

## **2. Grundlagen**

# Inhalt

- **2.1 Darstellung von Zahlen**
- **2.2 Darstellung von Zeichen**
- **2.3 Boolesche Algebra**
- **2.4 Aussagenlogik**
- **2.5 Logische Funktionen**

## 2.1 Darstellung von Zahlen

- Im Alltag rechnen wir gewöhnlich im **Dezimalsystem**, d.h. wir stellen Zahlen zur **Basis** 10 mit Hilfe der **Ziffern** 0, ..., 9 dar
- Schreiben wir die Zahl 1315, so bedeutet dies eigentlich
$$1315 = 1 \cdot 10^3 + 3 \cdot 10^2 + 1 \cdot 10^1 + 5 \cdot 10^0$$
- Wir können Zahlen zu **anderen Basen** (z.B. 2 und 16) darstellen und in diesen Darstellungen **rechnen** (z.B. addieren und subtrahieren)

# Darstellung von Zahlen mit Basisangabe

- Wir können zu Zahlen ihre **zugehörige Basis** angeben:

$$(1315)_{10}$$

bezeichnet dann die Zahl 1315 im Dezimalsystem,  
d.h. die **Zahl** 1315 steht in Klammern  
und die **Basis** 10 tiefgestellt dahinter

- In der Informatik spielen das **Binärsystem** (Basis 2)  
sowie das **Hexadezimalsystem** (Basis 16)  
eine wichtige Rolle

## 2.1.1 Binärsystem

- Rechner kennen intern **nur zwei Zustände** (Strom an/aus) und stellen Zahlen daher im **Binärsystem** zur **Basis 2** dar
- Als **Ziffern** stehen nur 0 und 1 zur Verfügung, welche als **Bits** bezeichnet werden
- Schreiben wir  $(10101)_2$ , so bedeutet dies

$$(10101)_2 = 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$

und entspricht im Dezimalsystem der Zahl

$(21)_{10}$

## 2.1.2 Hexadezimalsystem

- Das **Hexadezimalsystem** stellt Zahlen zur **Basis** 16 dar und wird ebenfalls oft in der Informatik verwendet
- Als **Ziffern** stehen 0, ..., 9, A, B, C, D, E, F zur Verfügung, wobei die Buchstaben A, ..., F die Ziffern 10, ..., 15 darstellen und so **Mehrdeutigkeiten** vermeiden
- Schreiben wir  $(FAB4)_{16}$ , so bedeutet dies

$$(FAB4)_{16} = 15 \cdot 16^3 + 10 \cdot 16^2 + 11 \cdot 16^1 + 4 \cdot 16^0$$

und entspricht im Dezimalsystem der Zahl

$(64180)_{10}$

# Anwendungsbeispiel Hexadezimalsystem

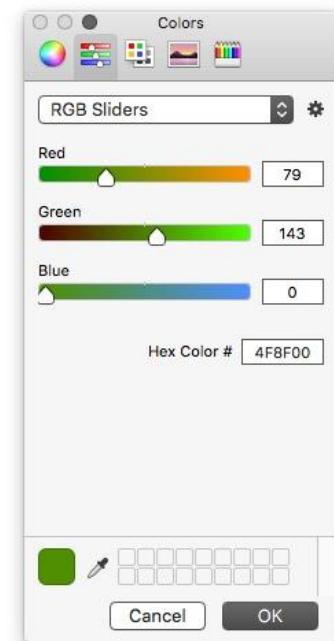
- Das Hexadezimalsystem wird zur Angabe von Farben gemäß des **RGB-Farbmodells** verwendet
- Farbe wird als Mischung der Grundfarben **Rot**, **Grün** und **Blau** angegeben und es stehen pro Grundfarbe  $256 = 16^2$  Abstufungen zur Verfügung



#FF2F92



#4BB4E7



## 2.1.3 Zahlendarstellung zu einer beliebigen Basis $b$

- Wir können Zahlen zu einer **beliebigen natürlichen Basis**

$$b \in \mathbb{N} \quad (b > 1)$$

darstellen (sogenannte  **$b$ -adische Darstellung**)

- Als **Ziffern** stehen die Zahlen  $0, 1, \dots, (b-1)$  zur Verfügung
- Schreiben wir nun eine Zahl  $(d_m \dots d_0)_b$  mit  $(m + 1)$  Stellen, so bedeutet dies

$$(d_m \dots d_0)_b = \left( \sum_{i=0}^m d_i \cdot b^i \right)_{10} = (d_m \cdot b^m + \dots + d_0 \cdot b^0)_{10}$$

$$b = 6$$

Als Ziffern stehen uns zur Verfügung 0, 1, 2, 3, 4, 5

$$\begin{array}{r} (110)_6 \\ \diagup \quad \diagdown \\ d_2 \quad d_1 \quad d_0 \\ \diagdown \quad \diagup \\ b \end{array} = 1 \cdot 6^2 + 1 \cdot 6^1 + 0 \cdot 6^0 \\ = (42)_{10}$$

$$\begin{array}{r} (54)_6 \\ \diagup \quad \diagdown \\ d_1 \quad d_0 \\ \diagdown \quad \diagup \\ b \end{array} = 5 \cdot 6^1 + 4 \cdot 6^0 \\ = (34)_{10}$$

## 2.1.4 Umwandeln zwischen den Zahlensystemen

- Wie können wir eine Zahl von ihrer **Darstellung** zur Basis  $b_1$  in ihre Darstellung zur Basis  $b_2$  **umwandeln**?
- **Dezimalsystem** dient uns als **Zwischendarstellung**
  - konvertiere aus ursprünglicher Darstellung ins Dezimalsystem
  - konvertiere aus Dezimalsystem in gewünschte Darstellung

$$(\dots)_{b_1} \rightarrow (\dots)_{10} \rightarrow (\dots)_{b_2}$$

- Später sehen wir, wie man zwischen bestimmten Zahlensystemen **schneller umwandeln** kann

# Umwandeln ins Dezimalsystem

- Eine Zahl  $(d_m \dots d_0)_b$  in b-adischer Darstellung wandeln wir gemäß deren Definition **ins Dezimalsystem** um, indem wir folgenden Wert berechnen

$$(n)_{10} = \sum_{i=0}^m d_i \cdot b^i$$

# Umwandeln ins Dezimalsystem

$$(11011)_2 =$$

$$\begin{aligned} & 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 \\ & = 16 + 8 + 2 + 1 = (27)_{10} \end{aligned}$$

$$(FAB)_{16} =$$

$$\begin{aligned} & 15 \cdot 16^2 + 10 \cdot 16^1 + 11 \cdot 16^0 \\ & = 15 \cdot 256 + 10 \cdot 16 + 11 \cdot 1 \\ & = (4011)_{10} \end{aligned}$$

$$(4032)_5 =$$

$$\begin{aligned} & 4 \cdot 5^3 + 0 \cdot 5^2 + 3 \cdot 5^1 + 2 \cdot 5^0 \\ & = 4 \cdot 125 + 3 \cdot 5 + 2 \cdot 1 \\ & = (517)_{10} \end{aligned}$$

$$(121)_8 =$$

$$\begin{aligned} & 1 \cdot 8^2 + 2 \cdot 8^1 + 1 \cdot 8^0 \\ & = 1 \cdot 64 + 2 \cdot 8 + 1 \cdot 1 = (81)_{10} \end{aligned}$$

# Ganzzahlige Division

- Zur Umwandlung benötigen wir die **ganzzahlige Division**
- **Ganzzahliger Rest** der Division von  $x$  durch  $b$

$$(x \bmod b) = x - b \cdot \left\lfloor \frac{x}{b} \right\rfloor$$

- **Ganzzahliger Quotient** der Division von  $x$  durch  $b$

$$(x \text{ div } b) = \left\lfloor \frac{x}{b} \right\rfloor$$

$\lfloor \rfloor$  bedeutet Abrunden ("floor")

$\lceil \rceil$  bedeutet Aufrunden ("ceil")

$$\left\lfloor \frac{36}{5} \right\rfloor = 7 = (36 \text{ div } 5)$$

$$(36 \text{ mod } 5) = 1$$

Ganzzahlige Division

36 geteilt durch 5 ist 7 Rest 1

# Ganzzahlige Division

$$(17 \bmod 5) =$$

2

$$(17 \text{ div } 5) =$$

3

$$(32 \bmod 7) =$$

4

$$(32 \text{ div } 7) =$$

4

$$(267 \bmod 16) =$$

11

$$(267 \text{ div } 16) =$$

16

$$(1025 \bmod 2) =$$

1

$$(1025 \text{ div } 2) =$$

512

# Umwandeln aus dem Dezimalsystem

- Aus dem Dezimalsystem können wir eine Zahl  $(n)_{10}$  mittels folgender **Berechnungsvorschrift** (später: Algorithmus) in **die  $b$ -adische Darstellung** umwandeln
  - $n \leftarrow n_{(10)}$  (hierbei ist  $\leftarrow$  eine Wertzuweisung)
  - $i \leftarrow 0$  (wir betrachten zuerst die letzte Stelle  $d_0$ )
  - **solange** gilt, dass  $n > 0$  ist
    - $d_i \leftarrow (n \bmod b)$  (ganzzahliger Rest der Division durch  $b$ )
    - $n \leftarrow (n \text{ div } b)$  (ganzzahliger Quotient der Division durch  $b$ )
    - $i \leftarrow (i + 1)$

