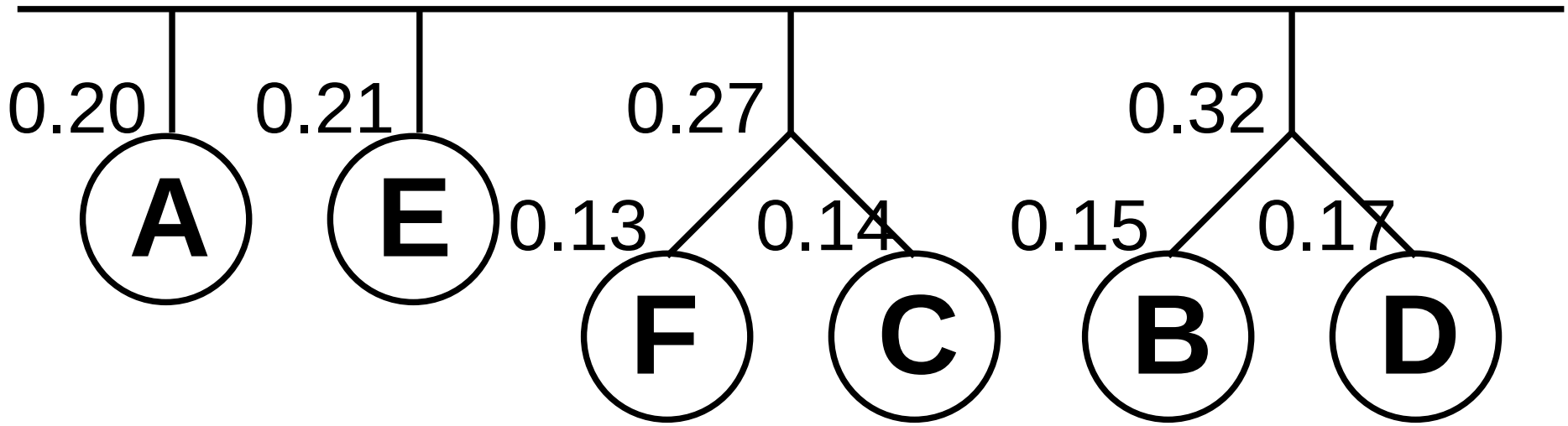
Beispiel: Darstellung als Binärbaum



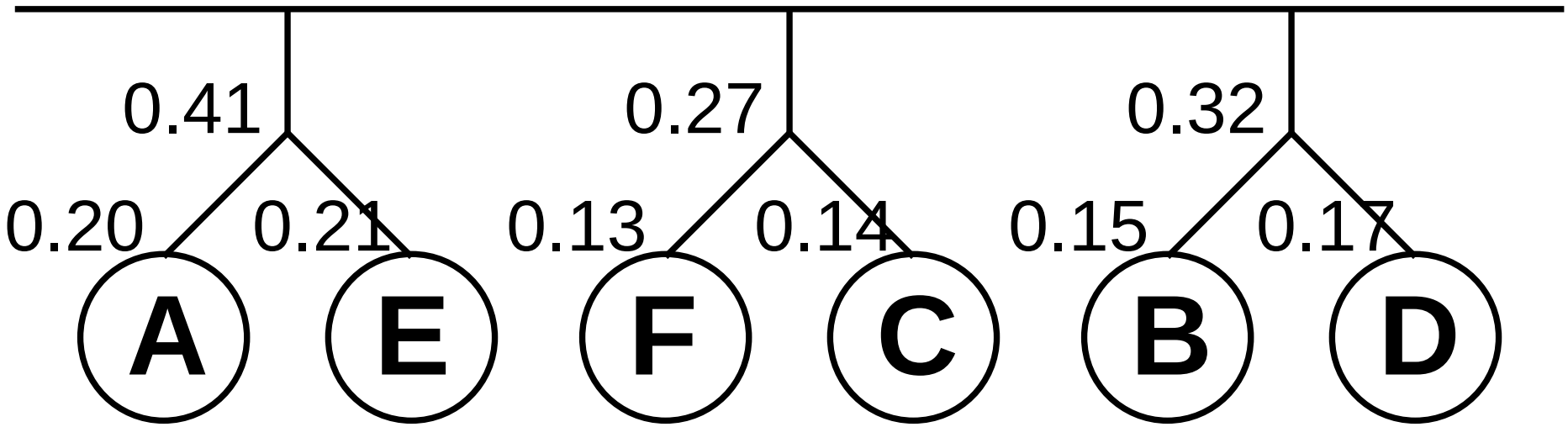
Beispiel: Darstellung als Binärbaum



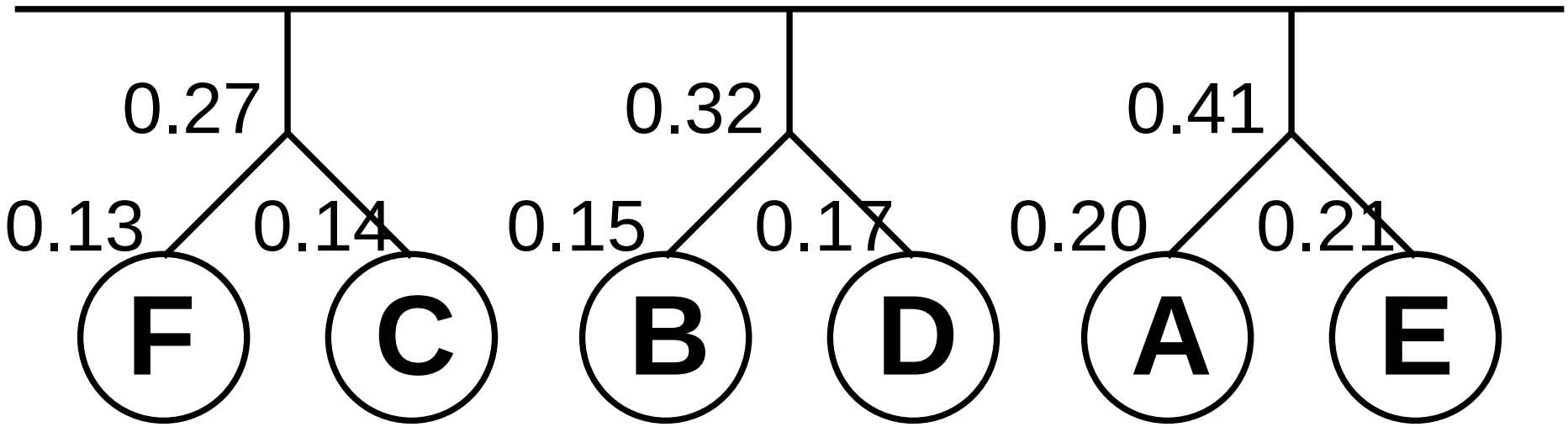
Beispiel: Darstellung als Binärbaum



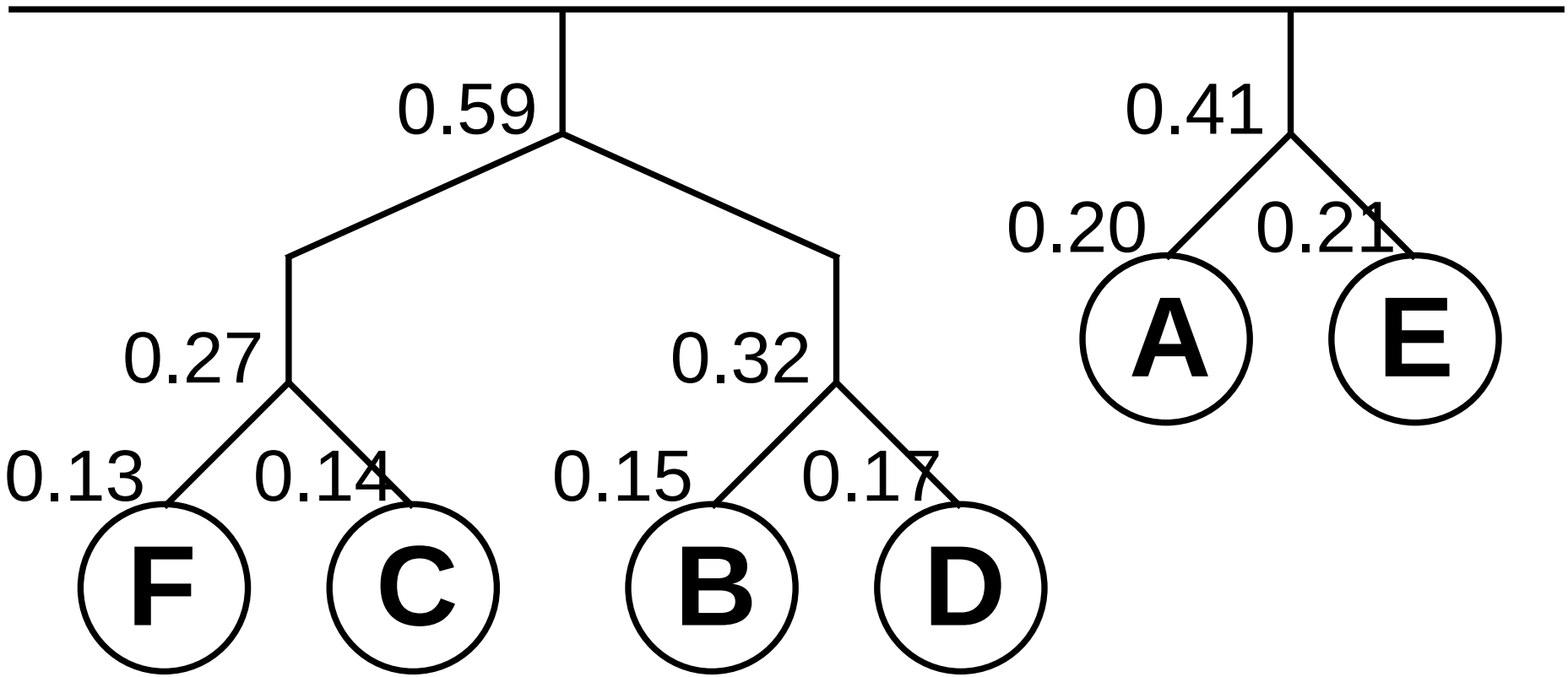
Beispiel: Darstellung als Binärbaum



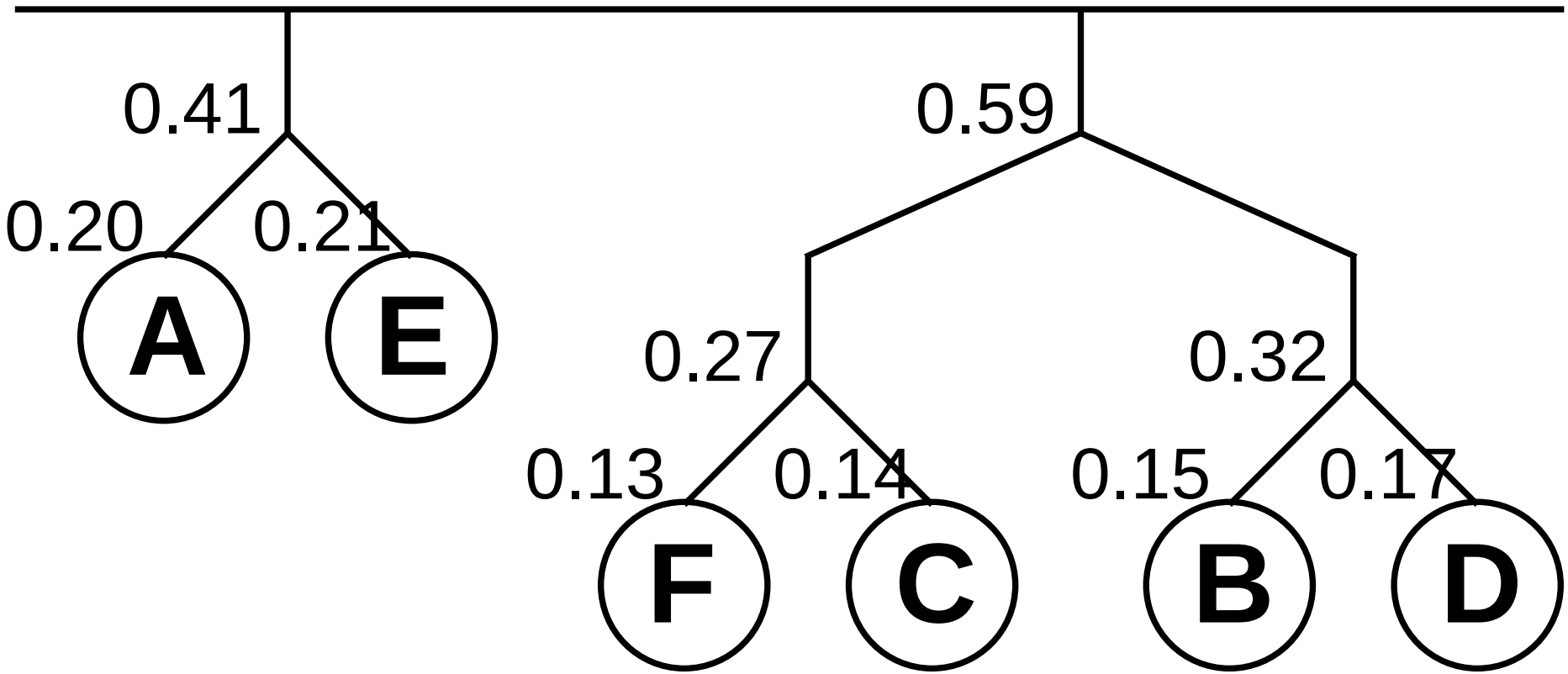
