Algorithmen und Datenstrukturen (für ET/IT)

Wintersemester 2012/13

Dr. Tobias Lasser

Computer Aided Medical Procedures
Technische Universität München

Programm

2. Mathematische Grundlagen
   - Mengen
   - Abbildungen
   - Zahldarstellung
   - Boolesche Logik

Definition Menge

Menge nach Georg Cantor (1883)

Eine Menge ist eine Zusammenfassung M von bestimmten, wohlunterschiedenen Objekten m unserer Anschauung oder unseres Denkens zu einem Ganzen.

- die m heißen Elemente der Menge M
- in Zeichen: $m \in M$

Mengen Beispiele

- $\varnothing = \{\}$ die leere Menge
- $\mathbb{N} = \{1, 2, 3, \ldots\}$ die natürlichen Zahlen
- $\mathbb{N}_0 = \{0, 1, 2, 3, \ldots\}$
- $\mathbb{Z} = \{0, 1, -1, 2, -2, \ldots\}$ die ganzen Zahlen
- $\mathbb{Q} = \left\{ \frac{m}{n} : \ m \in \mathbb{Z}, \ n \in \mathbb{N} \right\}$ die rationalen Zahlen
- $\mathbb{R} = \left\{ \lim_{n\to\infty} a_n : \ (a_n)_{n\geq 1} \subset \mathbb{Q} \text{ konvergent} \right\}$
  die reellen Zahlen
- $M = \{\varnothing, \mathbb{N}, \mathbb{N}_0, \mathbb{Z}, \mathbb{Q}, \mathbb{R}\}$
  Mengen können auch Elemente von Mengen sein

Russellsche Antinomie

Die Eigenschaft $E(x)$ bedeute daß $x \notin x$.

Definiere

$$M := \{x : \ E(x)\} = \{x : \ x \notin x\}.$$

Frage: Ist $M \in M$ oder $M \notin M$?

Angenommen $M \notin M$. Dann hat M Eigenschaft E. Also $M \in M$. Widerspruch!

Angenommen $M \in M$. Dann hat M nicht Eigenschaft E. Also $M \notin M$. Widerspruch!

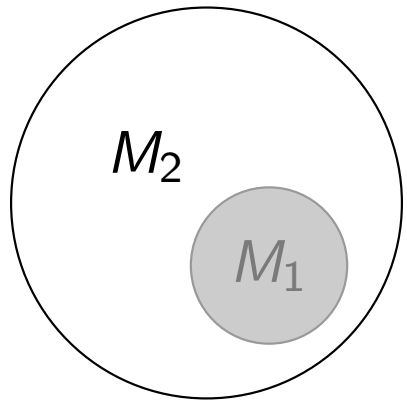
Lösung: moderne Mengenlehre nach Zermelo-Fraenkel mit Aussonderungsaxiom, M ist keine Menge.

Teilmengen

Definition Teilmenge

Seien M_1, M_2 Mengen.

- M_1 ist eine Teilmenge von M_2 (in Zeichen: $M_1 \subset M_2$)
- genau dann, wenn
- aus $m \in M_1$ folgt $m \in M_2$



M_1 und M_2 sind gleich ($M_1 = M_2$), genau dann wenn $M_1 \subset M_2$ und $M_2 \subset M_1$.

Kardinalität

Kardinalität

Die Anzahl der Elemente einer Menge M heißt Kardinalität und wird mit $|M|$ bezeichnet.

Beispiele:

- $|\{0, 8, 15\}| = 3$
- $|\varnothing| = 0$
- $|\mathbb{N}| = \infty$, d.h. $\mathbb{N}$ hat unendlich viele Elemente

Operationen auf Mengen

Seien M, N Mengen.

- Vereinigung: $M \cup N := \{x : \ x \in M \text{ oder } x \in N\}$

- Durchschnitt: $M \cap N := \{x : \ x \in M \text{ und } x \in N\}$

- Differenz: $M \setminus N := \{x : \ x \in M \text{ und } x \notin N\}$

Operationen auf Mengen: Rechenregeln

Seien A, B, C Mengen.

$$A \cup B = B \cup A \qquad \text{kommutativ}$$
$$A \cap B = B \cap A$$

$$A \cup (B \cup C) = (A \cup B) \cup C \qquad \text{assoziativ}$$
$$A \cap (B \cap C) = (A \cap B) \cap C$$

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C) \qquad \text{distributiv}$$
$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

$$(A \setminus B) \cup (A \setminus C) = A \setminus (B \cap C) \qquad \text{de Morgan Gesetz}$$
$$(A \setminus B) \cap (A \setminus C) = A \setminus (B \cup C)$$

$$(A \setminus B) \cup (B \setminus A) = (A \cup B) \setminus (A \cap B)$$

Potenzmenge

Potenzmenge

Sei M Menge. Die Menge

$$\mathcal{P}(M) := \{N : \ N \subset M\}$$

heißt Potenzmenge von M.