# Umwandeln aus dem Dezimalsystem

- Beispiel: Umwandeln von  $(133)_{10}$  ins Binärsystem

■ $n = 133$		
■ $d_0 = (133 \bmod 2) =$	1	$n = 66$
■ $d_1 = (66 \bmod 2) =$	0	$n = 33$
■ $d_2 = (33 \bmod 2) =$	1	$n = 16$
■ $d_3 = (16 \bmod 2) =$	0	$n = 8$
■ $d_4 = (8 \bmod 2) =$	0	$n = 4$
■ $d_5 = (4 \bmod 2) =$	0	$n = 2$
■ $d_6 = (2 \bmod 2) =$	0	$n = 1$
■ $d_7 = (1 \bmod 2) =$	1	$n = 0$

$(133)_{10} = (10000101)_2$

# Umwandeln aus dem Dezimalsystem

- Beispiel: Umwandeln von  $(815)_{10}$  ins Hexadezimalsystem
  - $n = 815$
  - $d_0 = (815 \bmod 16) = F \quad n = 50$
  - $d_1 = (50 \bmod 16) = 2 \quad n = 3$
  - $d_2 = (3 \bmod 16) = 3 \quad n = 0 \quad (815)_{10} = (32F)_{16}$
- Beispiel: Umwandeln von  $(133)_{10}$  ins Hexadezimalsystem
  - $n = 133$
  - $d_0 = (133 \bmod 16) = 5 \quad n = 8$
  - $d_1 = (8 \bmod 16) = 8 \quad n = 0 \quad (133)_{10} = (85)_{16}$

# Umwandeln aus dem Dezimalsystem

- Beispiel: Umwandeln von  $(133)_{10}$  ins Oktalsystem ( $b=8$ )

$$n = 133$$

$$d_0 = (133 \bmod 8) = 5$$

$$d_1 = (16 \bmod 8) = 0$$

$$d_2 = (2 \bmod 8) = 2$$

$$n = (133 \text{ div } 8) = 16$$

$$n = (16 \text{ div } 8) = 2$$

$$n = (2 \text{ div } 8) = 0$$

$$\begin{array}{r} (133)_{10} = (205)_8 \\ \hline (133)_{10} = (10000101)_2 \\ (133)_{10} = (85)_{16} \end{array}$$

## 2.1.5 Schnelleres Umwandeln

- **Bestimmte Kombinationen** von Zahlensystemen erlauben eine **schnellere Umwandlung**
- Betrachten wir wiederum die Zahl  $(133)_{10}$

$$(133)_{10} = (1000 \ 0101)_2 = (85)_{16}$$

dargestellt im Binärsystem und im Hexadezimalsystem

- **Korrespondenz** von Stellen im Hexadezimalsystem und Vierergruppen von Stellen im Binärsystem (z.B. 8 → 1000)

# Umwandeln vom Hexadezimal- ins Binärsystem

- Jede Stelle wird als Vierergruppe von Stellen (0 oder 1) dargestellt, führende Nullen werden entfernt

$$(\text{AFFE})_{16} = (1010\ 1111\ 1111\ 1110)_2$$

$$(\text{15FB})_{16} = (\underline{0001}\ 0101\ 1111\ 1011)_2$$

$$(314)_{16} = \boxed{(\underline{0011}\ 0001\ 0100)_2}$$

$$(\text{FEE})_{16} = \boxed{(\underline{1111}\ 1110\ 1110)_2}$$

# Umwandeln vom Binär- ins Hexadezimalsystem

- Füge führende Nullen ein bis die Stellenzahl ein  
**Vielfaches von Vier** ist, wandle Vierergruppen von Stellen in die entsprechende Ziffer im Hexadezimalsystem um

$$(1100\ 0101\ 0000)_2 = (\text{C}50)_{16}$$

0 (110 0000 1110)<sub>2</sub> = (60E)<sub>16</sub>

$$(1011\ 0001\ 1010\ 0011)_2 = (\text{B}\ \text{1}\ \text{A}\ \text{3})_{16}$$

ooo (1 1111 1011)<sub>2</sub> = (1 F B)<sub>16</sub>

# Wann ist eine schnellere Umwandlung möglich?

- Eine schnellere Umwandlung ist immer dann möglich, wenn für die Basen  $b_1$  und  $b_2$  der beiden Zahlensysteme folgendes gilt

$$b_2 = b_1^m$$

- Eine Stelle in der Darstellung zur Basis  $b_2$  entspricht dann immer einer Gruppe von  $m$  Ziffern in der Darstellung zur Basis  $b_1$

# Zusammenfassung

- Zahlen können zu anderer Basis als 10 dargestellt werden
- **Binärsystem** (mit Basis 2) und **Hexadezimalsystem** (mit Basis 16) spielen wichtige Rolle in der Informatik
- Umwandlung zwischen beliebigen Zahlensystem mittels **Dezimalsystem als Zwischendarstellung**
- **Schnellere Umwandlung** ist möglich für bestimmte Kombinationen von Zahlensystemen

# Literatur

- [1] **H.-P. Gumm und M. Sommer:** *Einführung in die Informatik*, Oldenbourg Verlag, 2012 (**Kapitel 1**)

$$n = 97$$

$b=2$  [d.h. Umwandlung ins Binärsystem)

$$\begin{array}{l} d_0 = (97 \bmod 2) = 1 \\ d_1 = (48 \bmod 2) = 0 \\ d_2 = (24 \bmod 2) = 0 \\ d_3 = (12 \bmod 2) = 0 \\ d_4 = (6 \bmod 2) = 0 \\ d_5 = (3 \bmod 2) = 1 \\ d_6 = (1 \bmod 2) = 1 \end{array} \quad \left| \begin{array}{l} n = (97 \text{ div } 2) = 48 \\ n = (48 \text{ div } 2) = 24 \\ n = (24 \text{ div } 2) = 12 \\ n = (12 \text{ div } 2) = 6 \\ n = (6 \text{ div } 2) = 3 \\ n = (3 \text{ div } 2) = 1 \\ n = (1 \text{ div } 2) = 0 \end{array} \right.$$

$$(97)_{10} = (1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 1)_2$$
$$\begin{matrix} & 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\ & 64 & 32 & 16 & 8 & 4 & 2 & 1 \end{matrix}$$

# Rückblick

- Zahlendarstellung zu einer beliebigen Basis  $b$

$$(214)_5 = 2 \cdot 5^2 + 1 \cdot 5^1 + 4 \cdot 5^0 \\ = (58)_{10}$$

- Umwandlung zwischen Zahlendarstellung

$$(278)_{10} = (?)_8$$

$$\begin{array}{l} n = 278 \\ d_0 = (278 \bmod 8) = 6 \\ d_1 = (34 \bmod 8) = 2 \\ d_2 = (4 \bmod 8) = 4 \end{array} \quad \begin{array}{l} n = 34 \\ n = 4 \\ n = 0 \end{array}$$

$$(278)_{10} = (426)_8$$

# Rückblick

- Schnellere Umwandlung zwischen Binärdarstellung und Hexadezimaldarstellung

$$(1FFE)_{16} = (?)_2$$

= ( 0001 1111 1111 1110 )<sub>2</sub>

$$(1100|1101)_2 = (?)_{16}$$

= (?)<sub>16</sub>

## 2.1.6 Arithmetik in den Zahlensystemen

- Aus der **Schulzeit bekannte Verfahren** zum „Rechnen von Hand“ funktionieren **unabhängig** vom verwendeten **Zahlensystem**
- Beispiel: Addition im Dezimalsystem

$$\begin{array}{r} \text{Summanden} \\ \hline & 1 & 4 & 1 & 6 & 7 \\ + & & & 2 & 7 & 6 \\ \hline & & & 1 & 1 & \\ \text{Überträge} & & & & & \\ \hline & = & 1 & 4 & 4 & 4 & 3 \\ \text{Ergebnis} & & & & & & \end{array}$$

# Arithmetik in den Zahlensystemen

- Beispiele: Addition im Binärsystem

$$\begin{array}{r} 1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \\ + \quad \quad \quad 1 \ 1 \ 0 \ 0 \ 1 \\ \hline 1 \quad \quad \quad \quad \quad 1 \\ \hline = \quad 1 \ 1 \ 0 \ 1 \ 1 \ 1 \ 0 \end{array}$$

$$\begin{array}{r} 1 \ 1 \ 1 \ 0 \ 1 \ 1 \ 1 \\ + \quad \quad \quad 1 \ 1 \ 0 \ 1 \ 1 \ 1 \\ \hline 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \\ \hline = 1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1 \ 0 \end{array}$$

# Arithmetik in den Zahlensystemen

- Beispiele: Addition im Hexadezimalsystem

$$\begin{array}{r} & F & A & B & 4 \\ + & & 6 & 8 & 6 \\ \hline & 1 & 1 & & \\ \hline = & 1 & 0 & 1 & 3 & A \\ \hline \end{array}$$

$$(30)_{10}$$

$n=30$   
 $d_0 = E$  ↑  $n=1$   
 $d_1 = 1$  ↑  $n=0$

$$\begin{array}{r} & F & F & B & A \\ + & E & 1 & D & F \\ \hline & 1 & 1 & 1 & 1 \\ \hline = & 1 & E & 1 & 9 & 9 \\ \hline \end{array}$$

$$\begin{aligned} A + F \\ = (25)_{10} \\ = (19)_{16} \end{aligned}$$

# Arithmetik in den Zahlensystemen

- Beispiel: Multiplikation im Dezimalsystem

Faktoren	3	1	6	.	2	1
	6	3	2			
	3	1	6			
<hr/>						
Überträge						
Ergebnis	=	6	6	3	6	