Beispiel: Darstellung als Binärbaum



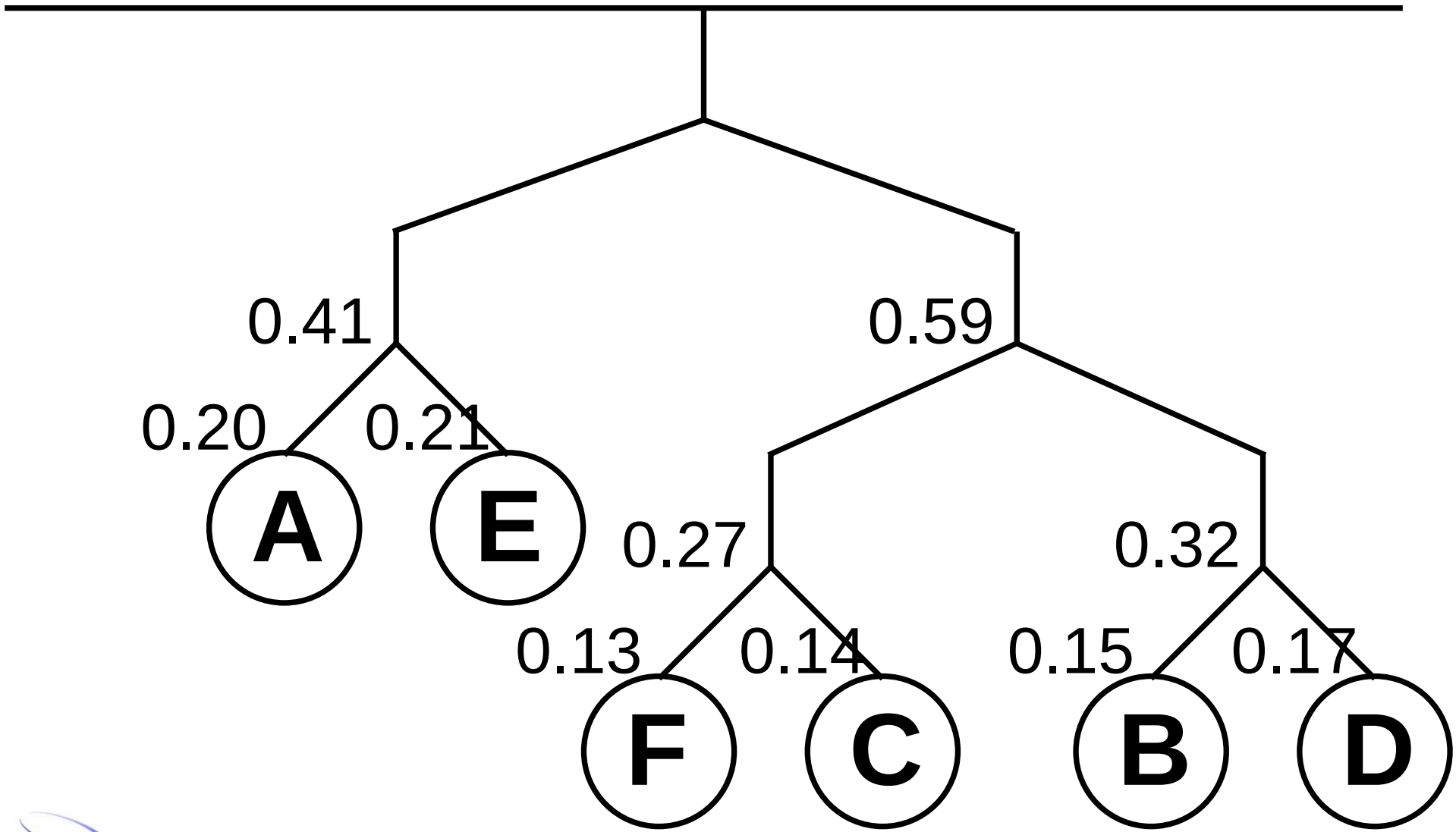
Beispiel: Darstellung als Binärbaum



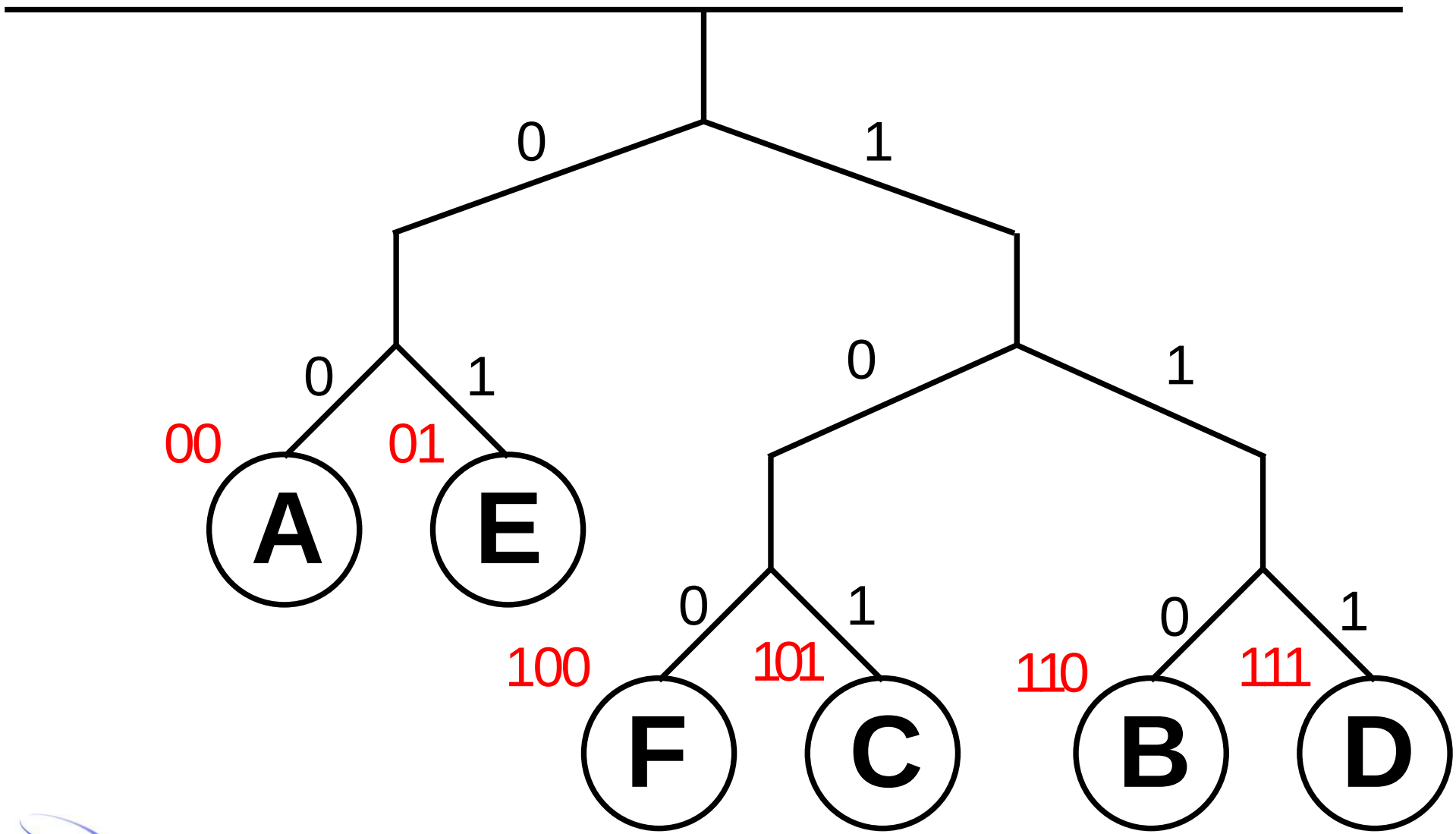
Beispiel: Darstellung als Binärbaum



Beispiel: Darstellung als Binärbaum



Beispiel: Darstellung als Binärbaum



4.4 Zahlentypen

Ziel: Darstellung von Zahlen und Operationen (insbesondere Ganzzahlen, Gleitkommazahlen)

