Grundbegriffe der Informatik

Einheit 8: Übersetzungen und Codierungen

Mattias Ulbrich
(basierend auf Folien von Thomas Worsch)

KIT · Institut für Theoretische Informatik

Wintersemester 2023/2024

Themen dieses Kapitels

Als erstes geht es um die
Interpretation von Wörtern als Zahlen
und die
Darstellung von Zahlen als Wörter auf unterschiedliche Weisen.

Nach einem kurzen Einschub zu
Notationen von Funktionen beschäftigen wir uns mit
Übersetzungen: Abbildungen, die die Bedeutung erhalten.

Huffman-Codierung ist eine Übersetzung, die umgekehrt werden kann
und auch benutzt wird, um (manche) Wörter
ohne Informationsverlust zu komprimieren.

Zahldarstellung auch für negative Zahlen betrachten wir im letzten Abschnitt

Wo sind wir?

Von Wörtern zu Zahlen und zurück

Notationen für Funktionen

Von einem Alphabet zum anderen

Huffman-Codierung

Zahldarstellung auch für negative Zahlen

Wo sind wir?

Von Wörtern zu Zahlen und zurück

- Dezimaldarstellung von Zahlen
- Andere Zahldarstellungen
- Von Zahlen zu ihren Darstellungen

Notationen für Funktionen

Von einem Alphabet zum anderen

Huffman-Codierung

Zahldarstellung auch für negative Zahlen

Dezimaldarstellung von Zahlen

- Ziffern des Alphabetes $Z_{10} = \{\texttt{0}, \ldots, \texttt{9}\}$.
- Bedeutung einzelner Ziffer x als Zahl $\mathrm{num}_{10}(x)$:

x	$\texttt{0}$	$\texttt{1}$	$\texttt{2}$	$\cdots$	$\texttt{8}$	$\texttt{9}$
$\mathrm{num}_{10}(x)$	*null*	*eins*	*zwei*	$\cdots$	*acht*	*neun*

Dezimaldarstellung von Zahlen

- Ziffern des Alphabetes $Z_{10} = \{\texttt{0}, \ldots, \texttt{9}\}$.
- Bedeutung einzelner Ziffer x als Zahl $\mathrm{num}_{10}(x)$:

x	0	1	2	3	4	5	6	7	8	9
$\mathrm{num}_{10}(x)$	0	1	2	3	4	5	6	7	8	9

Dezimaldarstellung von Zahlen

- Ziffern des Alphabetes $Z_{10} = \{\mathtt{0}, \ldots, \mathtt{9}\}$.
- Bedeutung einzelner Ziffer x als Zahl $\mathrm{num}_{10}(x)$:

Syntax

Semantik

x	$\mathtt{0}$	$\mathtt{1}$	$\mathtt{2}$	$\mathtt{3}$	$\mathtt{4}$	$\mathtt{5}$	$\mathtt{6}$	$\mathtt{7}$	$\mathtt{8}$	$\mathtt{9}$
$\mathrm{num}_{10}(x)$	0	1	2	3	4	5	6	7	8	9

Dezimaldarstellung von Zahlen

- Ziffern des Alphabetes $Z_{10} = \{\mathtt{0}, \dots, \mathtt{9}\}$.
- Bedeutung einzelner Ziffer x als Zahl $\mathrm{num}_{10}(x)$:

Syntax

Semantik

x	$\mathtt{0}$	$\mathtt{1}$	$\mathtt{2}$	$\mathtt{3}$	$\mathtt{4}$	$\mathtt{5}$	$\mathtt{6}$	$\mathtt{7}$	$\mathtt{8}$	$\mathtt{9}$
$\mathrm{num}_{10}(x)$	0	1	2	3	4	5	6	7	8	9

- Bedeutung einer Ziffernfolge $x_{k-1} \cdots x_0 \in Z_{10}^*$

$$\mathrm{Num}_{10} : Z_{10}^* \longrightarrow \mathbb{N}_0$$

Dezimaldarstellung von Zahlen

- Ziffern des Alphabetes $Z_{10} = \{0, \dots, 9\}$.
- Bedeutung einzelner Ziffer x als Zahl $\mathrm{num}_{10}(x)$:

Syntax

Semantik

x	0	1	2	3	4	5	6	7	8	9
$\mathrm{num}_{10}(x)$	0	1	2	3	4	5	6	7	8	9

- Bedeutung einer Ziffernfolge $x_{k-1} \cdots x_0 \in Z_{10}^*$

$$\mathrm{Num}_{10} : Z_{10}^* \longrightarrow \mathbb{N}_0$$

- $\mathrm{Num}_{10}(x_{k-1} \cdots x_1 x_0)$

Dezimaldarstellung von Zahlen

- Ziffern des Alphabetes $Z_{10} = \{0, \ldots, 9\}$.
- Bedeutung einzelner Ziffer x als Zahl $\mathrm{num}_{10}(x)$:

Syntax

Semantik

x	0	1	2	3	4	5	6	7	8	9
$\mathrm{num}_{10}(x)$	0	1	2	3	4	5	6	7	8	9

- Bedeutung einer Ziffernfolge $x_{k-1} \cdots x_0 \in Z_{10}^*$

$$\mathrm{Num}_{10} : Z_{10}^* \longrightarrow \mathbb{N}_0$$

- $$\begin{aligned} &\mathrm{Num}_{10}(x_{k-1} \cdots x_1 x_0) \\ &= 10^{k-1} \cdot \mathrm{num}_{10}(x_{k-1}) + \cdots + 10^1 \cdot \mathrm{num}_{10}(x_1) + 10^0 \cdot \mathrm{num}_{10}(x_0) \end{aligned}$$

Dezimaldarstellung von Zahlen

Karlsruher Institut für Technologie

- Ziffern des Alphabetes $Z_{10} = \{\mathtt{0}, \dots, \mathtt{9}\}$.
- Bedeutung einzelner Ziffer x als Zahl $\mathrm{num}_{10}(x)$:

Syntax

Semantik

x	$\mathtt{0}$	$\mathtt{1}$	$\mathtt{2}$	$\mathtt{3}$	$\mathtt{4}$	$\mathtt{5}$	$\mathtt{6}$	$\mathtt{7}$	$\mathtt{8}$	$\mathtt{9}$
$\mathrm{num}_{10}(x)$	0	1	2	3	4	5	6	7	8	9