# Arithmetik in den Zahlensystemen

- Beispiel: Multiplikation im Binärsystem

$$\begin{array}{ccccccccc}
 1 & 0 & 0 & 1 & 0 & 1 & \cdot & 1 & 1 & 0 & 1 \\
 \hline
 1 & 0 & 0 & 1 & 0 & 1 & & & & & \\
 1 & 0 & 0 & 1 & 0 & 1 & & & & & \\
 0 & 0 & 0 & 0 & 0 & 0 & & & & & \\
 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
 \hline
 & 1 & 1 & 1 & 1 & & & & & \\
 \hline
 = & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\
 \hline
 2^8 & 2^7 & 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 & 
 \end{array}$$

$$(100101)_2 = (37)_{10}$$

$$2^8 = 256$$

$$(1101)_2 = (13)_{10}$$

$$2^7 = 128$$

$$(37)_{10} \cdot (13)_{10} = (481)_{10}$$

$$2^6 = 64$$

$$7^5 = 32$$

11

$$\begin{array}{r} \underline{-12} \\ 48 \end{array}$$

$$\begin{aligned}
 (100101)_2 &= 1 \cdot 2^5 + 1 \cdot 2^2 + 1 \cdot 2^0 \\
 d_5 \ d_4 \ d_3 \ d_2 \ d_1 \ d_0 &= 32 + 4 + 1 \\
 2^5 \ 2^4 \ 2^3 \ 2^2 \ 2^1 \ 2^0 &= (37)_{10}
 \end{aligned}$$

# Arithmetik in den Zahlensystemen

- Beispiel: Multiplikation im Binärsystem

$$\begin{array}{ccccccccc} 1 & 1 & 1 & 0 & 1 & 1 & \cdot & 1 & 1 \\ \hline 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ \hline \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline \\ = 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{array}$$

$$(100)_2 = (4)_{10}$$

# Arithmetik in den Zahlensystemen

- Beispiel: Multiplikation im Hexadezimalsystem

$$\begin{array}{r} & & A & 1 & \cdot & B & 2 \\ \hline (A1)_{16} \cdot (B)_{16} & & 6 & E & B & & \\ = (161)_{10} \cdot (11)_{10} & & & 1 & 4 & 2 & \\ \hline & & & & & & \\ & & = & 6 & F & F & 2 \\ \hline \end{array}$$

$$\begin{aligned} & (A1)_{16} \cdot (2)_{16} \\ & = (161)_{10} \cdot (2)_{10} \\ & = (322)_{10} \\ & = (142)_{16} \end{aligned}$$

# Arithmetik in den Zahlensystemen

- Beispiel: Multiplikation im Hexadezimalsystem

$$(A\bar{F})_{16} \cdot (\bar{C}\bar{F})_{16}$$

$$= (175)_{10} \cdot (15)_{10}$$

$$= (2625)_{10}$$

$$= (A41)_{16}$$

$$n = 2625$$

$$d_0 = (2625 \bmod 16) = 1 \quad n = 164$$

$$d_1 = (164 \bmod 16) = 4 \quad n = 10$$

$$d_2 = (10 \bmod 16) = A \quad n = 0$$

$$\begin{array}{r} & A & F & \cdot & F & E \\ \hline & A & 4 & 1 & & \\ & & 9 & 9 & 2 & \\ \hline & & & & & \\ & = & A & D & A & 2 \\ \hline \end{array}$$

$$\begin{aligned} & (A\bar{F})_{16} \cdot (\bar{E})_{16} \\ & = (175)_{10} \cdot (14)_{10} \\ & = (2450)_{10} \end{aligned}$$

$$n = 2450$$

$$\begin{aligned} d_0 & = (2450 \bmod 16) = 2 \\ n & = 153 \end{aligned}$$

$$d_1 = (153 \bmod 16) = 5 \quad n = 9$$

$$d_2 = (9 \bmod 16) = 9 \quad n = 0$$

## 2.1.7 Negative Zahlen

- Wie lassen sich auch **negative Zahlen** im vom Rechner verwendeten **Binärsystem** darstellen?
- Möglichkeit 1: Darstellung mit **Vorzeichenbit**

Wir verwenden eine feste Anzahl  $l$  von Bits und reservieren das **erste (höchstwertige) Bit**, um das Vorzeichen der Zahl und die verbleibenden  $(l - 1)$  Bits um den Betrag der Zahl darzustellen

Stehen uns L Bits zur Verfügung, so können wir  $2^L$  viele unterschiedliche "Dinge" (z.B. Zahlen) unterscheiden.

$$\underbrace{[011][011] \dots [011]}_{L \text{ Bits}} = 2^L$$

$$2 \cdot 2 \cdot \dots \cdot 2 = 2^L$$

Die Potenzmenge  $P(M)$  einer Menge  $M$  beinhaltet alle Teilmengen von  $M$ .

$$M = \{1, 5, 7\} \quad P(M) = \{\emptyset, \{1\}, \{5\}, \{7\}, \{1, 5\}, \{1, 7\}, \{5, 7\}, \{1, 5, 7\}\}$$

Allgemein gilt:  $|P(M)| = 2^{|M|}$

$$M = \{1, 5, 7\}$$

0	0	0	$\rightarrow \emptyset$
1	0	0	$\rightarrow \{1\}$
0	1	0	$\rightarrow \{5\}$
:			
1	1	1	$\rightarrow \{1, 5, 7\}$

Wir können Teilmengen von  $M$  als Folgen von  $|M|$  Bits darstellen

Für eine Menge  $M$  mit  $|M|=20$  gibt es also  $2^{20}$  viele Teilmengen

$$2^{20} = 2^{10} \cdot 2^{10} = 1024^2 \approx 1000000$$

# Negative Zahlen dargestellt mit Vorzeichenbit

- Beispiele: Bei Verwendung von  $l = 4$  Bits
  - die Zahl 7 wird dargestellt als [0|1|1|1]
  - die Zahl -7 wird dargestellt als [1|1|1|1]
- Ein Nachteil dieser Darstellung ist, dass die uns bekannten **arithmetischen Verfahren nicht länger funktionieren**
- Beispiel: Addition von 7 und 2 bei  $l = 4$  Bits
  - die Zahl 7 wird dargestellt als [0|1|1|1]
  - die Zahl 2 wird dargestellt als [0|0|1|0]

$$\begin{array}{r} 0111 \\ + 0010 \\ \hline 1001 \end{array}$$

wäre die Zahl -1

# Negative Zahlen dargestellt als Zweierkomplement

- Möglichkeit 2: Darstellung als **Zweierkomplement**

Wir verwenden weiterhin eine feste Anzahl von  $l$  Bits.

Eine negative Zahl  $-n$  wird als ihr Zweierkomplement  $(2^l - n)$  dargestellt. Das Zweierkomplement einer Zahl  $n$  ist also ihre Differenz zur Zweierpotenz  $2^l$

- Beispiele: Verwendung von  $l = 4$  Bits

- $(-5)_{10} = (0101)_2$  wird dargestellt als  $2^4 - 5 = 16 - 5 = (1011)_2$
- $(-3)_{10} = (0011)_2$  wird dargestellt als  $2^4 - 3 = 16 - 3 = (1101)_2$

# Negative Zahlen dargestellt als Zweierkomplement

- Die bekannten **arithmetischen Verfahren** funktionieren bei der Darstellung als Zweierkomplement wie gewohnt

$$(6)_{10} + (-3)_{10} = (0110)_2 + (1101)_2 = (0011)_2 = (3)_{10}$$

- Einfachere Berechnung** des Zweierkomplements als

$$(2^l - n) = (2^l - 1 - n) + 1$$

- $(2^l - 1)$  ist die Binärzahl bestehend aus  $l$  Einsen
- $-n$  (mit bekanntem Verfahren zur Subtraktion)
- $+1$  (mit bekanntem Verfahren zur Addition)

A binary addition diagram showing the calculation of  $0110 + 1101$ . The result is  $10011$ , where the most significant bit (MSB) is 1, indicating a negative number. A blue bracket groups the first two terms, and a blue curly brace groups all three terms:  $0110 + 1101 + 1$ .

$$\begin{array}{r} 0110 \\ + 1101 \\ \hline 10011 \end{array}$$

# Negative Zahlen dargestellt als Zweierkomplement

- Dies entspricht der (noch) **einfacheren Vorgehensweise**
  - Bestimme  $(2^l - 1 - n)$  durch **Invertieren** der Bits der Binärdarstellung der Zahl  $n$
  - **Addiere** 1
- Beispiel: Zweierkomplement von  $n = 6$  bei  $l = 4$  Bits
  - Binärdarstellung von 6 ist  $(0110)_2$
  - Invertieren der Bits ergibt  $(1001)_2$
  - Addieren von 1 ergibt  $(1010)_2 = (10)_{10}$

$$L = 16$$

$$n = 131$$

1. Binärdarstellung von 131 mit 16 Stellen

$$\begin{array}{cccccccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ & & & & & & & & 2^7 & & & & & & 2^4 & 2^0 \end{array}$$

2. Invertiere alle Bits

$$1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 0 \ 1 \ 1 \ 1 \ 1 \ 1 \ 0 \ 0$$

3. Addier 1

$$1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 0 \ 1 \ 1 \ 1 \ 1 \ 1 \ 0 \ 1$$