Formal: Zusammenfassung von Objekten einer „Art“ und auf ihnen definierten Operationen (z.B. Addition bei Ganzzahlen)

Folgerung: Einordnung auf konkrete Wertebereiche und darauf definierte Operationen

Herausforderung: Zahlentypen entscheiden über die Interpretation von Werten
(Erinnerung: Computer kennen grundsätzlich nur 0 und 1)

4.4.1 Elementare Zahlentypen

Elementare Zahlentypen:

- Endlicher ***Wertebereich*** durch eingeschränkten Speicherumfang
- Fest definierte Unter- und Obergrenze

Bemerkung:

Für die Bezeichnungen soll hier vorrangig die Notation der Programmiersprache C verwendet werden

Ganzzahlen

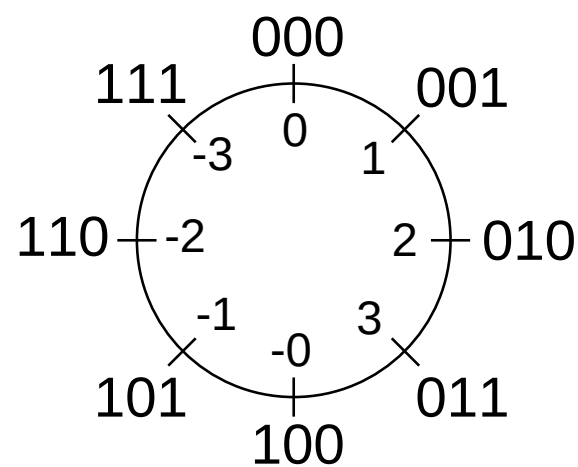
- Bezeichnungen:
  - Vorzeichenbehaftet: `int` (auch `Integer`)