Eigenschaft: Ist $|M| = n$, so ist $|\mathcal{P}(M)| = 2^n$.

Beispiel:

- $\mathcal{P}(\varnothing) = \{\varnothing\}$
- $\mathcal{P}(\{1, 2, 3\}) = ?$

Kartesisches Produkt von Mengen

Statt Mengen $\{x, y\}$ kann man auch geordnete Paare (x, y) betrachten.

Kartesisches Produkt

Seien X, Y Mengen. Die Menge

$$X \times Y := \{(x, y) : \ x \in X, y \in Y\}$$

heißt kartesisches Produkt von X und Y.

Eigenschaft: $|X \times Y| = |X| \ |Y|$.

Beispiele:

- $\{a, b, c\} \times \{1, 2\} = \{(a, 1), (a, 2), (b, 1), (b, 2), (c, 1), (c, 2)\}$
- $\mathbb{R} \times \mathbb{R} = \mathbb{R}^2 = \{(x, y) : \ x, y \in \mathbb{R}\}$ zwei-dimensionaler Raum
- x-Achse $= \{(x, 0) : \ x \in \mathbb{R}\}$, y-Achse $= \{(0, y) : \ y \in \mathbb{R}\}$

Programm

2. Mathematische Grundlagen
   - Mengen
   - Abbildungen
   - Zahldarstellung
   - Boolesche Logik

Definition Abbildung

Abbildung / Funktion

Seien X, Y Mengen. Eine Vorschrift, die jedem $x \in X$ genau ein $y \in Y$ zuordnet, heißt Abbildung oder Funktion.

Das dem $x \in X$ zugeordnete $y \in Y$ wird bezeichnet mit

$$y = f(x)$$

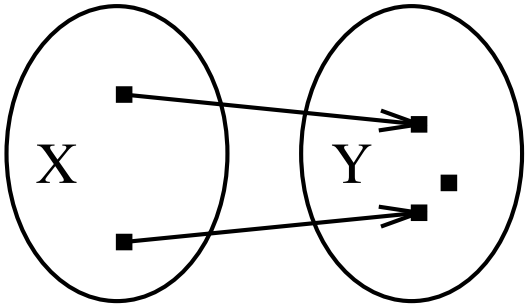
und heißt *das* Bild von x unter f. x heißt *ein* Urbild von y unter f.

X heißt Definitionsbereich von f, Y heißt Bildbereich von f.

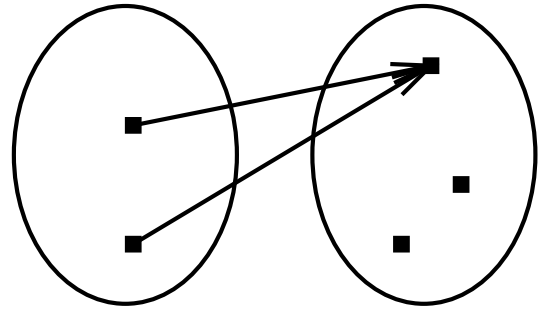
Kurzschreibweise: $f : X \to Y$ oder $f : x \mapsto y$.

Achtung: Es muss nicht jedes $y \in Y$ als Bild eines $x \in X$ auftreten.

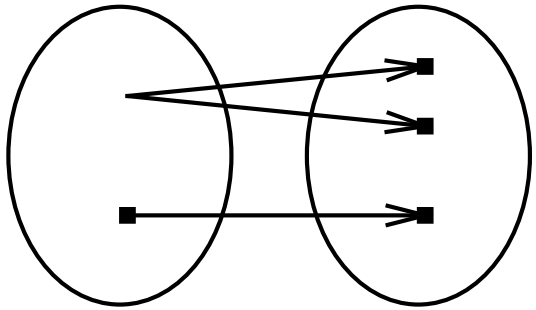
Abbildung Beispiele



Abbildung

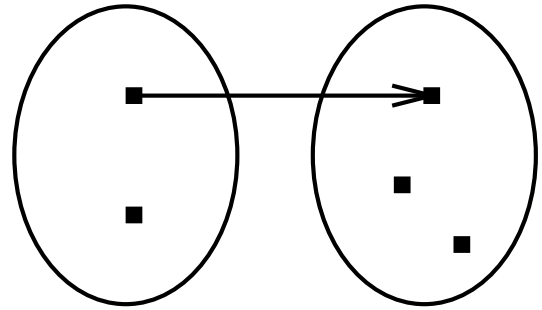


Abbildung



keine Abbildung

(Zwei y haben dasselbe Urbild)



keine Abbildung

(Ein x hat kein Bild)

Abbildung Beispiele

Beispiel-Problem Navigationssystem Auto:

Berlin $\longrightarrow$  $\longrightarrow$ München

$X =$ Menge der Verbindungen zwischen den Städten
(z.B. München-Frankfurt)

$Y = \mathbb{R}$ reelle Zahlen

$f : X \to Y$ weist jeder Verbindung die Distanz in Kilometer zu

z.B. $f(\text{München-Frankfurt}) = 392$
$f(\text{Frankfurt-Berlin}) = 549$

Abbildungseigenschaften

Injektiv, surjektiv, bijektiv

Sei $f : X \to Y$ Abbildung.