- Bedeutung einer Ziffernfolge $x_{k-1} \cdots x_0 \in Z_{10}^*$

$$\mathrm{Num}_{10} : Z_{10}^* \longrightarrow \mathbb{N}_0$$

- $$\begin{aligned} &\mathrm{Num}_{10}(x_{k-1} \cdots x_1 x_0) \\ &= 10^{k-1} \cdot \mathrm{num}_{10}(x_{k-1}) + \cdots + 10^1 \cdot \mathrm{num}_{10}(x_1) + 10^0 \cdot \mathrm{num}_{10}(x_0) \\ &= 10 \left(10^{k-2} \cdot \mathrm{num}_{10}(x_{k-1}) + \cdots + 10^0 \cdot \mathrm{num}_{10}(x_1)\right) + \mathrm{num}_{10}(x_0) \end{aligned}$$

Dezimaldarstellung von Zahlen

- Ziffern des Alphabetes $Z_{10} = \{\mathtt{0}, \dots, \mathtt{9}\}$.
- Bedeutung einzelner Ziffer x als Zahl $\mathrm{num}_{10}(x)$:

Syntax

Semantik

x	$\mathtt{0}$	$\mathtt{1}$	$\mathtt{2}$	$\mathtt{3}$	$\mathtt{4}$	$\mathtt{5}$	$\mathtt{6}$	$\mathtt{7}$	$\mathtt{8}$	$\mathtt{9}$
$\mathrm{num}_{10}(x)$	0	1	2	3	4	5	6	7	8	9

- Bedeutung einer Ziffernfolge $x_{k-1} \cdots x_0 \in Z_{10}^*$

$$\mathrm{Num}_{10} : Z_{10}^* \longrightarrow \mathbb{N}_0$$

- $$\begin{aligned} &\mathrm{Num}_{10}(x_{k-1} \cdots x_1 x_0) \\ &= 10^{k-1} \cdot \mathrm{num}_{10}(x_{k-1}) + \cdots + 10^1 \cdot \mathrm{num}_{10}(x_1) + 10^0 \cdot \mathrm{num}_{10}(x_0) \\ &= 10\left(10^{k-2} \cdot \mathrm{num}_{10}(x_{k-1}) + \cdots + 10^0 \cdot \mathrm{num}_{10}(x_1)\right) + \mathrm{num}_{10}(x_0) \\ &= 10 \cdot \mathrm{Num}_{10}(x_{k-1} \cdots x_1) + \mathrm{num}_{10}(x_0) \end{aligned}$$

$10 \cdot 251 + 3 = 2513$

Dezimaldarstellung von Zahlen

Karlsruher Institut für Technologie

- Ziffern des Alphabetes $Z_{10} = \{0, \dots, 9\}$.
- Bedeutung einzelner Ziffer x als Zahl $\mathrm{num}_{10}(x)$:

Syntax

Semantik

x	0	1	2	3	4	5	6	7	8	9
$\mathrm{num}_{10}(x)$	0	1	2	3	4	5	6	7	8	9

- Bedeutung einer Ziffernfolge $x_{k-1} \cdots x_0 \in Z_{10}^*$

$$\mathrm{Num}_{10} : Z_{10}^* \longrightarrow \mathbb{N}_0$$

- $$\mathrm{Num}_{10}(\varepsilon) = 0$$

$10 \cdot 251 + 3 = 2513$

$$\forall w \in Z_{10}^* \; \forall x \in Z_{10} : \mathrm{Num}_{10}(wx) = 10 \cdot \mathrm{Num}_{10}(w) + \mathrm{num}_{10}(x)$$

Induktive Definitionen — „über die Wortlänge“

- Abbildungen z. B. der Form $f\colon A^* \to M$
- definiere induktiv für jede Wortlänge $n \in \mathbb{N}_0$
  - $f(v)$ für jedes $v \in A^n$
- „Definitionsanfang“: Wortlänge $n = 0$
  - definiere $f(\varepsilon)$
- „Definitionsschritt“: für jede Wortlänge n von n nach $n+1$
  - für jedes $v \in A^{n+1}$ definiere $f(v)$
  - und benutze dabei schon $f(w)$ für $w \in A^n$
  - jedes $v \in A^{n+1}$ ist von der Form wx (bzw. xw) für $w \in A^n$ und $x \in A$

Wo sind wir?

Von Wörtern zu Zahlen und zurück

- Dezimaldarstellung von Zahlen
- Andere Zahldarstellungen
- Von Zahlen zu ihren Darstellungen

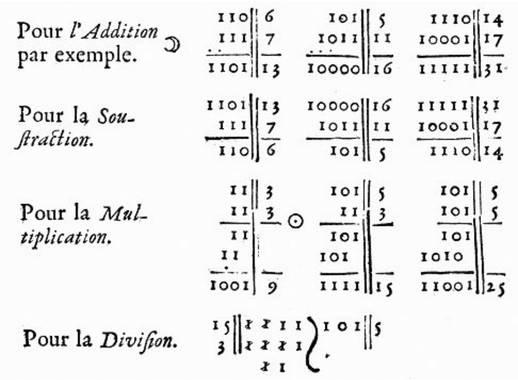
Notationen für Funktionen

Von einem Alphabet zum anderen

Huffman-Codierung

Zahldarstellung auch für negative Zahlen

Gottfried Wilhelm Leibniz: zwei Ziffern reichen aus!



http://commons.wikimedia.org/wiki/Image:Leibniz_binary_system_1703.png

- Gottfried Wilhelm Leibniz
  - geboren 1. Juli 1646 in Leipzig gestorben am 14. November 1716 in Hannover
  - baute erste Maschine, die zwei Zahlen multiplizieren konnte
- Beobachtung
  - mit nur zwei Ziffern 0 und 1 kann man
  - alle nichtnegativen ganzen Zahlen darstellen und vernünftig rechnen