  - Ohne Vorzeichen: `unsigned int` (auch `Word, natural`)
- Länge: 32 Bit (auch 16 Bit `short` oder 64 Bit `long`)
- Wertebereich bei 32 Bit:
  - Vorzeichenbehaftet: $-2^{31}..2^{31}-1$
  - Ohne Vorzeichen: $0..2^{32}-1$

Ganzzahl-Repräsentation: Vorzeichen-Betrag Darstellung

- „Naiver“ Ansatz:
  - Erstes Bit bestimmt Vorzeichen (0 positiv, 1 negativ)
  - Restliche Bits werden zur Codierung des Betrags verwendet
- Problem: Darstellung nicht eindeutig durch doppelte Null (positive und negative)

Beispiel:

$$6_{10} = 0110_2$$
$$-6_{10} = 1110_{\text{Vorz.-Betrag}}$$



Ganzzahl-Repräsentation: Zweierkomplement

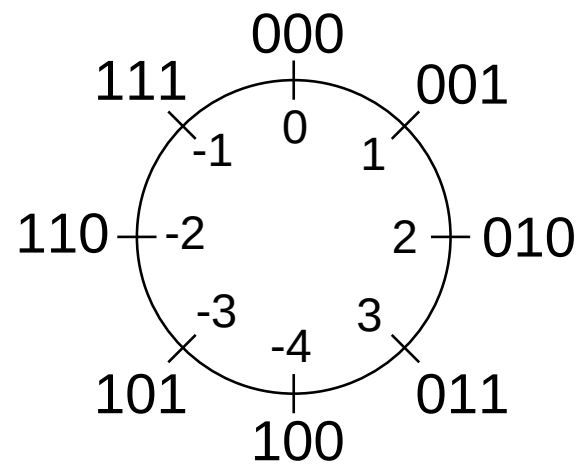
- Negative Zahlen: Invertiere positive Zahl bitweise und addiere 1