- f heißt injektiv genau dann, wenn zu jedem $y \in Y$ höchstens ein $x \in X$ existiert mit $f(x) = y$.
- f heißt surjektiv genau dann, wenn zu jedem $y \in Y$ mindestens ein $x \in X$ existiert mit $f(x) = y$.
- f heißt bijektiv genau dann, wenn zu jedem $y \in Y$ genau ein $x \in X$ existiert mit $f(x) = y$.

Eigenschaft: f ist bijektiv genau dann, wenn es injektiv und surjektiv ist.

Beispiel: $f : \mathbb{N} \to \mathbb{N}$, $f(n) = n + 1$ ist injektiv, aber nicht surjektiv (da $1 \notin f(\mathbb{N})$) und damit nicht bijektiv.

Folgen

Folge

Sei M Menge. Eine Folge ist eine Abbildung $a : \mathbb{N} \to M$ mit

$$i \mapsto a_i \quad \text{bzw.} \quad a(i) = a_i.$$

Eine endliche Folge ist eine Abbildung $a : \{1, \ldots, n\} \to M$ für $n \in \mathbb{N}$. n ist die Länge der Folge.

Kurzschreibweise: $(a_i)_{i \in \mathbb{N}}$ bzw. $(a_i)_{i=1,\ldots,n}$

Beispiele:

- Folge der Primzahlen: $(2, 3, 5, 7, \ldots)$
- Folge der ersten acht Zweierpotenzen: $(2, 4, 8, 16, 32, 64, 128, 256)$

Polynome

Polynome

Sei $n \in \mathbb{N}$. Eine Funktion $p : \mathbb{R} \to \mathbb{R}$ der Form

$$p(x) = c_n x^n + c_{n-1} x^{n-1} + \ldots + c_1 x^1 + c_0$$

heißt Polynom in x vom Grad n. Hierbei heißen die $c_i \in \mathbb{R}$, $i = 1, \ldots, n$, Koeffizienten von p, $c_n \neq 0$.

Beispiel: Dezimaldarstellung von Zahlen

$$123 = 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0$$

Programm

2. Mathematische Grundlagen
   - Mengen
   - Abbildungen
   - Zahldarstellung
   - Boolesche Logik

Zahldarstellung

- Dezimalsystem:
  - Basis $x = 10$
  - Koeffizienten $c_n \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
  - Beispiel: $123_{10} = 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0$
- Binärsystem:
  - Basis $x = 2$
  - Koeffizienten $c_n \in \{0, 1\}$
  - Beispiel: $1101_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 13_{10}$

Zahldarstellung

- Oktalsystem:
  - Basis $x = 8\ (= 2^3)$
  - Koeffizienten $c_n \in \{0, 1, 2, 3, 4, 5, 6, 7\}$
  - Beispiel: $173_8 = 1 \cdot 8^2 + 7 \cdot 8^1 + 3 \cdot 8^0 = 123_{10}$
- Hexadezimalsystem:
  - Basis $x = 16\ (= 2^4)$
  - Koeffizienten $c_n \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$
  - Beispiel: $7B_{16} = 7 \cdot 16^1 + B \cdot 16^0 = 123_{10}$

Wie viele Ziffern pro Zahl?

Problem

Gegeben Zahl $z \in \mathbb{N}$, wie viele Ziffern m werden bezüglich Basis x benötigt?

Lösung

$$m = \lfloor \log_x(z) \rfloor + 1$$

Erläuterung: $(a \in \mathbb{R})$

- $\lfloor a \rfloor = \text{floor}(a) =$ größte ganze Zahl kleiner gleich a
- $\lceil a \rceil = \text{ceil}(a) =$ kleinste ganze Zahl größer gleich a

$$a - 1 < \lfloor a \rfloor \leq a \leq \lceil a \rceil < a + 1$$

- $\log_x(z) = \frac{\ln(z)}{\ln(x)}$, wobei „ln“ der natürliche Logarithmus ist

Wie viele Ziffern pro Zahl?