Binärdarstellung von Zahlen — Stellenwertsystem zur Basis 2

- Ziffernmenge $Z_2 = \{\mathtt{0}, \mathtt{1}\}$
- definiere: $\mathrm{num}_2(\mathtt{0}) = 0$ und $\mathrm{num}_2(\mathtt{1}) = 1$ und
$$\mathrm{Num}_2(\varepsilon) = 0$$
$$\forall w \in Z_2^* \ \forall x \in Z_2 : \mathrm{Num}_2(wx) = 2 \cdot \mathrm{Num}_2(w) + \mathrm{num}_2(x)$$
- $$\begin{aligned} \mathrm{Num}_2(\mathtt{1101}) &= 2 \cdot \mathrm{Num}_2(\mathtt{110}) + 1 \\ &= 2 \cdot (2 \cdot \mathrm{Num}_2(\mathtt{11}) + 0) + 1 \\ &= 2 \cdot (2 \cdot (2 \cdot \mathrm{Num}_2(\mathtt{1}) + 1) + 0) + 1 \\ &= 2 \cdot (2 \cdot (2 \cdot (2 \cdot \mathrm{Num}_2(\varepsilon) + 1) + 1) + 0) + 1 \\ &= 2 \cdot (2 \cdot (2 \cdot (2 \cdot 0 + 1) + 1) + 0) + 1 \\ &= 2^3 \cdot 1 + 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 13 \end{aligned}$$

Binärdarstellung von Zahlen — Stellenwertsystem zur Basis 2

- Ziffernmenge $Z_2 = \{0, 1\}$
- definiere: $\mathrm{num}_2(0) = 0$ und $\mathrm{num}_2(1) = 1$ und

$$\mathrm{Num}_2(\varepsilon) = 0$$

$$\forall w \in Z_2^* \; \forall x \in Z_2 : \mathrm{Num}_2(wx) = 2 \cdot \mathrm{Num}_2(w) + \mathrm{num}_2(x)$$

- $$\begin{aligned} \mathrm{Num}_2(1101) &= 2 \cdot \mathrm{Num}_2(110) + 1 \\ &= 2 \cdot (2 \cdot \mathrm{Num}_2(11) + 0) + 1 \\ &= 2 \cdot (2 \cdot (2 \cdot \mathrm{Num}_2(1) + 1) + 0) + 1 \\ &= 2 \cdot (2 \cdot (2 \cdot (2 \cdot \mathrm{Num}_2(\varepsilon) + 1) + 1) + 0) + 1 \\ &= 2 \cdot (2 \cdot (2 \cdot (2 \cdot 0 + 1) + 1) + 0) + 1 \\ &= 2^3 \cdot 1 + 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 13 \end{aligned}$$

Binärdarstellung von Zahlen — Stellenwertsystem zur Basis 2

- Ziffernmenge $Z_2 = \{0, 1\}$
- definiere: $\mathrm{num}_2(0) = 0$ und $\mathrm{num}_2(1) = 1$ und

$$\mathrm{Num}_2(\varepsilon) = 0$$

$$\forall w \in Z_2^* \ \forall x \in Z_2 : \mathrm{Num}_2(wx) = 2 \cdot \mathrm{Num}_2(w) + \mathrm{num}_2(x)$$

- $$\begin{aligned}
\mathrm{Num}_2(1101) &= 2 \cdot \mathrm{Num}_2(110) + 1 \\
&= 2 \cdot (2 \cdot \mathrm{Num}_2(11) + 0) + 1 \\
&= 2 \cdot (2 \cdot (2 \cdot \mathrm{Num}_2(1) + 1) + 0) + 1 \\
&= 2 \cdot (2 \cdot (2 \cdot (2 \cdot \mathrm{Num}_2(\varepsilon) + 1) + 1) + 0) + 1 \\
&= 2 \cdot (2 \cdot (2 \cdot (2 \cdot 0 + 1) + 1) + 0) + 1 \\
&= 2^3 \cdot 1 + 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 13
\end{aligned}$$

Hexadezimaldarstellung oder Sedezimaldarstellung

- Ziffern $Z_{16} = \{\mathtt{0}, \mathtt{1}, \mathtt{2}, \mathtt{3}, \mathtt{4}, \mathtt{5}, \mathtt{6}, \mathtt{7}, \mathtt{8}, \mathtt{9}, \mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}, \mathtt{E}, \mathtt{F}\}$.

x	`0`	`1`	`2`	`3`	`4`	`5`	`6`	`7`
$\mathrm{num}_{16}(x)$	0	1	2	3	4	5	6	7
x	`8`	`9`	`A`	`B`	`C`	`D`	`E`	`F`
$\mathrm{num}_{16}(x)$	8	9	10	11	12	13	14	15

- Zuordnung von Wörtern zu Zahlen gegeben durch

$$\mathrm{Num}_{16}(\varepsilon) = 0$$
$$\forall w \in Z_{16}^* \forall x \in Z_{16} : \mathrm{Num}_{16}(wx) = 16 \cdot \mathrm{Num}_{16}(w) + \mathrm{num}_{16}(x)$$

- $\mathrm{Num}_{16}(\mathtt{20A}) = 2 \cdot 16^2 + 0 \cdot 16^1 + 10 \cdot 16^0 = 522$

Hexadezimaldarstellung oder Sedezimaldarstellung

- Ziffern $Z_{16} = \{\mathtt{0}, \mathtt{1}, \mathtt{2}, \mathtt{3}, \mathtt{4}, \mathtt{5}, \mathtt{6}, \mathtt{7}, \mathtt{8}, \mathtt{9}, \mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}, \mathtt{E}, \mathtt{F}\}$.

x	`0`	`1`	`2`	`3`	`4`	`5`	`6`	`7`
$\mathrm{num}_{16}(x)$	0	1	2	3	4	5	6	7
x	`8`	`9`	`A`	`B`	`C`	`D`	`E`	`F`
$\mathrm{num}_{16}(x)$	8	9	10	11	12	13	14	15

- Zuordnung von Wörtern zu Zahlen gegeben durch

$$\mathrm{Num}_{16}(\varepsilon) = 0$$
$$\forall w \in Z_{16}^* \forall x \in Z_{16} : \mathrm{Num}_{16}(wx) = 16 \cdot \mathrm{Num}_{16}(w) + \mathrm{num}_{16}(x)$$