# Negative Zahlen dargestellt als Zweierkomplement

- Zweierkomplement bei Verwendung von  $l = 4$  Bits

Binärdarstellung	Bitfolge	Zweierkomplement	
7	0111	Positive Zahlen	7
6	0110		6
5	0101		5
4	0100		4
3	0011		3
2	0010		2
1	0001		1
0	0000		0
15	1111	Negative Zahlen	-1
14	1110		-2
13	1101		-3
12	1100		-4
11	1011		-5
10	1010		-6
9	1001		-7
8	1000		-8

“Flippen“  
der Bits

↑ +1  
←

```
int n = 0;  
while (true) {  
    n = n + 1;  
    System.out.println(n);  
}
```

Irgendwann werden negative Zahlen ausgegeben

# Negative Zahlen dargestellt als Zweierkomplement

- Beispiel: Zweierkomplement von  $n = 67$  bei  $l = 8$  Bits

- Binärdarstellung von 67 ist

0 1 0 0 0 0 1 1

- Invertieren der Bits ergibt

1 0 1 1 1 1 0 0

- Addieren von 1 ergibt

1 0 1 1 1 1 0 1

$2^7 \ 2^6 \ 2^5 \ 2^4 \ 2^3 \ 2^2 \ 2^1 \ 2^0$

$$128 + 32 + 16 + 8 + 4 + 1 = 189$$

$$(2^8 - 67) = (256 - 67) = 189$$

- Bei der **Subtraktion** mittels Zweierkomplement müssen beide Zahlen mit **gleicher Stellenanzahl** betrachtet werden

# Negative Zahlen dargestellt als Zweierkomplement

- **Darstellbare Zahlenbereiche** bei Verwendung von  $l$  Bits
  - $l = 8$  :  $-2^7 \leq n < 2^7$
  - $l = 16$  :  $-2^{15} \leq n < 2^{15}$  (entspricht `short` in Java)
  - $l = 32$  :  $-2^{31} \leq n < 2^{31}$  (entspricht `int` in Java)
  - $l = 64$  :  $-2^{63} \leq n < 2^{63}$  (entspricht `long` in Java)
  - $l$  :  $-2^{(l - 1)} \leq n < 2^{(l - 1)}$
- Einige Programmiersprachen (z.B. C) unterscheiden **Datentypen mit/ohne Vorzeichen**. Der darstellbare Zahlenbereich verdoppelt sich in diesem Fall
  - $l = 16$  :  $0 \leq n < 2^{16}$  (`unsigned int` in C)

# Zusammenfassung

- **Addition** und **Subtraktion** in der b-adischen Darstellung funktionieren mit den **bekannten Vorgehensweisen**
- **Negative Zahlen** lassen sich mittels **Vorzeichenbit** oder als **Zweierkomplement** darstellen

# Literatur

- [1] **H.-P. Gumm und M. Sommer:** *Einführung in die Informatik*, Oldenbourg Verlag, 2012 (**Kapitel 1**)

# Rückblick

- **Addition** in der b-adischen Darstellung **wie gewohnt**

$$\begin{array}{r} & 5 & 0 & C & E \\ + & D & 4 & 2 & D \\ \hline & 1 & & & 1 \\ = & 1 & 2 & 4 & F \quad B \\ \hline \end{array}$$

$$(F6E)_{16} = (1111\ 0110\ 1110)_2$$

$$n = 27$$

$$d_0 = (27 \bmod 16) = 3 \uparrow n=1$$
$$d_1 = (1 \bmod 16) = 1 \downarrow n=0$$

$$(27)_{10} = (13)_{16}$$

# Rückblick

- **Multiplikation in der b-adischen Darstellung wie gewohnt**

$$\begin{array}{r} \overline{1 \ 0 \ 1 \ 0 \ . \ 1 \ 0 \ 1} \\ \times \overline{1 \ 0 \ 1 \ 0} \\ \hline \\ \overline{1 \ 0 \ 1 \ 0} \\ \hline \\ \overline{1 \ 1 \ 0 \ 0 \ 1 \ 0} \end{array}$$

# Rückblick

- Darstellung negativer Zahlen mit **Vorzeichenbit**
  - Bei Verwendung von  $l = 8$  Bits wird -9 dargestellt als

[ 1 1 0 1 0 1 0 1 1 1 0 1 0 1 1 ]



Vorzeichenbit zeigt uns an, dass wir es mit einer negativen Zahl zu tun haben

- Darstellung negativer Zahlen als **Zweierkomplement**
  - Bei Verwendung von  $l = 8$  Bits wird -9 dargestellt als

[ 1 1 1 1 1 1 0 1 1 1 1 ]

$$2^8 - 9 = 256 - 9 = 247$$

[ 0 1 0 1 0 1 0 1 1 0 1 0 1 1 ] Binärdarstellung von 5

[ 1 1 1 1 1 1 1 0 1 1 1 1 0 ] Invertiert Bits

[ 1 1 1 1 1 1 1 0 1 1 1 1 ] +1

$$2^7 \ 2^6 \ 2^5 \ 2^4 \ 2^3 \ 2^2 \ 2^1 \ 2^0$$

$$128 + 64 + 32 + 16 + 4 + 2 + 1 = 247$$

## 2.1.8 Kommazahlen

- Wie lassen sich **Kommazahlen** im Rechner darstellen?
- Möglichkeit 1: Erweiterung der  $b$ -adischen Darstellung  
Wir stellen den **ganzzahligen Anteil** wie gehabt mit  $l$  Bits dar; die **Stellen hinter dem Komma** stellen wir separat mit  $m$  Bits dar

Eine solche Zahl  $(d_{l-1} \dots d_0, d_{-1} \dots d_{-m})$  entspricht dann

$$(d_{l-1} \dots d_0, d_{-1} \dots d_{-m}) = \left( \sum_{i=0}^{l-1} d_i \cdot b^i \right) + \left( \sum_{i=1}^m d_{-i} \cdot b^{-i} \right)$$

$3,14$   
 d<sub>0</sub>    d<sub>-1</sub>    d<sub>-2</sub>  
 $10^0$       |  
 ist 3 mal  
 enthalten

$10^{-1}$   
 ist 1 mal  
 enthalten

$10^{-2}$   
 ist 4 mal  
 enthalten

$$3 \cdot 10^0 + 1 \cdot 10^{-1} + 4 \cdot 10^{-2}$$

$$= 3 + 0,1 + 0,04$$

$$= 3,14$$

# Kommazahlen in b-adischer Darstellung

- Beispiel: 2,375 dargestellt mit  $l = 4$  und  $m = 4$  Bits

$$(2,375)_{10} = (0010, 0110)_2$$

$2^{-4} 2^{-3} 2^{-2} 2^{-1}$

denn

$$0,375 = 0 \cdot \frac{1}{2} + 1 \cdot \frac{1}{4} + 1 \cdot \frac{1}{8} + 0 \cdot \frac{1}{16}$$

- Beispiel: 5,75 dargestellt mit  $l = 4$  und  $m = 4$  Bits

$$(5,75)_{10} = (0101, 1100)_2$$

# Kommazahlen in b-adischer Darstellung

- Berechnungsvorschrift zur Umwandlung in die b-adische Darstellung kann wie folgt **erweitert werden**:
  - Wandle **ganzzahligen Anteil** wie bekannt um
  - Es sei  $r$  der **Anteil (Rest) nach dem Komma**
    - $i \leftarrow 1$
    - **solange** gilt, dass  $r > 0$  ist
      - **wenn**  $2^{-i} \leq r$ 
        - $d_{-i} \leftarrow 1$
        - $r \leftarrow r - 2^{-i}$
      - **sonst**
        - $d_{-i} \leftarrow 0$
    - $i \leftarrow i + 1$

# Kommazahlen in b-adischer Darstellung

- Beispiel: 2,375 dargestellt mit  $l = 4$  und  $m = 4$  Bits
  - ganzzahliger Anteil  $(2)_{10} = (0010)_2$
  - $r = 0,375$
  - $r = 0,375 < 2^{-1}$ , daher  $d_{-1} = 0$
  - $r = 0,375 \geq 2^{-2}$ , daher  $d_{-2} = 1$  und neuer Rest  $r = 0,125$
  - $r = 0,125 \geq 2^{-3}$ , daher  $d_{-3} = 1$  und neuer Rest  $r = 0$
- Ergebnis:  $(2,375)_{10} = (0010,0110)_2$



von oben nach unten los

Die Berechnung stoppt, bevor die Stelle  $d_{-4}$  betrachtet wird.

Stellen, die nicht betrachtet wurden, werden als 0 angenommen.