Lösung

$$m = \lfloor \log_x(z) \rfloor + 1$$

Beispiele: $z = 123$

- Basis $x = 10$:

$$m = \lfloor \log_{10}(123) \rfloor + 1 = \lfloor 2.0899\ldots \rfloor + 1 = 3$$

- Basis $x = 2$:

$$m = \lfloor \log_2(123) \rfloor + 1 = \lfloor 6.9425\ldots \rfloor + 1 = 7$$

- Basis $x = 8$:

$$m = \lfloor \log_8(123) \rfloor + 1 = \lfloor 2.3141\ldots \rfloor + 1 = 3$$

- Basis $x = 16$:

$$m = \lfloor \log_{16}(123) \rfloor + 1 = \lfloor 1.7356\ldots \rfloor + 1 = 2$$

Größte Zahl pro Anzahl Ziffern?

Problem

Gegeben Basis x und m Ziffern, was ist die größte darstellbare Zahl?

Lösung

$$z_{max} = x^m - 1$$

Beispiele:

- $x = 2,\ m = 4$:

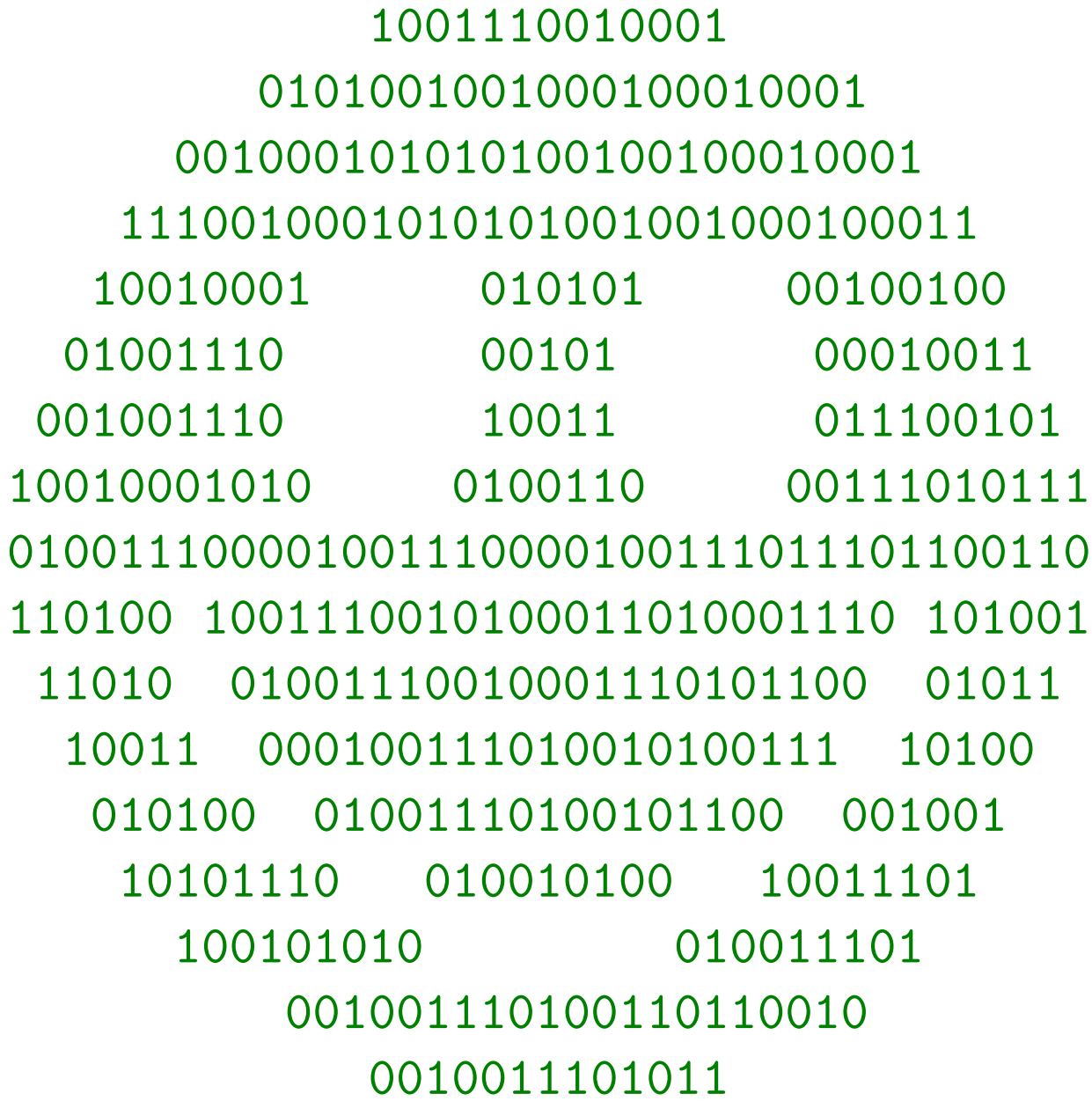
$$z_{max} = 2^4 - 1 = 15 = 1111_2$$

- $x = 2,\ m = 8$:

$$z_{max} = 2^8 - 1 = 255 = 11111111_2$$

- $x = 16,\ m = 2$:

$$z_{max} = 16^2 - 1 = 255 = FF_{16}$$


1001110010001
01010010010001000100001
00100010101010010010001000 1
11100100010101010010010001000 11
10010001 010101 00100100
01001110 00101 00010011
001001110 10011 011100101
10010001010 0100110 00111010111
0100111000010011100001001110111011001 10
110100 1001110010100011010001110 101001
11010 0100111001000111010110 0 01011
10011 0001001110100101001 11 10100
010100 0100111010010110 0 001001
10101110 010010100 10011101
100101010 010011101
0010011101001101100 10
0010011101011