- $\mathrm{Num}_{16}(\mathtt{20A}) = 2 \cdot 16^2 + 0 \cdot 16^1 + 10 \cdot 16^0 = 522$

Hexadezimaldarstellung oder Sedezimaldarstellung

- Ziffern $Z_{16} = \{\mathtt{0}, \mathtt{1}, \mathtt{2}, \mathtt{3}, \mathtt{4}, \mathtt{5}, \mathtt{6}, \mathtt{7}, \mathtt{8}, \mathtt{9}, \mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}, \mathtt{E}, \mathtt{F}\}$.

x	0	1	2	3	4	5	6	7
$\mathrm{num}_{16}(x)$	0	1	2	3	4	5	6	7
x	8	9	A	B	C	D	E	F
$\mathrm{num}_{16}(x)$	8	9	10	11	12	13	14	15

- Zuordnung von Wörtern zu Zahlen gegeben durch

$$\mathrm{Num}_{16}(\varepsilon) = 0$$

$$\forall w \in Z_{16}^* \forall x \in Z_{16} : \mathrm{Num}_{16}(wx) = 16 \cdot \mathrm{Num}_{16}(w) + \mathrm{num}_{16}(x)$$

- $\mathrm{Num}_{16}(\mathtt{20A}) = 2 \cdot 16^2 + 0 \cdot 16^1 + 10 \cdot 16^0 = 522$

Ein kleines Problem

- die Alphabete Z_2, Z_3, usw. sind nicht disjunkt
- Darstellungen mehrdeutig

$\mathrm{Num}_2(\mathtt{111})$	die Zahl sieben
$\mathrm{Num}_8(\mathtt{111})$	die Zahl dreiundsiebzig
$\mathrm{Num}_{10}(\mathtt{111})$	die Zahl einhundertelf
$\mathrm{Num}_{16}(\mathtt{111})$	die Zahl zweihundertdreiundsiebzig

- in *manchen* Programmiersprachen
  - Präfix `0b` für Darstellungen zur Basis 2
  - Präfix `0o` für Darstellungen zur Basis 8
  - Präfix `0x` für Darstellungen zur Basis 16

Achtung

- in Java, C und … ist das anders:
  - Zahldarstellungen *mit führender* `0` *sind immer oktal*
  - `012` bedeutet die Zahl zehn   und z. B. `089` ist illegal!

Wo sind wir?

Von Wörtern zu Zahlen und zurück
- Dezimaldarstellung von Zahlen
- Andere Zahldarstellungen
- Von Zahlen zu ihren Darstellungen

Notationen für Funktionen

Von einem Alphabet zum anderen

Huffman-Codierung

Zahldarstellung auch für negative Zahlen

Noch offene Frage: Ist jede Zahl darstellbar?

- eben: zu jedem Wort dargestellte Zahl definierbar
  (und „berechenbar“)
- nun: zu jeder Zahl eine Darstellung definierbar
  (und „berechenbar“)

k-äre Darstellung von Zahlen

- Es sei $k \in \mathbb{N}_0$ mit $k \geq 2$.
- Alphabet Z_k mit k Ziffern
  - Bedeutung: die Zahlen in $\mathbb{Z}_k$
  - für $i \in \mathbb{Z}_k$ sei $\text{repr}_k(i)$ das entsprechende Zeichen
  - repr_k ist also gerade die Umkehrfunktion zu num_k
- gesehen $\text{Num}_k : Z_k^* \longrightarrow \mathbb{N}_0$
- gesucht $\text{Repr}_k : \mathbb{N}_0 \longrightarrow Z_k^*$
  - Repräsentation von $n \in \mathbb{N}_0$ als $w \in Z_k^*$ mit $\text{Num}_k(w) = n$
  - für die naheliegende Definition von Num_k
- Redeweisen
  - *binäre* Darstellung falls $k = 2$
  - *ternäre* Darstellung falls $k = 3$

k-äre Darstellung von Zahlen

- Es sei $k \in \mathbb{N}_0$ mit $k \geq 2$.
- Alphabet Z_k mit k Ziffern
  - Bedeutung: die Zahlen in $\mathbb{Z}_k$
  - für $i \in \mathbb{Z}_k$ sei $\mathrm{repr}_k(i)$ das entsprechende Zeichen
  - repr_k ist also gerade die Umkehrfunktion zu num_k
- gesehen $\mathrm{Num}_k : Z_k^* \longrightarrow \mathbb{N}_0$
- gesucht $\mathrm{Repr}_k : \mathbb{N}_0 \longrightarrow Z_k^*$
  - Repräsentation von $n \in \mathbb{N}_0$ als $w \in Z_k^*$ mit $\mathrm{Num}_k(w) = n$
  - für die naheliegende Definition von Num_k
- Redeweisen
  - *binäre* Darstellung falls $k = 2$
  - *ternäre* Darstellung falls $k = 3$

Definition auf übernächster Folie

Division mit Rest: Operationen div und mod

- es sei $x \in \mathbb{N}_0$ und $y \in \mathbb{N}_+$
- $x \textbf{ div } y \in \mathbb{N}_0$
  - Ergebnis der ganzzahligen Division von x durch y
- $x \textbf{ mod } y \in \mathbb{N}_0$
  - Rest der ganzzahligen Division von x durch y
  - $0 \leq x \textbf{ mod } y < y$
- für jedes $x \in \mathbb{N}_0$ und jedes $y \in \mathbb{N}_+$ gilt

$$x = y \cdot (x \textbf{ div } y) + (x \textbf{ mod } y)$$

- Beispiele
  - $6 \textbf{ div } 2 = 3 \quad \text{und} \quad 6 \textbf{ mod } 2 = 0$
  - $7 \textbf{ div } 2 = 3 \quad \text{und} \quad 7 \textbf{ mod } 2 = 1$
  - $8 \textbf{ div } 2 = 4 \quad \text{und} \quad 8 \textbf{ mod } 2 = 0$