# Kommazahlen in b-adischer Darstellung

- Beispiel: 4,5625 dargestellt mit  $l = 4$  und  $m = 4$  Bits

Ganzzahliger Anteil : 4

$(0100)_2$

$$r = 0,5625$$

$$r = 0,5625 \geq 2^{-1} = 0,5, \text{ daher } d_{-1} = 1 \quad \text{und} \quad r = 0,0625$$

$$r = 0,0625 < 2^{-2} = 0,25, \text{ daher } d_{-2} = 0 \quad \text{und} \quad r = 0,0625$$

$$r = 0,0625 < 2^{-3} = 0,125, \text{ daher } d_{-3} = 0 \quad \text{und} \quad r = 0,0625$$

$$r = 0,0625 \geq 2^{-4} = 0,0625, \text{ daher } d_{-4} = 1 \quad \text{und} \quad r = 0$$

$(0100, 1001)_2$

# Kommazahlen in b-adischer Darstellung

- Beispiel: 0,3 dargestellt mit  $l = 4$  und  $m = 4$  Bits
  - ganzzahliger Anteil  $(0)_{10} = (0000)_2$
  - $r = 0,3$
  - $r = 0,3 < 2^{-1}$ , daher  $d_{-1} = 0$
  - $r = 0,3 \geq 2^{-2}$ , daher  $d_{-2} = 1$  und neuer Rest  $r = 0,05$
  - $r = 0,05 < 2^{-3}$ , daher  $d_{-3} = 0$  und neuer Rest  $r = 0,05$
  - $r = 0,05 < 2^{-4}$ , daher  $d_{-4} = 0$  und neuer Rest  $r = 0,05$
- Die Zahl 0,3 lässt sich also mit  $m = 4$  Stellen hinter dem Komma **nicht exakt** darstellen

# Kommazahlen in b-adischer Darstellung

- Die Zahl  $0,3$  lässt sich also mit  $m = 4$  Stellen hinter dem Komma **nicht exakt** darstellen
- Selbst bei **beliebiger Stellenanzahl** m kann  $0,3$  **nicht exakt** dargestellt werden, da  $3/10$  nicht als Summe von Potenzen der Form  $1/2^i$  dargestellt werden kann
- Bei genauerer Betrachtung ergibt sich ein „**periodischer**“ Dualbruch

# IEEE Gleitkommazahlendarstellung

- Möglichkeit 2: Kommazahlen werden im Rechner als **Gleitkommazahlen** (auch: Fließkommazahlen) dargestellt
- Im Englischen heißen diese *floating point numbers*, was auf die dort gängige Zahlendarstellung hindeutet
  - der **Punkt** wird anstelle des Kommas verwendet
  - das **Komma** dient als **Tausendertrennzeichen**
- Beispiele:
  - 2,375 wird geschrieben als 2.375
  - 1.000.000 wird geschrieben als 1,000,000

# IEEE Gleitkommazahlendarstellung

- Weit verbreitet ist der **IEEE Floating Point Standard 754** des Institute of Electrical and Electronics Engineering
- Grundidee hierbei ist, **keine feste Anzahl** von Bits vor und hinter dem **Komma** zu reservieren, sondern dieses **wandern** zu lassen

# IEEE Gleitkommazahlendarstellung

- Eine Zahl wird dargestellt als

$$n = (-1)^s \cdot (1 + m) \cdot 2^{(e-b)}$$

mit den Bestandteilen

- **Vorzeichen**  $s \in \{0, 1\}$
- **Mantisse**  $m \in [0,0; 1,0]$  in Binärdarstellung
- **Exponent**  $e \in [0, 2^l)$  in Binärdarstellung, von dem ein festgelegter Wert  $b$  (*bias*) abgezogen wird
- Die **Mantisse** stellt die Zahlen aus dem Intervall  $[0,0; 1,0]$ , für die dies bei gegebener Bitanzahl möglich ist, exakt und andere näherungsweise dar

# IEEE Gleitkommazahlendarstellung

- Beispiel: Die Zahl 0,6640625 lässt sich bei **ausreichender Bitanzahl** exakt darstellen als

$$0,6640625 = \frac{1}{2} + \frac{1}{8} + \frac{1}{32} + \frac{1}{128}$$

- Die Zahl 0,6640626 hingegen ist **nicht exakt darstellbar** und würde bei Berechnungen durch  
0,66406261920928955078125  
**näherungsweise** dargestellt werden

# IEEE Gleitkommazahlendarstellung

- Java verwendet die IEEE Gleitkommazahlendarstellung (binary32 und binary64) für seine primitiven Datentypen

- `float`: (23 Bit Mantisse, 8 Bit Exponent, 1 Bit Vorzeichen) zur Darstellung von Zahlen im Bereich

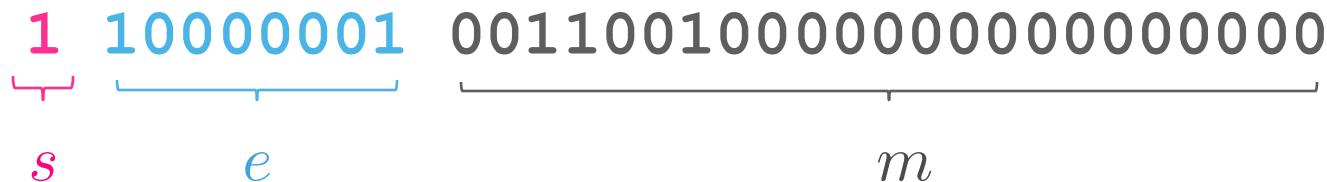
$$-2^{127} \approx -1,7 \cdot 10^{38} \quad \text{bis} \quad 2^{127} \approx 1,7 \cdot 10^{38}$$

- `double`: (52 Bit Mantisse, 11 Bit Exponent, 1 Bit Vorzeichen) zur Darstellung von Zahlen im Bereich

$$-2^{1023} \approx -1,8 \cdot 10^{308} \quad \text{bis} \quad 2^{1023} \approx 1,8 \cdot 10^{308}$$

# IEEE Gleitkommazahlendarstellung

- Beispiel: Bei Verwendung von 32 Bits stellen folgende Bits

  
The binary number is 1 10000001 001100100000000000000000. A pink bracket under the first bit is labeled 's' (sign). A blue bracket under the next 7 bits is labeled 'e' (exponent). A long grey bracket under the remaining 23 bits is labeled 'm' (fraction).

die Zahl -4,78125 exakt dar

$$\begin{aligned} & (-1)^1 \cdot 2^{(129-127)} \cdot \left(1 + \frac{1}{8} + \frac{1}{16} + \frac{1}{128}\right) \\ &= -1 \cdot 2^2 \cdot (1,1953125) \end{aligned}$$

- Gleitkommazahlenrechner:

<https://www.h-schmidt.net/FloatConverter/IEEE754de.html>

## 2.2 Darstellung von Zeichen

- Zur Darstellung von Zeichen werden **Zeichensatztabellen** (*code pages*) verwendet, die jedem **Zeichen** seine **Darstellung zuordnen**, z.B.:
  - **ASCII** (American Standard Code for Information Interchange) stellt jedes Zeichen in 7 Bits dar, z.B. a → (97)<sub>10</sub> und z → (122)<sub>10</sub>, keine Umlaute, nur 128 Zeichen
  - **ASCII + 8. Bit** (viele kompatible Zeichensatztabellen)
    - ISO-8859-1: Westeuropäisch, ASCII + 8.Bit, z.B. ä → (228)<sub>10</sub>
    - ISO-8859-2: Slavische Sprachen
    - ...
    - ISO-8859-15: Informationstechnologie (deckt gängigste moderne Sprachen und Sonderzeichen ab), z.B. € → (164)<sub>10</sub>

# ASCII Zeichensatztabelle

## ■ ISO-8859-1 Zeichensatztabelle

Codepage 819 - Latin 1 - ISO 8859-1																
-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F	
0-	0001	0002	0003	0004	0005	0006	0007	0008	0009	000A	000B	000C	000D	000E	000F	
1-	0010	0011	0012	0013	0014	0015	0016	0017	0018	0019	001A	001B	001C	001D	001E	001F
2-	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/	002F
3-	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4-	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5-	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	^	-	005F
6-	~	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7-	p	q	r	s	t	u	v	w	x	y	z	{		}	~	007F
8-	0080	0081	0082	0083	0084	0085	0086	0087	0088	0089	008A	008B	008C	008D	008E	008F
9-	0090	0091	0092	0093	0094	0095	0096	0097	0098	0099	009A	009B	009C	009D	009E	009F
A-	00A0	í	¢	£	¤	¥	ı	§	“	©	¤	«	¬	-	®	-
B-	°	±	2	3	‘	µ	¶	·	„	1	º	»	¼	½	¾	¸
C-	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D-	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	Þ
E-	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F-	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

# Unicode

- Mit 8 Bits können wir nur  $2^8 = 256$  Zeichen darstellen, was nicht für einen **universellen Zeichensatz**, der alle Sprachen der Welt abdeckt, reicht
- **Unicode** möchte solch ein universeller Zeichensatz sein; man unterscheidet verschiedene Varianten:
  - **UTF-16** verwendet immer 2 Bytes (also 16 Bits) pro Zeichen, so dass insgesamt  $2^{16} = 65,536$  Zeichen darstellbar sind; der **Speicherplatz** wächst jedoch um einen **Faktor 2**

# Unicode

- **UTF-8** verwendet eine **flexible Anzahl von Bytes** und stellt diese Anzahl mittels Unärdarstellung als Präfix dar

■ <u>[0   ...]</u>	1 Byte (ASCII)	Über: n wird dargestellt durch n Einsen
■ <u>[1   0   ...]    [ ... ]</u>	2 Bytes	gefolgt von einer Null
■ <u>[1   1   0   ...]    [ ... ]</u>	3 Bytes	1110 stellt 3 dar
■ <u>[1   1   1   0   ...]    [ ... ]</u>	4 Bytes	110 stellt 2 dar

— zeigt an, wie viele Bytes vom gleichen Zeichen noch folgen

- **Manche Dateiformate** (z.B. HTML) **geben** den zugehörigen **Zeichensatz** (z.B. ISO-8859-1) **an**, andernfalls (z.B. bei reinen Textdateien) hilft nur Raten