Primitive Datentypen

Primitive Datentypen

In Computern verwendet man Binärdarstellung mit einer fixen Anzahl Ziffern (genannt Bits).

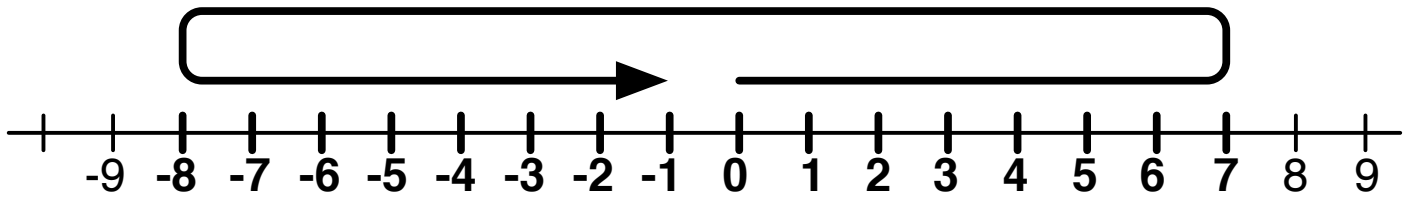
Die primitiven Datentypen sind

- 8 Bits (auch genannt Byte), darstellbare Zahlen: $\{0, \ldots, 255\}$
  in C, C++: `unsigned char`
- 16 Bits, darstellbare Zahlen: $\{0, \ldots, 65535\}$
  in C, C++: `unsigned short`
- 32 Bits, darstellbare Zahlen: $\{0, \ldots, 4294967295\}$
  in C, C++: `unsigned long`
- 64 Bits, darstellbare Zahlen: $\{0, \ldots, 2^{64} - 1\}$
  in C, C++: `unsigned long long`

Negative Zahlen

Darstellung durch 2-Komplement

Beispiel für 4 Bits (darstellbare Zahlen: $2^4 = 16$):



Damit erhält man:

0000 = +0	0100 = +4	1000 = -8	1100 = -4
0001 = +1	0101 = +5	1001 = -7	1101 = -3
0010 = +2	0110 = +6	1010 = -6	1110 = -2
0011 = +3	0111 = +7	1011 = -5	1111 = -1

Das erste Bit ist also das Vorzeichen!

2-Komplement Darstellung I

2-Komplement Darstellung

Sei $x \in \mathbb{N}$, $x > 0$. Die 2-Komplement Darstellung $-x_z$ von $-x$ mittels n Bits ist gegeben durch

$$-x_z = 2^n - x.$$

Vorheriges Beispiel war: $-5 = 1011$, also $x = 5$ und $n = 4$.

Nun:

$$-5_z = 2^4 - 5 = 16 - 5 = 11 = 1011_2$$

2-Komplement Darstellung II

Sei $b_n b_{n-1} \dots b_1$ eine Bitfolge.

- $(b_n b_{n-1} \dots b_1)_z$ sei der Zahlwert in 2-Komplement Darstellung
- für positive Zahlen von 0 bis $2^{n-1} - 1$ entspricht $(b_n b_{n-1} \dots b_1)_z$ der Binärdarstellung:

$$(0 b_{n-1} \dots b_1)_z = (0 b_{n-1} \dots b_1)_2$$

- für negative Zahlen von -2^{n-1} bis -1 gilt

$$(1 b_{n-1} \dots b_1)_z = -2^{n-1} + (0 b_{n-1} \dots b_1)_2$$

- allgemein:

$$(b_n b_{n-1} \dots b_1)_z = b_n \cdot (-2^{n-1}) + (b_{n-1} \dots b_1)_2$$

Eigenschaften 2-Komplement

- Für $n \in \mathbb{N}$ gilt

$$\begin{aligned}(111\ldots11)_z &= (-2^{n-1}) + 2^{n-2} + \ldots + 2^1 + 2^0 \\ &= -2^{n-1} + (2^{n-1} - 1) \\ &= -1\end{aligned}$$