Division mit Rest: Operationen **div** und **mod**

- $x \ \mathbf{div}\ y \in \mathbb{N}_0$ Ergebnis der ganzzahligen Division
- $x \ \mathbf{mod}\ y \in \mathbb{N}_0$ Rest der ganzzahligen Division
  - $0 \leq x \ \mathbf{mod}\ y < y$
- für jedes $x \in \mathbb{N}_0$ und jedes $y \in \mathbb{N}_+$ gilt

$$x = y \cdot (x \ \mathbf{div}\ y) + (x \ \mathbf{mod}\ y)$$

Beispiele

- $2513 \ \mathbf{div}\ 10 = 251$ und $2513 \ \mathbf{mod}\ 10 = 3$
- $251 \ \mathbf{div}\ 10 = 25$ und $251 \ \mathbf{mod}\ 10 = 1$
- $25 \ \mathbf{div}\ 10 = 2$ und $25 \ \mathbf{mod}\ 10 = 5$
- $2 \ \mathbf{div}\ 10 = 0$ und $2 \ \mathbf{mod}\ 10 = 2$

k-äre Darstellung von Zahlen (2)

- Es sei $k \in \mathbb{N}_0$ und $k \geq 2$.

$$\mathrm{Repr}_k : \mathbb{N}_0 \to Z_k^*$$
$$n \mapsto \begin{cases} \mathrm{repr}_k(n) & \text{falls } n < k \\ \mathrm{Repr}_k(n \;\mathbf{div}\; k) \cdot \mathrm{repr}_k(n \;\mathbf{mod}\; k) & \text{falls } n \geq k \end{cases}$$

- kann man auch so schreiben

$$n \mapsto \begin{cases} \mathrm{repr}_k(n \;\mathbf{mod}\; k) & \text{falls } n < k \\ \mathrm{Repr}_k(n \;\mathbf{div}\; k) \cdot \mathrm{repr}_k(n \;\mathbf{mod}\; k) & \text{falls } n \geq k \end{cases}$$

- Ist das eine Definition?

Die Definition von Repr_k ist sinnvoll. (1)

- $\mathrm{Repr}_k : \mathbb{N}_0 \longrightarrow Z_k^*$

$$n \mapsto \begin{cases} \mathrm{repr}_k(n) & \text{falls } n < k \\ \mathrm{Repr}_k(n \mathbin{\mathbf{div}} k) \cdot \mathrm{repr}_k(n \mathbin{\mathbf{mod}} k) & \text{falls } n \geq k \end{cases}$$

- Behauptung: Für jedes $n \in \mathbb{N}_0$ ist $\mathrm{Repr}_k(n)$ definiert.

Die Definition von Repr_k ist sinnvoll. (1)

- $\mathrm{Repr}_k : \mathbb{N}_0 \to Z_k^*$

$$n \mapsto \begin{cases} \mathrm{repr}_k(n) & \text{falls } n < k \\ \mathrm{Repr}_k(n \textbf{ div } k) \cdot \mathrm{repr}_k(n \textbf{ mod } k) & \text{falls } n \geq k \end{cases}$$

- Behauptung: Für jedes $n \in \mathbb{N}_0$ ist $\mathrm{Repr}_k(n)$ definiert.
- Behauptung: Für jedes $m \in \mathbb{N}_+$ gilt:
  Für jedes $n \in \mathbb{N}_0$ mit $n < k^m$ ist $\mathrm{Repr}_k(n)$ definiert.
- Beweis durch vollständige Induktion

Die Definition von Repr_k ist sinnvoll. (2)

- $$\mathrm{Repr}_k(n) = \begin{cases} \mathrm{repr}_k(n) & \text{falls } n < k \\ \mathrm{Repr}_k(n \mathbin{\mathbf{div}} k) \cdot \mathrm{repr}_k(n \mathbin{\mathbf{mod}} k) & \text{falls } n \geq k \end{cases}$$
- Behauptung: Für jedes $m \in \mathbb{N}_+$ gilt:
  Für jedes $n \in \mathbb{N}_0$ mit $n < k^m$ ist $\mathrm{Repr}_k(n)$ definiert.

Die Definition von Repr_k ist sinnvoll. (2)

- $$\mathrm{Repr}_k(n) = \begin{cases} \mathrm{repr}_k(n) & \text{falls } n < k \\ \mathrm{Repr}_k(n \textbf{ div } k) \cdot \mathrm{repr}_k(n \textbf{ mod } k) & \text{falls } n \geq k \end{cases}$$
- Behauptung: Für jedes $m \in \mathbb{N}_+$ gilt:

$\mathcal{A}_m$ Für jedes $n \in \mathbb{N}_0$ mit $n < k^m$ ist $\mathrm{Repr}_k(n)$ definiert.

Die Definition von Repr_k ist sinnvoll. (2)

- $$\mathrm{Repr}_k(n) = \begin{cases} \mathrm{repr}_k(n) & \text{falls } n < k \\ \mathrm{Repr}_k(n \,\mathbf{div}\, k) \cdot \mathrm{repr}_k(n \,\mathbf{mod}\, k) & \text{falls } n \geq k \end{cases}$$
- Behauptung: Für jedes $m \in \mathbb{N}_+$ gilt:

$\mathcal{A}_m$ Für jedes $n \in \mathbb{N}_0$ mit $n < k^m$ ist $\mathrm{Repr}_k(n)$ definiert.

- *Ind.anfang:* $m = 1$, also $n < k$: ✓
- *Ind.Schritt:* sei $m \in \mathbb{N}_+$ beliebig aber nachfolgend fest
  - es gelte IV: für jedes $n \in \mathbb{N}_0$ mit $n < k^m$ ist $\mathrm{Repr}_k(n)$ definiert.
  - zeige IB: für jedes $n \in \mathbb{N}_0$ mit $n < k^{m+1}$ ist $\mathrm{Repr}_k(n)$ definiert.

Die Definition von Repr_k ist sinnvoll. (2)

- $$\mathrm{Repr}_k(n) = \begin{cases} \mathrm{repr}_k(n) & \text{falls } n < k \\ \mathrm{Repr}_k(n \;\mathbf{div}\; k) \cdot \mathrm{repr}_k(n \;\mathbf{mod}\; k) & \text{falls } n \geq k \end{cases}$$
- Behauptung: Für jedes $m \in \mathbb{N}_+$ gilt:

$\mathcal{A}_m$ Für jedes $n \in \mathbb{N}_0$ mit $n < k^m$ ist $\mathrm{Repr}_k(n)$ definiert.