[0| ...] [1|0| ...] [...|] [1|1|0| ...] [...] [...] [0|...]  
1. Zeichen 2. Zeichen 3. Zeichen 4. Zeichen

# Zusammenfassung

- **Kommazahlen** lassen sich durch Erweiterung der **b-adischen Darstellung** darstellen
- **IEEE Floating Point Standard 754** als weit verbreitete Darstellung für Gleitkommazahlen
- **Zeichendarstellung** mittels Zeichensatztabellen wie z.B. ASCII, ISO-8859-1, ..., ISO-8859-15, UTF-8, UTF-16

# Literatur

- [1] **H.-P. Gumm und M. Sommer:** *Einführung in die Informatik*, Oldenbourg Verlag, 2012 (**Kapitel 1**)

# Rückblick

## ■ Zweierkomplement bei Verwendung von $l = 4$ Bits

Binärdarstellung	Bitfolge	Zweierkomplement	
7	0111	Positive Zahlen	7
6	0110		6
5	0101		5
4	0100		4
3	0011		3
2	0010		2
1	0001		1
0	0000		0
15	1111	Negative Zahlen	-1
14	1110		-2
13	1101		-3
12	1100		-4
11	1011		-5
10	1010		-6
9	1001		-7
8	1000		-8

“Flippen“  
der Bits

↑ +1  
←

# Rückblick

- Erweiterte b-adische Darstellung von **Kommazahlen**
  - 7,1875 dargestellt mit  $l = 4$  und  $m = 4$  Bits

ganzzahliger Anteil: 7     $(0111)_2$      $(d_{l+1} \dots d_2 d_1 d_0, d_{-1} d_{-2} \dots d_{-m})$

$$r = 0,1875$$

$$r < 2^{-1} = 0,5, \text{ daher } d_{-1} = 0, r = 0,1875$$

$$r < 2^{-2} = 0,25, \text{ daher } d_{-2} = 0, r = 0,1875$$

$$r \geq 2^{-3} = 0,125, \text{ daher } d_{-3} = 1, r = 0,0625$$

$$r \geq 2^{-4} = 0,0625, \text{ daher } d_{-4} = 1, r = 0$$

$$(0\ 111, \ 0011)$$

$$\begin{array}{c} 7 \\ \swarrow \\ l \text{ Bits} \end{array} \quad \begin{array}{c} 1875 \\ \swarrow \\ m \text{ Bits} \end{array}$$

# Rückblick

- **Gleitkommazahlen** (IEEE Floating Point Standard 754) lassen das Komma bei der Darstellung wandern
- Zeichen werden mit Hilfe von **Zeichensatztabellen** (z.B. ASCII, ISO-8859-X, UTF-8, UTF-16) dargestellt

## 2.3 Boolesche Algebra

- **Definition:** Eine **Boolesche Algebra** ist eine Menge  $X$  mit Operationen  $+$ ,  $\cdot$  und  $\neg$ , die als „**oder**“, „**und**“ und „**nicht**“ bezeichnet werden und für die folgende Regeln gelten

### I. Assoziativität

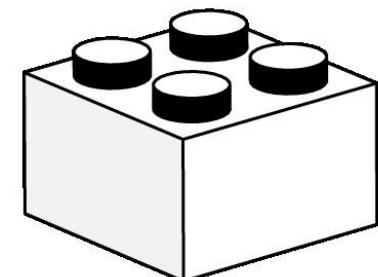
$$\forall a, b, c \in X : (a + b) + c = a + (b + c)$$

$$\forall a, b, c \in X : (a \cdot b) \cdot c = a \cdot (b \cdot c)$$

### II. Kommutativität

$$\forall a, b \in X : (a + b) = (b + a)$$

$$\forall a, b \in X : (a \cdot b) = (b \cdot a)$$



# Boolesche Algebra

## III. Distributivitat

$$\forall a, b, c \in X : a + (b \cdot c) = (a + b) \cdot (a + c)$$

$$\forall a, b, c \in X : a \cdot (b + c) = (a \cdot b) + (a \cdot c)$$

## IV. Absorptionsgesetz

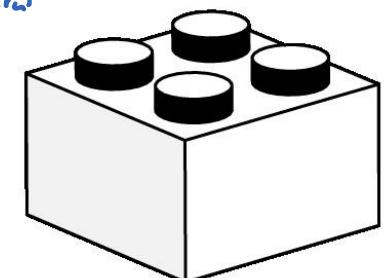
$$\forall a, b \in X : a + (a \cdot b) = a$$

$$\forall a, b \in X : a \cdot (a + b) = a$$

## V. Inverse Elemente

$$\forall a \in X : a + \bar{a} = 1 \quad \text{1-Element}$$

$$\forall a \in X : a \cdot \bar{a} = 0 \quad \text{0-Element}$$



# Potenzmenge als Boolesche Algebra

- Beispiel 1: Die **Potenzmenge**  $P(M)$  einer beliebigen Menge  $M$  bildet mit den Operationen  $\cap$  (Schnitt),  $\cup$  (Vereinigung) und  $\neg$  (Mengenkomplement)  
nicht      und      oder
- Betrachten wir die Menge  $M = \{0, 1, 2\}$ , dann hat ihre Potenzmenge die folgenden Elemente

$$P(M) = \left\{ \begin{array}{c} \emptyset, \\ \{0\}, \{1\}, \{2\}, \\ \{0, 1\}, \{0, 2\}, \{1, 2\}, \\ \{0, 1, 2\} \end{array} \right\}$$



# Potenzmenge als Boolesche Algebra

- Betrachten wir die Menge  $M = \{1, 3, 5, 7\}$ , dann hat ihre Potenzmenge die folgenden Elemente

$$P(M) = \left\{ \emptyset, \{1\}, \{3\}, \{5\}, \{7\}, \{1, 3\}, \{1, 5\}, \{1, 7\}, \{3, 5\}, \{3, 7\}, \{5, 7\}, \{1, 3, 5\}, \{1, 3, 7\}, \{1, 5, 7\}, \{3, 5, 7\}, \{1, 3, 5, 7\} \right\}$$

$n! = \prod_{i=1}^n i$   
Faktoriell  $n!$  gibt uns die Anzahl von Anordnungen von  $n$  Elementen an

$$|P(M)| = 2^{|M|}$$

hier:  $|P(M)| = 16 = 2^4 = 2^{|M|}$

$\vdots$	$\overbrace{\quad \quad \quad \quad}$	$\{1, 3, 5, 7\}$	$2^4$
$\vdots$	$\begin{matrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \vdots & & & \\ 1 & 0 & 1 & 0 \\ \vdots & & & \\ 1 & 1 & 1 & 1 \end{matrix}$	$\emptyset$	$\text{Belegungen von 4 Bits}$
$\vdots$	$\{7\}$	$\{1, 5\}$	
$\vdots$	$\{1, 3, 5, 7\}$		

## V. Inverse Elemente

$$\forall a \in X: a + \bar{a} = 1$$

$$\forall a \in P(M): a \cup \bar{a} = M$$

1-Element ist die ursprüngliche Menge M

Mengenkomplement  $\bar{a} = M \setminus a$

Beispiel:  $\overline{\{1, 3\}} = \{1, 3, 5, 7\} \setminus \{1, 3\} = \{5, 7\}$

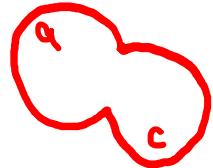
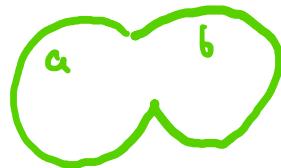
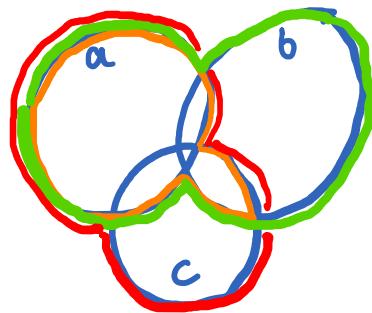
$$\overline{\{1, 3, 5, 7\}} = \{1, 3, 5, 7\} \setminus \{1, 3, 5, 7\} = \emptyset$$

$$\forall a \in X: a \cdot \bar{a} = 0$$

$$\forall a \in P(M): a \cap \bar{a} = \emptyset$$

0-Element ist die leere Menge  $\emptyset$

## Venn-Diagramm



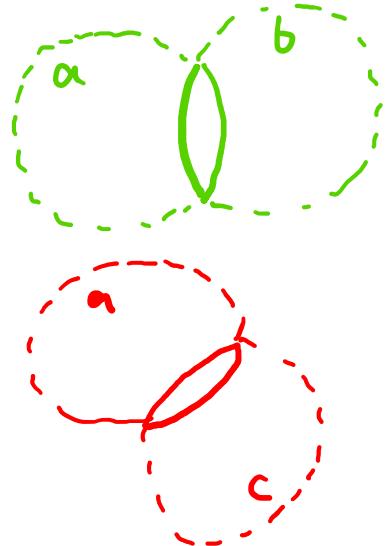
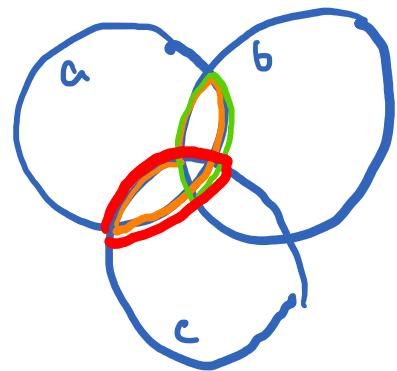
Distributivität

$$\forall a, b, c \in X : a + (b \cdot c) = (a + b) \cdot (a + c)$$

■  $a \cup (b \cap c)$

■  $a \cup b$

■  $a \cup c$



Distributivgesetz:

$$\forall a, b, c \in X : a \cdot (b + c) = (a \cdot b) + (a \cdot c)$$

- $a \cap (b \cup c)$
- $a \cap b$
- $a \cap c$

# Potenzmenge als Boolesche Algebra

- Beispiel 2: Die Menge der **Booleschen Werte**  $\mathbb{B} = \{0, 1\}$  bildet mit den Operationen  $\vee$  (oder),  $\wedge$  (und) und  $\neg$  (nicht) eine Boolesche Algebra, welche Grundlage der Aussagenlogik ist