- Um $-x$ aus x in 2-Komplement Darstellung zu erhalten:
  Bilde bitweises Komplement und addiere 1.
  - Beispiel: Negatives von $6 = (0110)_2$ mit $n = 4$

$$-6 = (\bar{0}\bar{1}\bar{1}\bar{0})_z + 1 = (1001)_z + 1 = (1010)_z$$

  - und zurück:

$$6 = (\bar{1}\bar{0}\bar{1}\bar{0})_z + 1 = (0101)_z + 1 = (0110)_z$$

Ganze Zahlen in C, C++

Ganze Zahlen in C, C++

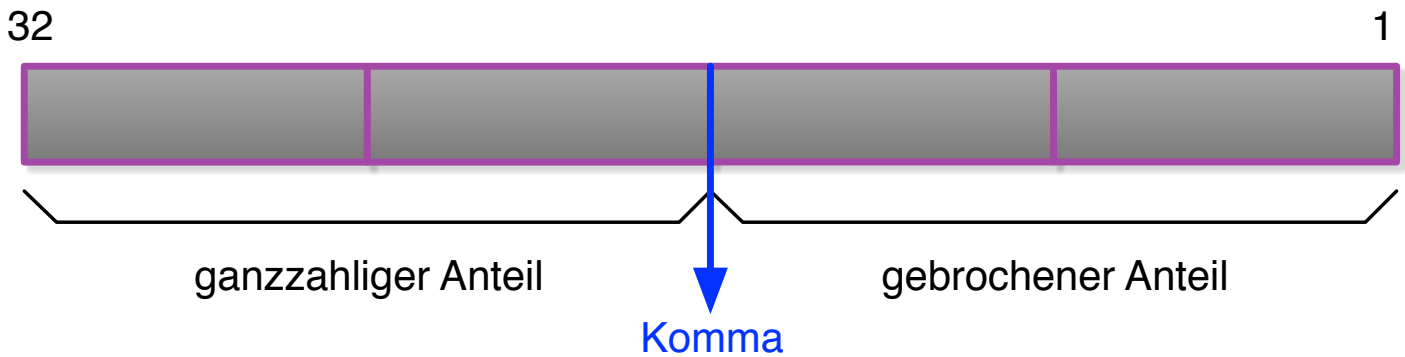
- 8 Bits: `unsigned char` $\{0, \ldots, 255\}$
  `signed char` $\{-128, \ldots, 127\}$
- 16 Bits: `unsigned short` $\{0, \ldots, 65535\}$
  `signed short` $\{-32768, \ldots, 32767\}$
- 32 Bits: `unsigned long` $\{0, \ldots, 2^{32} - 1\}$
  `signed long` $\{-2^{31}, \ldots, 2^{31} - 1\}$
- 64 Bits: `unsigned long long` $\{0, \ldots, 2^{64} - 1\}$
  `signed long long` $\{-2^{63}, \ldots, 2^{63} - 1\}$

- `signed` kann weggelassen werden (ausser bei `char`!)
- `unsigned int` und `signed int` sind je nach System 16, 32 oder 64 Bit

Rationale Zahlen I

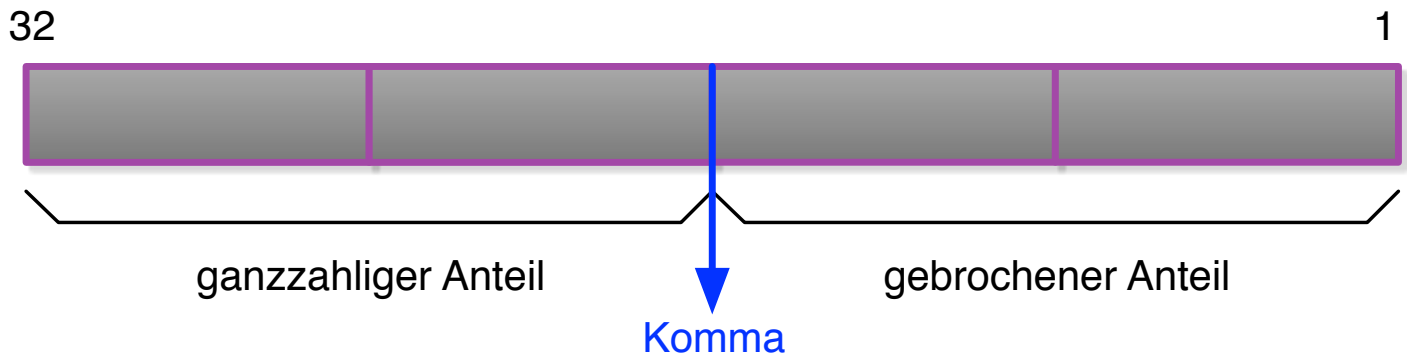
Festkomma Darstellung:

- Komma an fester Stelle in Zahl
- Beispiel mit $n = 32$:



- Nachteile:
  - weniger große Zahlen darstellbar
  - feste Genauigkeit der Nachkommastellen