- *Ind.anfang:* $m = 1$, also $n < k$: ✓
- *Ind.Schritt:* sei $m \in \mathbb{N}_+$ beliebig aber nachfolgend fest
  - es gelte IV: für jedes $n \in \mathbb{N}_0$ mit $n < k^m$ ist $\mathrm{Repr}_k(n)$ definiert.
  - zeige IB: für jedes $n \in \mathbb{N}_0$ mit $n < k^{m+1}$ ist $\mathrm{Repr}_k(n)$ definiert.
    - falls sogar $n < k^m$: ✓ wegen IV
    - falls $k^m \leq n < k^{m+1}$: dann $n \;\mathbf{div}\; k < k^m$, also nach IV $\mathrm{Repr}_k(n \;\mathbf{div}\; k)$ definiert, also auch $\mathrm{Repr}_k(n)$

Num_k ist linksinvers zu Repr_k (1)

Lemma

- $\mathrm{Num}_k(\mathrm{Repr}_k(n)) = n$
- Beweis: ähnliche vollständige Induktion wie eben

- „umgekehrt“ im allgemeinen $\mathrm{Repr}_k(\mathrm{Num}_k(w)) \neq w$
  weil „überflüssige“ führende Nullen wegfallen.

Num_k ist linksinvers zu Repr_k (1)

Karlsruher Institut für Technologie

Lemma

- $\mathrm{Num}_k(\mathrm{Repr}_k(n)) = n$
- Beweis: ähnliche vollständige Induktion wie eben
- „umgekehrt“ im allgemeinen $\mathrm{Repr}_k(\mathrm{Num}_k(w)) \neq w$ weil „überflüssige“ führende Nullen wegfallen.
- Das Wort linksinvers wird gleich noch erklärt

Num_k ist linksinvers zu Repr_k (2)

Lemma

- Für jedes $n \in \mathbb{N}_0$ ist $\mathrm{Num}_k(\mathrm{Repr}_k(n)) = n$ bzw.
  Für jedes $m \in \mathbb{N}_+$ gilt: $\forall n \in \mathbb{N}_0$ mit $n < k^m$ ist $\mathrm{Num}_k(\mathrm{Repr}_k(n)) = n$.

Num_k ist linksinvers zu Repr_k (2)

Lemma

- Für jedes $n \in \mathbb{N}_0$ ist $\mathrm{Num}_k(\mathrm{Repr}_k(n)) = n$ bzw.
  Für jedes $m \in \mathbb{N}_+$ gilt: $\forall n \in \mathbb{N}_0$ mit $n < k^m$ ist $\mathrm{Num}_k(\mathrm{Repr}_k(n)) = n$.

Beweis

- *IA:* $m = 1$: für jedes $n \in \mathbb{N}_0$ mit $n < k^1$: ✓

Num_k ist linksinvers zu Repr_k (2)

Lemma

- Für jedes $n \in \mathbb{N}_0$ ist $\mathrm{Num}_k(\mathrm{Repr}_k(n)) = n$ bzw.
  Für jedes $m \in \mathbb{N}_+$ gilt: $\forall n \in \mathbb{N}_0$ mit $n < k^m$ ist $\mathrm{Num}_k(\mathrm{Repr}_k(n)) = n$.

Beweis

- *IA:* $m = 1$: für jedes $n \in \mathbb{N}_0$ mit $n < k^1$: ✓
- *IS:* $m \rightsquigarrow m + 1$: es sei $m \in \mathbb{N}_+$ beliebig aber nachfolgend fest
- es gelte *IV:* für jedes $n \in \mathbb{N}_0$ mit $n < k^m$ ist $\mathrm{Num}_k(\mathrm{Repr}_k(n)) = n$
- zeige *IB:* für jedes $n \in \mathbb{N}_0$ mit $n < k^{m+1}$ ist $\mathrm{Num}_k(\mathrm{Repr}_k(n)) = n$.

Num_k ist linksinvers zu Repr_k (2)

Lemma

- Für jedes $n \in \mathbb{N}_0$ ist $\mathrm{Num}_k(\mathrm{Repr}_k(n)) = n$ bzw.
  Für jedes $m \in \mathbb{N}_+$ gilt: $\forall n \in \mathbb{N}_0$ mit $n < k^m$ ist $\mathrm{Num}_k(\mathrm{Repr}_k(n)) = n$.

Beweis

- *IA:* $m = 1$: für jedes $n \in \mathbb{N}_0$ mit $n < k^1$: ✓
- *IS:* $m \rightsquigarrow m+1$: es sei $m \in \mathbb{N}_+$ beliebig aber nachfolgend fest
- es gelte *IV:* für jedes $n \in \mathbb{N}_0$ mit $n < k^m$ ist $\mathrm{Num}_k(\mathrm{Repr}_k(n)) = n$
- zeige *IB:* für jedes $n \in \mathbb{N}_0$ mit $n < k^{m+1}$ ist $\mathrm{Num}_k(\mathrm{Repr}_k(n)) = n$.
  - falls sogar $n < k^m$: ✓ wegen IV

Num_k ist linksinvers zu Repr_k (2)

Lemma

- Für jedes $n \in \mathbb{N}_0$ ist $\mathrm{Num}_k(\mathrm{Repr}_k(n)) = n$ bzw.
  Für jedes $m \in \mathbb{N}_+$ gilt: $\forall n \in \mathbb{N}_0$ mit $n < k^m$ ist $\mathrm{Num}_k(\mathrm{Repr}_k(n)) = n$.