$\vee$	
0 0	0
0 1	1
1 0	1
1 1	1

$\wedge$	
0 0	0
0 1	0
1 0	0
1 1	1

$\neg$	
0	1
1	0



## 2.4 Aussagenlogik

- Auf der Menge der **Booleschen Werte**  $\mathbb{B} = \{0, 1\}$  mit den Operationen  $\vee$  (oder),  $\wedge$  (und) und  $\neg$  (nicht) baut die sogenannte **Aussagenlogik** (vgl. Mathematik 1) auf
- Sie betrachtet **Aussagen** mit **eindeutigen Wahrheitswert**
  - 11 ist eine Primzahl
  - 16 ist ganzzahlig durch 3 teilbar
- **Aussagen** können miteinander verknüpft werden, um so **zusammengesetzte Aussagen** zu bilden

# Aussagenlogik

- Neben den bereits bekannten Booleschen Operationen  $\vee$ ,  $\wedge$  und  $\neg$  gibt es weitere Verknüpfungen z.B.
  - **XOR** („entweder oder“)

$\oplus$	
0	0
0	1
1	0
1	1

- **Implikation** („folgt aus“, „wenn...dann“)

$\Rightarrow$	
0	0
0	1
1	0
1	1

# Aussagenlogik

- Äquivalenz („genau dann wenn“)

		$\Leftrightarrow$
0	0	1
0	1	0
1	0	0
1	1	1

# Logische Variablen

- Definition: Wir nennen ein  $a \in \{0,1\}$  eine **logische Variable**
- Definition: Eine Zuweisung von Werten zu logischen Variablen nennen wir **Belegung**

# Logische Ausdrücke

- Definition: Eine Verknüpfung von endlich vielen logischen Konstanten (0 und 1) und logischen Variablen heißt **logischer Ausdruck**. Die Reihenfolge, in der Verknüpfungen anzuwenden sind, wird durch Klammern bestimmt. Ausdrücke, die nur die Verknüpfungen  $\vee$ ,  $\wedge$  und  $\neg$  enthalten, heißen **Boolesche Ausdrücke**.
- Um Klammern zu sparen, gelten **Vorrangregeln**
  - $\neg$  bindet am stärksten
  - $\wedge$  bindet stärker als  $\vee$

$\neg a \vee b \wedge c$  ist implizit geklammert als  
 $((\neg a) \vee (b \wedge c))$

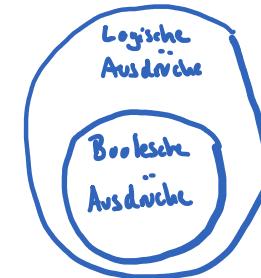
Wir können Klammern setzen, um von der Vorrangreihenfolge abzuweichen

$\neg(a \vee b) \wedge c$

$\neg a \wedge b \vee \neg c \wedge d$  ist implizit geklammert als  
 $((\neg a) \wedge b) \vee ((\neg c) \wedge d)$

# Logische Äquivalenz

- Beispiele:
  - $(a \wedge b) \Rightarrow c$  ist ein logischer Ausdruck
  - $\neg a \vee (b \wedge c)$  ist ein Boolescher Ausdruck
- Definition: Zwei logische Ausdrücke  $A_1$  und  $A_2$  heißen **äquivalent** (in Zeichen “ $=$ ”), wenn sie bei gleicher Belegung **gemeinsamer Variablen** stets den gleichen Wahrheitswert liefern
- $A_1 = A_2$  heißt dann **logische Gleichung**



# Logische Äquivalenz mittels Wertetabellen

- In der Vorlesung Mathematik 1 wurde gezeigt, wie man die **logische Äquivalenz** zweier Ausdrücke  $A_1$  und  $A_2$  mittels ihrer **Wertetabellen** überprüfen kann
- Beispiel:**  $A_1 = (a \vee \neg b) \Rightarrow c$  und  $A_2 = (\neg a \wedge b) \vee c$

	$a$	$b$	$c$	$(a \vee \neg b)$	$(a \vee \neg b) \Rightarrow c$		$a$	$b$	$c$	$(\neg a \wedge b)$	$(\neg a \wedge b) \vee c$
0	0	0	0	1	0		0	0	0	0	0
1	0	0	1	1	1		0	0	1	0	1
2	0	1	0	0	1		0	1	0	1	1
3	0	1	1	0	1		0	1	1	1	1
4	1	0	0	1	0		1	0	0	0	0
5	1	0	1	1	1		1	0	1	0	1
6	1	1	0	1	0		1	1	0	0	0
7	1	1	1	1	1		1	1	1	0	1

# Logische Äquivalenz mittels Wertetabellen

- Beispiel:  $A_1 = (a \Rightarrow b)$  und  $A_2 = (\neg a \vee b)$

$a$	$b$	$(a \Rightarrow b)$	$(\neg a \vee b)$
0	0	1	1
0	1	1	1
1	0	0	0
1	1	1	1

Sind  
äquivalent!

- Beispiel:  $A_1 = \neg(a \Rightarrow b)$  und  $A_2 = (a \wedge \neg b)$

$a$	$b$	$\neg(a \Rightarrow b)$	$(a \wedge \neg b)$
0	0	0	0
0	1	0	0
1	0	1	1
1	1	0	0

Sind  
äquivalent!

Sind die beiden Ausdrücke

$$A_1 = \neg a \wedge (b \vee \neg c)$$

und

$$A_2 = a \vee \neg(b \wedge c) \vee \neg d \wedge (e \vee f)$$

äquivalent?

a	b	c	$\neg a \wedge (b \vee \neg c)$	$a \vee \neg(b \wedge c) \vee \neg d \wedge (e \vee f)$
0	0	0	1	1
0	0	1	0	1
0	1	0	.	.
0	1	1	.	.
1	0	0		
1	0	1		
1	1	0		
1	1	1		

1  
1 → Die beiden Ausdrücke  
· sind nicht äquivalent  
:  
(In Klausur bitte Wertetabelle vollständig  
ausfüllen!)

# Logische Äquivalenz mittels Umformen

- Eine zweite Methode zum Zeigen der logischen Äquivalenz zweier Ausdrücke  $A_1$  und  $A_2$  besteht darin, einen **anhand von Rechenregeln** in den anderen **umzuformen**
- **Rechenregeln** für logische Ausdrücke (vgl. Handout):
  - I. Negation der Negation

$$\neg(\neg a) = a$$

## II. Kommutativ-Gesetze

$$a \wedge b = b \wedge a$$

$$a \vee b = b \vee a$$

# Logische Äquivalenz mittels Umformen

## III. Assoziativ-Gesetze

$$(a \wedge b) \wedge c = a \wedge (b \wedge c)$$

$$(a \vee b) \vee c = a \vee (b \vee c)$$

## IV. Distributiv-Gesetze

$$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$$

$$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$$

## V. Idempotenz-Gesetze

$$a \wedge a = a$$

$$a \vee a = a$$

# Logische Äquivalenz mittels Umformen

## VI. Komplement-Gesetze

$$a \wedge \neg a = 0$$

$$a \vee \neg a = 1$$

## VII. 0-1-Gesetze

$$a \wedge 0 = 0$$

$$a \wedge 1 = a$$

$$a \vee 0 = a$$

$$a \vee 1 = 1$$

# Logische Äquivalenz mittels Umformen

## VIII. Absorptions-Gesetze

$$a \wedge (a \vee b) = a$$

$$a \vee (a \wedge b) = a$$

$$(a \wedge b) \vee (a \wedge \neg b) = a$$

$$(a \vee b) \wedge (a \vee \neg b) = a$$

$$(a \wedge \neg b) \vee b = a \vee b$$

$$(a \vee \neg b) \wedge b = a \wedge b$$

## IX. De Morgan

$$\neg(a \wedge b) = \neg a \vee \neg b$$

$$\neg(a \vee b) = \neg a \wedge \neg b$$

$a$	$b$	$\neg(a \wedge b)$	$\neg a$	$\vee$	$\neg b$
0	0	1		1	
0	1	1		1	
1	0	1		1	
1	1	0		0	