Rationale Zahlen II



- Interpretation für $r \in \mathbb{Q}$:

$$r = c_n \cdot 2^n + \ldots + c_0 \cdot 2^0 + c_{-1}2^{-1} + \ldots + c_{-m} \cdot 2^{-m}$$

mit n Vorkomma- und m Nachkomma-Ziffern

- Beispiel:

$$\begin{aligned} 11.01_2 &= 1 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} \\ &= 2 + 1 + 0 + \tfrac{1}{4} = 3.25_{10} \end{aligned}$$

Floating Point Zahlen I

Wissenschaftliche Notation:

- $x = a \cdot 10^b$ für $x \in \mathbb{R}$, wobei:
  - $a \in \mathbb{R}$ mit $1 \leq |a| < 10$
  - $b \in \mathbb{Z}$

- Beispiele:
  - $-2.7315 \cdot 10^2$ °C absoluter Nullpunkt
  - $1.0 \cdot 10^9$ Hz Taktfrequenz A5X Prozessor

- Drei Bestandteile:
  - Vorzeichen
  - Mantisse $|a|$
  - Exponent b
- Problem: bei fester Länge der Mantisse (z.B. 3 Ziffern)
  - zwischen $1.23 \cdot 10^4 = 12300$ und $1.24 \cdot 10^4 = 12400$ keine Zahl darstellbar!

Floating Point Zahlen II

- wissenschaftliche Darstellung mit Basis 2

$$f = (-1)^V \cdot (1 + M) \cdot 2^{E - bias}$$

- Vorzeichen Bit V
- Mantisse M hat immer die Form $1.abc$, also wird erste Stelle weggelassen („hidden bit“)
- Exponent E wird vorzeichenlos abgespeichert, verschoben um $bias$
  - bei 32 bit float: $bias = 127$, bei 64 bit double: $bias = 1023$

Floating Point Zahlen III

Übliche Floating Point Formate:

Bit	Vorz.	Exponent	Mantisse	gültige Dezimalst.	darstellbarer Bereich
32	1 Bit	8 Bit	23 Bit	~ 7	$\pm 2 \cdot 10^{-38}$ bis $\pm 2 \cdot 10^{38}$
64	1 Bit	11 Bit	52 Bit	~ 15	$\pm 2 \cdot 10^{-308}$ bis $\pm 2 \cdot 10^{308}$
80	1 Bit	15 Bit	64 Bit	~ 19	$\pm 1 \cdot 10^{-4932}$ bis $\pm 1 \cdot 10^{4932}$

In C, C++:

`float` (32 Bit), `double` (64 Bit), `long double` (80 Bit)

Vorsicht mit Floating Point!

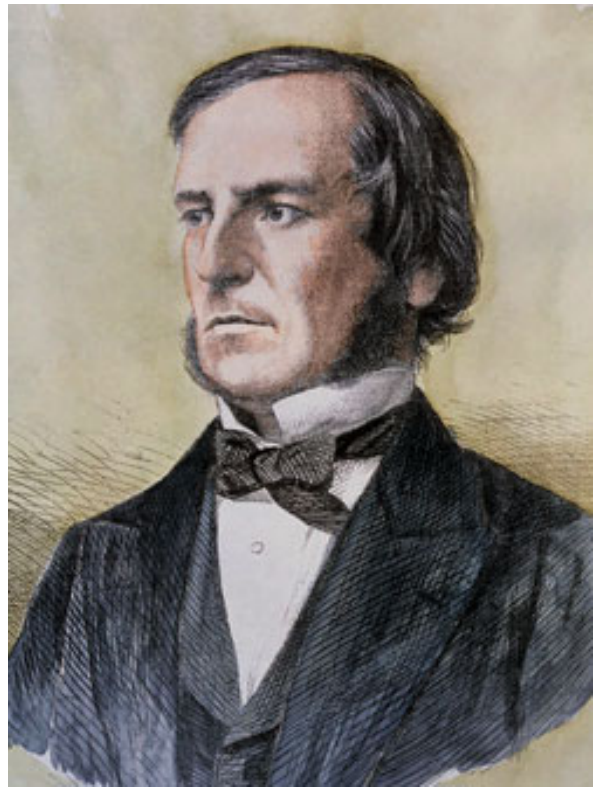
Floating Point Zahlen sind bequem, aber Vorsicht!