Beweis

- *IA:* $m = 1$: für jedes $n \in \mathbb{N}_0$ mit $n < k^1$: ✓
- *IS:* $m \rightsquigarrow m+1$: es sei $m \in \mathbb{N}_+$ beliebig aber nachfolgend fest
- es gelte *IV:* für jedes $n \in \mathbb{N}_0$ mit $n < k^m$ ist $\mathrm{Num}_k(\mathrm{Repr}_k(n)) = n$
- zeige *IB:* für jedes $n \in \mathbb{N}_0$ mit $n < k^{m+1}$ ist $\mathrm{Num}_k(\mathrm{Repr}_k(n)) = n$.
  - falls sogar $n < k^m$: ✓ wegen IV
  - falls $k^m \leq n < k^{m+1}$: dann $n \mathbin{\mathbf{div}} k < k^m$, also
    nach IV $\mathrm{Num}_k(\mathrm{Repr}_k(n \mathbin{\mathbf{div}} k)) = n \mathbin{\mathbf{div}} k$,

Num_k ist linksinvers zu Repr_k (2)

Karlsruher Institut für Technologie

Lemma

- Für jedes $n \in \mathbb{N}_0$ ist $\mathrm{Num}_k(\mathrm{Repr}_k(n)) = n$ bzw.
  Für jedes $m \in \mathbb{N}_+$ gilt: $\forall n \in \mathbb{N}_0$ mit $n < k^m$ ist $\mathrm{Num}_k(\mathrm{Repr}_k(n)) = n$.

Beweis

- *IA:* $m = 1$: für jedes $n \in \mathbb{N}_0$ mit $n < k^1$: ✓
- *IS:* $m \rightsquigarrow m+1$: es sei $m \in \mathbb{N}_+$ beliebig aber nachfolgend fest
- es gelte *IV:* für jedes $n \in \mathbb{N}_0$ mit $n < k^m$ ist $\mathrm{Num}_k(\mathrm{Repr}_k(n)) = n$
- zeige *IB:* für jedes $n \in \mathbb{N}_0$ mit $n < k^{m+1}$ ist $\mathrm{Num}_k(\mathrm{Repr}_k(n)) = n$.
  - falls sogar $n < k^m$: ✓ wegen IV
  - falls $k^m \leq n < k^{m+1}$: dann $n \mathbin{\mathbf{div}} k < k^m$, also
    nach IV $\mathrm{Num}_k(\mathrm{Repr}_k(n \mathbin{\mathbf{div}} k)) = n \mathbin{\mathbf{div}} k$, also auch

$$\begin{aligned}
\mathrm{Num}_k(\mathrm{Repr}_k(n)) &= \mathrm{Num}_k(\mathrm{Repr}_k(k(n \mathbin{\mathbf{div}} k) + (n \mathbin{\mathbf{mod}} k))) \\
&= \mathrm{Num}_k(\mathrm{Repr}_k(n \mathbin{\mathbf{div}} k) \cdot \mathrm{repr}_k(n \mathbin{\mathbf{mod}} k)) \\
&= k \cdot \mathrm{Num}_k(\mathrm{Repr}_k(n \mathbin{\mathbf{div}} k)) + \mathrm{num}_k(\mathrm{repr}_k(n \mathbin{\mathbf{mod}} k)) \\
\text{nach IV} \qquad &= k(n \mathbin{\mathbf{div}} k) + (n \mathbin{\mathbf{mod}} k) = n
\end{aligned}$$

Mathe-Notation vs. Informatik-Notation

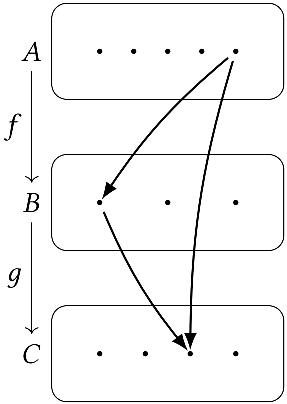
- In der Mathematik (und theo. Informatik) wird „modulo“ üblicherweise anders notiert
- “Der ganzzahlige Rest von 42 beim Teilen durch 18 ist 6.”
- **Informatik:** $42 \mathbin{\mathbf{mod}} 18 = 6$
- **Mathematik:** $42 \equiv 6 \pmod{18}$
- In der Mathematik gilt auch $42 \equiv 24 \pmod{18}$, in der Informatik gilt $42 \mathbin{\mathbf{mod}} 18 = 24$ **nicht**.

Das ist wichtig

- Das sollten Sie mitnehmen:
  - Umwandlungen zwischen Zahlen und Wörtern
  - Induktion ist ein „scharfes Schwert“ für Definitionen und Beweise
  - schon Leibniz kannte die Binärdarstellung
- Das sollten Sie üben:
  - selbe Zahlen verschieden repräsentieren
  - Definitionen auch in Randfällen ausprobieren
  - starke Induktion (aber das hatten wir ja schon …)

Wo sind wir?

Komposition von Funktionen



- für Mengen A, B und C und Funktionen $f: A \longrightarrow B$ und $g: B \longrightarrow C$
- $g \circ f: A \longrightarrow C : a \mapsto g(f(a))$
  - *Reihenfolge beachten* bei $g \circ f$!
- Funktionskomposition ist assoziativ: für $f: A \longrightarrow B$, $g: B \longrightarrow C$, $h: C \longrightarrow D$ ist
  $$h \circ (g \circ f) = (h \circ g) \circ f$$
- Funktionskomposition nicht kommutativ: z. B. für $f: \mathbb{N}_0 \longrightarrow \mathbb{N}_0 : n \mapsto n^2$ und $g: \mathbb{N}_0 \longrightarrow \mathbb{N}_0 : n \mapsto n+1$: $g(f(3)) = 3^2 + 1 = 10$ aber $f(g(3)) = (3+1)^2 = 16$

Identität — die neutralen Elemente bzgl. Komposition

- für Menge M heißt
  $\mathrm{I}_M : M \to M : x \mapsto x$
  identische Funktion auf M

- für jede Funktion $f : A \to B$ gilt:

$$\mathrm{I}_B \circ f = f \qquad \text{und} \qquad f \circ \mathrm{I}_A = f$$

Umkehrfunktion

- sei nun $f: A \longrightarrow B$ und $g: B \longrightarrow A$
  dann sowohl $g \circ f : A \longrightarrow A$ als auch $f \circ g : B \longrightarrow B$ definiert
- g *Umkehrfunktion zu f* oder *inverse Funktion*
  falls *sowohl* $g \circ f = \mathrm{I}_A$ *als auch* $f \circ g = \mathrm{I}_B$
- auch geschrieben $g = f^{-1}$
- dann auch f Umkehrfunktion zu g

Links- und Rechtsinverse

- sei $f\colon A \longrightarrow B$ und $g\colon B \longrightarrow A$
- g *Linksinverse zu* f
  falls $g \circ f = \mathrm{I}_A$
- g *Rechtsinverse zu* f falls $f \circ g = \mathrm{I}_B$

Man kann beweisen:

- f hat genau dann eine Linksinverse, wenn f injektiv ist.
- f hat genau dann eine Rechtsinverse, wenn f surjektiv ist.
- Wenn f eine Links- und eine Rechtsinverse besitzt,
  dann sind sie gleich.
- Also hat ein f höchstens *eine* Umkehrfunktion.
- f hat genau dann eine Umkehrfunktion, wenn f bijektiv ist.