Beispiel:

$$
\begin{aligned}
\mathbf{6_{10}} &= \mathbf{0110_2} \\
\mathbf{-6_{10}} &= \mathbf{\overline{0110_2} + 1_2} \\
&= \mathbf{1001_2 + 1_2} \\
&= \mathbf{1010}_{\text{2er-Kompl.}}
\end{aligned}
$$

Überprüfung:

$$
\begin{aligned}
\mathbf{-1010}_{\text{2er-Kompl.}} &= \mathbf{\overline{1010_2} + 1_2} \\
&= \mathbf{0101_2 + 1_2} \\
&= \mathbf{0110_2 = 6_{10}}
\end{aligned}
$$



4.4.1 Elementare Zahlentypen ...

- Gilt auch hier: Führende $1 \Leftrightarrow$ negative Zahl
- Vorteil gegenüber Vorzeichen-Betrag: Eindeutige 0

$$
\begin{aligned}
0_{10} &= 0000_2 \\
-0_{10} &= \overline{0000_2} + 1_2 \\
&= 1111_2 + 1_2 \\
&= 0000_{\text{2er-Kompl.}}
\end{aligned}
$$

Vergleich Vorzeichen-Betrag und Zweierkomplement

- Addition grundsätzlich gleich: Bitweise mit Übertrag
- Subtraktion in Vorzeichen-Betrag Darstellung jedoch komplizierter:
  1. Bestimmen der Zahl mit größerem Betrag
  2. Subtrahieren des kleineren Betrags vom größeren
  3. Vorzeichen der Zahl mit größerem Betrag bestimmt Vorzeichen des Ergebnisses
- Subtraktion im Zweierkomplement: Rückführung auf Addition der negativen Zahl
  1. Bilden des Zweierkomplements der zweiten Zahl (Negation)
  2. Addieren: Erste Zahl mit Zweierkomplement der zweiten Zahl

Beispiel: $3_{10} - 5_{10}$

- Vorzeichen-Betrag-Darstellung:

  $3_{10} < 5_{10}$? Nein $\Rightarrow$ Berechne $5_{10} - 3_{10}$:

$$\begin{array}{r} 0101 \\ -0011 \\ \hline = 0010 \end{array}$$

  Da 5_{10} größerer Betrag und negativ, ist Ergebnis $1010_{\text{Vorz.-Betrag}}$

- Zweierkomplement-Darstellung: $a - b = a + (-b)$

  Bilde Zweierkomplement von 5: $5 = \overline{0101} + 1 = 1010 + 1 = 1011$
  Addiere:

$$\begin{array}{r} 0011 \\ +1011 \\ \hline 1110 \end{array}$$

- Ergebnis: $1110_{\text{2er-Kompl.}}$

Ganzzahl-Repräsentation: Biased-Notation

- $00..0$ repräsentiert die kleinstmögliche Zahl, $11..1$ die größtmögliche
- Auf jede Zahl wird die sogenannte ***Magic Number*** addiert
- **Beispiel: 8-bit Zahl ($n = 8$), Magic Number: 127**

	Zahl	Berechnung	Biased Notation
Kleinste Zahl	-127	$-127 + 127 = 0$	00000000_{biased}
Größte Zahl	128	$128 + 127 = 255$	11111111_{biased}
Null:	0	$0 + 127 = 127$	01111111_{biased}

- Vorteil:
  - Größenvergleiche können bitweise durchgeführt werden
- Problem:
  - Mathematische Operationen können nicht mehr direkt durchgeführt werden

Gleitkommazahlen

- Bezeichnungen: `float`
- Länge: 32 Bit (auch 16 Bit `half` oder 64 Bit `double`)
- Wertebereich: Siehe folgende Folien
- Definiert im IEEE 754 Standard

Grundsätzliche Idee

Verwendung der wissenschaftlichen Schreibweise (***Scientific notation***)