- Viele Dezimalzahlen haben keine Floating Point Darstellung
  - Beispiel: $0.1_{10} = 0.0001100110011\ldots_2$ (periodisch)
- Kritisch sind Vergleiche von Floating Point Zahlen
  - Beispiel: `(0.1 + 0.2 == 0.3)` ist meist FALSE!
- Zins-Berechnungen und dergleichen NIE mit Floating Point Zahlen!
  - Stattdessen: spezielle Bibliotheken wie GMP

Programm

2. Mathematische Grundlagen
   - Mengen
   - Abbildungen
   - Zahldarstellung
   - Boolesche Logik

George Boole



Englischer Mathematiker (1815-1864)

Logische Werte

Boolesche Logik: Logik mit zwei Werten

- Repräsentationen:
  - 1 und 0
  - W und F (in Englisch: T und F)
  - L und O
- Mengensymbol $\mathbb{B}$
  - $\mathbb{B} = \{0, 1\} = \{F, W\} = \{O, L\}$

Logische Werte und Verknüpfungen

„Grundrechenarten“ mit logischen Werten:

- Konjunktion: $\wedge : \ \mathbb{B} \times \mathbb{B} \to \mathbb{B}$
  - ähnlich zu Multiplikation bei Zahlen
  - auch bezeichnet als UND bzw. AND
- Disjunktion: $\vee : \ \mathbb{B} \times \mathbb{B} \to \mathbb{B}$
  - ähnlich zu Addition bei Zahlen
  - auch bezeichnet als ODER bzw. OR
- Negation: $\neg : \ \mathbb{B} \to \mathbb{B}$
  - auch bezeichnet als NICHT bzw. NOT

Wahrheitstabelle:

a	b	$a \wedge b$
0	0	0
0	1	0
1	0	0
1	1	1

a	b	$a \vee b$
0	0	0
0	1	1
1	0	1
1	1	1

a	$\neg a$
0	1
1	0

Weitere Verknüpfungen I

- NAND: $\uparrow$: $\mathbb{B} \times \mathbb{B} \to \mathbb{B}$
  - $a \uparrow b = \neg(a \wedge b)$
  - mit NAND lassen sich NOT, OR, AND erzeugen
- NOR: $\downarrow$: $\mathbb{B} \times \mathbb{B} \to \mathbb{B}$
  - $a \downarrow b = \neg(a \vee b)$
  - mit NOR lassen sich ebenso NOT, OR, AND erzeugen
- XOR: $\oplus$: $\mathbb{B} \times \mathbb{B} \to \mathbb{B}$
  - auch exklusiv oder genannt
  - erzeugbar aus $\neg(a \wedge b) \wedge (a \vee b)$ (siehe Übung)

Wahrheitstabelle:

a	b	$a \uparrow b$
0	0	1
0	1	1
1	0	1
1	1	0

a	b	$a \downarrow b$
0	0	1
0	1	0
1	0	0
1	1	0

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

Weitere Verknüpfungen II

- Implikation: $\Rightarrow$: $\mathbb{B} \times \mathbb{B} \to \mathbb{B}$
  - oft verwendet für mathematische Sätze: „a impliziert b“, „aus a folgt b“
  - Beispiel: „aus $n < 3$ folgt $n < 5$“
  - erzeugbar aus $\neg a \vee b$

Wahrheitstabelle:

a	b	$a \Rightarrow b$
0	0	1
0	1	1
1	0	0
1	1	1

- Äquivalenz: $\Leftrightarrow$: $\mathbb{B} \times \mathbb{B} \to \mathbb{B}$
  - oft verwendet für mathematische Sätze: „a gilt genau dann, wenn b gilt“, „a und b sind äquivalent“
  - Beispiel: „f ist bijektiv genau dann, wenn f injektiv und surjektiv ist“
  - erzeugbar aus $(a \wedge b) \vee (\neg a \wedge \neg b)$

a	b	$a \Leftrightarrow b$
0	0	1
0	1	0
1	0	0
1	1	1

Rechenbeispiel

Zeige:

$$a \Leftrightarrow b \text{ entspricht } (a \wedge b) \vee (\neg a \wedge \neg b)$$

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

Rangfolge und Rechenregeln

Rangfolge:

- NICHT vor UND
- UND vor ODER

Beispiel:

$$\neg 0 \vee 1 \wedge 0 = (\neg 0) \vee (1 \wedge 0) = 1 \vee 0 = 1$$

De Morgan-Gesetze:

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

Ausblick

(Ausblick nicht klausur-relevant)

- Umformung von Booleschen Ausdrücken in Normalformen
  - Disjunktive Normalform (DNF)
  - Konjunktive Normalform (KNF)
- K-V Diagramme
- Entwurf von Schaltungen

$\longrightarrow$ Schaltungstechnik

Logische Ausdrücke in C, C++

- logische Variablen: `bool a,b;`
- logische Werte: `true` und `false`
- NOT Operator: `!a`
- AND Operator: `a && b`
- OR Operator: `a || b`

Beispiele:

- `( (2 == 2) && (3 < 1) )`
  ergibt `(true && false)`, also `false`
- `( !(2 == 2) || (3 > 1) )`
  ergibt `(false || true)`, also `true`
- Kurzform für `!(2 == 2)` ist `(2 != 2)`

Zusammenfassung

2. Mathematische Grundlagen
   - Mengen
   - Abbildungen
   - Zahldarstellung
   - Boolesche Logik