# Logische Äquivalenz mittels Umformen

- Beispiel:  $A_1 = \neg(a \wedge b) \vee \neg c$  und  $A_2 = \neg(a \wedge c) \vee \neg b$

$$\neg(a \wedge b) \vee \neg c = (\neg a \vee \neg b) \vee \neg c \quad (\text{ix})$$

$$= \neg a \vee (\neg b \vee \neg c) \quad (\text{iii})$$

$$= \neg a \vee (\neg c \vee \neg b) \quad (\text{ii})$$

$$= (\neg a \vee \neg c) \vee \neg b \quad (\text{iii})$$

$$= \neg(a \wedge c) \vee \neg b \quad (\text{ix})$$

# Logische Äquivalenz mittels Umformen

- Beispiel:  $A_1 = \neg(\neg(a \wedge \neg b)) \wedge \neg c$  und  $A_2 = a \wedge \neg(b \vee c)$

$$\begin{aligned}\neg(\neg(a \wedge \neg b)) \wedge \neg c &= (a \wedge \neg b) \wedge \neg c && (\text{i}) \text{ Negation der Negation} \\ &= a \wedge (\neg b \wedge \neg c) && (\text{iii}) \text{ Assoziativgesetz} \\ &= a \wedge \neg(b \vee c) && (\text{ix}) \text{ De Morgan}\end{aligned}$$

$$A_1 = \neg(a \wedge \neg b) \vee (\neg c \vee \neg d) \vee \neg c$$

$$A_2 = \neg a \vee b \vee \neg(\neg c \wedge d) \vee \neg c$$

1. Möglichkeit: Wertetabellen

A1

a	b	c	d	$\neg(a \wedge \neg b)$	$(\neg c \vee \neg d)$	$\neg c$	A1
0	0	0	0	1	1	1	1
0	0	0	1	1	0	1	1
0	0	1	0	1	1	0	1
0	0	1	1	1	0	1	1
0	1	1	1	1	1	0	1
:							

A2

a	b	c	d	$\neg c$	b	$\neg(\neg c \wedge d)$	$\neg c$	A2
0	0	0	0	1	0	1	1	1
0	0	0	1	1	0	0	1	1
0	0	1	0	1	0	1	0	1
0	0	1	1	1	0	1	0	1
0	1	1	1	0	1	1	0	1
:								

## 2. Möglichkeit : Umformen

$$A_1 = \neg(a \wedge \neg b) \vee (c \vee \neg d) \vee \neg c$$

$$= \neg a \vee b \vee (\neg c \vee \neg d) \vee \neg c \quad (\text{ix}) \quad \text{De Morgan}$$

$$= \neg a \vee b \vee \neg(\neg c \wedge d) \vee \neg c \quad (\text{ix}) \text{ De Morgan}$$

$$= A_2$$

Die beiden logischen Ausdrücke sind also äquivalent

$$a \Rightarrow b = \neg a \vee b$$

$$\begin{aligned} a \Leftrightarrow b &= (a \Rightarrow b) \wedge (b \Rightarrow a) = (\neg a \vee b) \wedge (\neg b \vee a) \\ &= (a \wedge b) \vee (\neg a \wedge \neg b) \end{aligned}$$

$$a \oplus b = (\neg a \wedge b) \vee (a \wedge \neg b)$$

<u>—</u>	<u>—</u>	<u>—</u>	<u>—</u>	<u>—</u>
$a_6$	$\Leftrightarrow$	$\wedge$	$G$	
00	1	0	0	
01	0	0	1	
10	0	0	1	
11	1	1	0	

## 2.5 Logische Funktionen

- **Logische Funktionen** spielen in der Informatik eine wichtige Rolle z.B. beim Entwurf von Schaltungen
- Definition: Eine Funktion  $f: \{0,1\}^n \rightarrow \{0,1\}$  heißt **vollständige (n-stellige) logische Funktion**. Sie ordnet jedem binären  $n$ -Tupel  $(a_1, \dots, a_n)$  einen Wahrheitswert in  $\{0,1\}$  zu

$$\mathbb{B}^n = \{0,1\}^n = \underbrace{\{0,1\} \times \dots \times \{0,1\}}_{n-\text{mal}}$$
$$(0,1,1,0,0) \in \mathbb{B}^5$$

# Logische Funktionen

- Beispiel: An einem Haus sind drei Alarmsensoren befestigt. Ein Alarm soll ausgelöst werden, wenn **mindestens zwei** der Sensoren aktiv sind
  - $a_i$  bezeichne den *i*-ten Sensor  
( $a_i = 0$  : Sensor *i* nicht aktiv /  $a_i = 1$  : Sensor *i* aktiv)

# Logische Funktionen

- Wir geben nun die **Wertetabelle** einer 3-stelligen logischen Funktion an, die angibt, wann ein Alarm ausgelöst wird

$a_1$	$a_2$	$a_3$	$f(a_1, a_2, a_3)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

- Wie können wir eine gegebene  $n$ -stellige logische Funktion **kompakt als logischen Ausdruck** darstellen?

# Disjunktive Normalform

- Definition: Ein logischer Ausdruck der Form

$$K_1 \vee K_2 \vee K_3 \vee \dots \vee K_n$$

heißt in disjunktiver Normalform (DN), wenn alle  
**Konjunktionsterme**  $K_i$  ( $1 \leq i \leq n$ )  
**paarweise verschieden** sind

- Beispiele: Folgende Ausdrücke sind in disjunktiver Normalform

$$(a \wedge b) \vee (\neg a \wedge \neg b)$$

$$(a \wedge b \wedge c) \vee (\neg a \wedge \neg b \wedge c) \vee (a \wedge \neg b \wedge \neg c)$$

# Disjunktive Normalform

- Beispiele: Folgende Ausdrücke sind nicht in disjunktiver Normalform

$$(a \wedge b) \vee (\neg a \wedge \neg b \wedge c) \vee (a \wedge b)$$

$$(a \wedge b \wedge \neg c) \vee (a \wedge b \wedge \neg c) \vee (a \wedge b)$$

- Zu jedem logischen Ausdruck existieren in der Regel **mehrere** äquivalente logische Ausdrücke in disjunktiver Normalform

- Beispiel:

$$(a \wedge b) \vee (a \wedge \neg b) \vee (a \wedge c) \vee (\neg a \wedge c) = a \vee c$$

# Kanonische disjuktive Normalform

- Definition: Ein logischer Ausdruck

$$K_1 \vee K_2 \vee K_3 \vee \dots \vee K_n$$

heißt in **kanonischer disjunktiver Normalform (KDN)**, wenn er in DN ist und zusätzlich jeder der vorhandenen Konjunktionsterme **alle  $n$  Variablen** enthält und **keine zwei Konjunktionsterme logisch äquivalent** sind

- Es gibt **keine zwei verschiedenen** logischen Ausdrücke, die **äquivalent** sind **und** sich in **KDN** befinden

$$\begin{aligned} & (\neg a \wedge b) \vee (a \wedge \neg b) \\ & \underline{\neg(a \vee \neg b) \vee \neg(\neg a \vee b)} \\ & \dots \end{aligned}$$

↳ Menge von logischen Ausdrücken, die paarweise äquivalent zueinander sind

KDN kann als Repräsentant dieser Menge verstanden werden

Bestimmen wir zu einem beliebigen Ausdruck der Menge die KDN, so muss genau dieser Repräsentant rauskommen

# Kanonische disjunktive Normalform

- Die KDN einer logischen Funktion lässt sich wie folgt bestimmen:
  - stelle die **Wertetabelle** der logischen Funktion auf
  - **konstruiere** die Konjunktionsterme aus den Zeilen mit **Wahrheitswert 1**
    - ist  $a_i$  in der Zeile wahr (d.h. mit 1 belegt)  
wird es zur Variablen  $a_i$
    - ist  $a_i$  in der Zeile falsch (d.h. mit 0 belegt)  
wird es zur Variablen  $\neg a_i$

# Kanonische disjuktive Normalform

- Beispiel: Die KDN zu unserem Alarm-Beispiel lässt sich aus der Wertetabelle

	$a_1$	$a_2$	$a_3$	$f(a_1, a_2, a_3)$
	0	0	0	0
	0	0	1	0
	0	1	0	0
1	0	1	1	1
	1	0	0	0
2	1	0	1	1
3	1	1	0	1
4	1	1	1	1

ablesen als

$$(\neg a_1 \wedge a_2 \wedge a_3) \vee (a_1 \wedge \neg a_2 \wedge a_3) \vee (a_1 \wedge a_2 \wedge \neg a_3) \vee (a_1 \wedge a_2 \wedge a_3)$$

1                    2                    3                    4

# Kanonische disjunktive Normalform

- Die KDN der durch die Wertetabelle

	$a_1$	$a_2$	$a_3$	$f(a_1, a_2, a_3)$
1	0	0	0	1
	0	0	1	0
2	0	1	0	1
	0	1	1	0
3	1	0	0	0
	1	0	1	1
	1	1	0	0
	1	1	1	0

dargestellten logischen Funktion lautet

$$\underbrace{(\neg a_1 \wedge \neg a_2 \wedge \neg a_3)}_1 \vee \underbrace{(\neg a_1 \wedge a_2 \wedge \neg a_3)}_2 \vee \underbrace{(a_1 \wedge \neg a_2 \wedge a_3)}_3$$

# Zusammenfassung

- **Boolesche Algebra** als abstrakte algebraische Struktur
- **Äquivalenz** von logischen Ausdrücken lässt sich mittels **Wertetabellen** oder durch **Umformung** überprüfen
- **Rechengesetze** zum Umformen logischer Ausdrücke
- **Kanonische disjunktive Normalform** als eindeutige Darstellung einer logischen Funktion

# Literatur

- [1] **H.-P. Gumm und M. Sommer:** *Einführung in die Informatik*, Oldenbourg Verlag, 2012 (**Kapitel 5.2**)