Beispiele im Dezimalsystem:

$$1214.31 = 1.21431 \cdot 10^3$$

$$0.0213178 = 2.13178 \cdot 10^{-2}$$

Gleitkommazahl-Repräsentation (IEEE 754)

Aufbau: 1 Bit Vorzeichen (s), 8 Bit Exponent (e), 23 Bit Mantisse (m)

Vorzeichen: 0 positiv, 1 negativ

Exponent: Biased Notation mit Magic Number 127

(Hinweis: In [Ernst] steht hier fälschlicherweise 128)

Mantisse: Im Dualsystem nur 0 und 1

$\Rightarrow$ Stelle vor dem Komma immer 1, kann also weggelassen werden

Dargestellte Gleitkommazahl: $(-1)^s \cdot (1.0 + m) \cdot 2^{e-127}$

Konvertierung einer Gleitkommazahl nach IEEE 754

Gegeben: Gleitkommazahl der Form $\pm(g_{10} + r_{10})$ mit:
$g_{10} \in \mathbb{N}; r_{10} \in \mathbb{R}, 0 \leq r < 1.$

Am Beispiel von $25.421875 : g_{10} = 25, r_{10} = 0.421875, s = 0$

1. Bilde g_{10} auf einen binären Wert g_2 ab: $25_{10} = 11001_2$
2. Bilde r_{10} auf einen binären Wert r_2 ab (s.u): $0.421875_{10} = 0.011011_2$
3. Bilde vorläufige Mantisse $m = g_2 + r_2$
   $m = 11001_2 + 0.011011_2 = 11001.011011_2$
4. Forme m in wissenschaftliche Schreibweise um
   $m = 11001.011011_2 = 1.1001011011 \cdot 2^4 (e = 4)$
5. Endgültige Mantisse durch Weglassen der ersten 1: $m = 1001011011$
6. Endgütliger Exponent durch Addieren von 127
   $e = 4_{10} + 127_{10} = 100_2 + 01111111_2 = 10000011_{\text{biased}}$

Gesamtergebnis

$$25.421875_{10} = 0\ 10000011\ 10010110110000000000000_{\text{IEEE 754}}$$

Umwandlung von Nachkommastellen (Dezimal → Binär)

Beispiel: 0.421875_{10} (Nachkommazahl von oben)

Achtung: Direkte Umwandlung von 421875_{10} in Dualsystem und Interpretation als Nachkommawert ist falsch!

Beobachtung: Multiplikation mit Basis bringt Nachkommastelle vor Komma:

$$0.421875_{10} \xrightarrow{\cdot 10} \boxed{4}.21875_{10}; \qquad 0.21875_{10} \xrightarrow{\cdot 10} \boxed{2}.1875_{10}$$

Dies gilt auch für Multiplikation mit anderer Basis

0.421875 nach Binär:

$$0.421875 \xrightarrow{\cdot 2} \boxed{0}.84375 \qquad 0.84375 \xrightarrow{\cdot 2} \boxed{1}.6875$$

$$0.6875 \xrightarrow{\cdot 2} \boxed{1}.375 \qquad 0.375 \xrightarrow{\cdot 2} \boxed{0}.75$$

$$0.75 \xrightarrow{\cdot 2} \boxed{1}.5 \qquad 0.5 \xrightarrow{\cdot 2} \boxed{1}.0$$

Darstellbare Werte:

S	Exponent	Mantisse	Bedeutung
$0/1$	11111111_2	$0..00_2$	$+/-$ unendlich
$*$	11111111_2	$\neq 0..00_2$	NaN (Not a Number)
$0/1$	00000000_2	$0..00_2$	$+/-0$ (Null)
$*$	00000000_2	$\neq 0..00_2$	denormalisierte Gleitkommazahl
$*$	$0..01_2$ bis $1..10_2$	beliebig	normalisierte Gleitkommazahl
$*$	11111110_2	$1..11_2$	Größte darstellbare Zahl $(3.4 \cdot 10^{38})$
$*$	00000001_2	$0..1_2$	Kleinste darstellbare Zahl $(1.2 \cdot 10^{-38})$

Hinweis: Denormalisierte Gleitkommazahlen erweitern die Darstellungsgenauigkeit nahe der Null