Wo sind wir?

Wo sind wir?

Von Hexadezimal- zu Binärdarstellung

- betrachte $\mathrm{Trans}_{2,16} : Z_{16}^* \to Z_2^*$ mit

  $\mathrm{Trans}_{2,16} = \mathrm{Repr}_2 \circ \mathrm{Num}_{16}$, also
  für jedes $w \in Z_{16}$: $\mathrm{Trans}_{2,16}(w) = \mathrm{Repr}_2(\mathrm{Num}_{16}(w))$

- Beispiel
  $$\begin{aligned}\mathrm{Trans}_{2,16}(\mathtt{A3}) &= \mathrm{Repr}_2(\mathrm{Num}_{16}(\mathtt{A3})) \\ &= \mathrm{Repr}_2(163) = \mathtt{10100011}\end{aligned}$$

- wesentlicher Punkt:
  - `A3` und `10100011` haben *die gleiche Bedeutung* nämlich die Zahl einhundertdreiundsechzig

- So etwas wollen wir eine Übersetzung nennen.

Von Hexadezimal- zu Binärdarstellung

$$\begin{tikzcd}
Z_{16}^* \arrow[rr, "\mathrm{Trans}_{2,16}"] \arrow[dr, "\mathrm{Num}_{16}"'] & \circ & Z_2^* \\
& \mathbb{N}_0 \arrow[ur, "\mathrm{Repr}_2"'] &
\end{tikzcd}$$

- betrachte $\mathrm{Trans}_{2,16} : Z_{16}^* \to Z_2^*$ mit

  $\mathrm{Trans}_{2,16} = \mathrm{Repr}_2 \circ \mathrm{Num}_{16}$, also
  für jedes $w \in Z_{16}$: $\mathrm{Trans}_{2,16}(w) = \mathrm{Repr}_2(\mathrm{Num}_{16}(w))$
- Beispiel
  $\mathrm{Trans}_{2,16}(\texttt{A3}) = \mathrm{Repr}_2(\mathrm{Num}_{16}(\texttt{A3}))$
  $= \mathrm{Repr}_2(163) = \texttt{10100011}$
- wesentlicher Punkt:
  - `A3` und `10100011` haben *die gleiche Bedeutung*
    nämlich die Zahl einhundertdreiundsechzig
- So etwas wollen wir eine Übersetzung nennen.

Übersetzungen — bedeutungserhaltende Abbildungen von Wörtern auf Wörter

- Wörter aus Sprache $L_A \subseteq A^*$ haben meist Bedeutung.
- Sem Menge von Bedeutungen
  - Zahlen
  - Ausführungen von Java-Programmen
  - ...

Übersetzungen — bedeutungserhaltende Abbildungen von Wörtern auf Wörter

$$\begin{tikzcd}
L_A \arrow[rr, "f"] \arrow[dr, "\mathrm{sem}_A"'] & \arrow[d, phantom, "\circ"] & L_B \arrow[dl, "\mathrm{sem}_B"] \\
& \mathrm{Sem} &
\end{tikzcd}$$

- Wörter aus Sprache $L_A \subseteq A^*$ haben meist Bedeutung.
- Sem Menge von Bedeutungen
  - Zahlen
  - Ausführungen von Java-Programmen
  - ...
- Gegeben:
  - Alphabete A und B
  - $L_A \subseteq A^*$ und $L_B \subseteq B^*$
  - Abbildungen $\mathrm{sem}_A \colon L_A \to \mathrm{Sem}$ und $\mathrm{sem}_B \colon L_B \to \mathrm{Sem}$
- $f : L_A \to L_B$ heiße eine Übersetzung wenn $\mathrm{sem}_A = \mathrm{sem}_B \circ f$ also

$$\forall w \in L_A : \mathrm{sem}_A(w) = \mathrm{sem}_B(f(w))$$

$\mathrm{Trans}_{2,16}$ — eine Übersetzung

- $\mathrm{Trans}_{2,16} = \mathrm{Repr}_2 \circ \mathrm{Num}_{16}$.
- einfacher Fall: $L_A = A^* = Z_{16}^*$ und $L_B = B^* = Z_2^*$ und $\mathrm{Sem} = \mathbb{N}_0$
- Bedeutungsfunktionen: $\mathrm{sem}_A = \mathrm{Num}_{16}$ und $\mathrm{sem}_B = \mathrm{Num}_2$
- Nachrechnen, dass $\mathrm{Trans}_{2,16}$ eine Übersetzung ist:

$$\begin{aligned}
\mathrm{sem}_B(f(w)) &= \mathrm{Num}_2(\mathrm{Trans}_{2,16}(w)) \\
&= \mathrm{Num}_2(\mathrm{Repr}_2(\mathrm{Num}_{16}(w))) \\
&= \mathrm{Num}_{16}(w) \\
&= \mathrm{sem}_A(w)
\end{aligned}$$

Wozu Übersetzungen

Wozu Übersetzungen

- Legalität: nur bestimmter Zeichensatz erlaubt
- Lesbarkeit: vergleiche `A3` mit `10100011`
- Verschlüsselung: nicht verbesserte Lesbarkeit, sondern
  - gar keine Lesbarkeit … für Außenstehende
  - siehe Vorlesungen über Kryptographie
- Kompression: Übersetzungen können kürzer sein
  - und zwar *ohne* größeres Alphabet
  - in diesem Kapitel: Huffman-Codes
- Fehlererkennung und Fehlerkorrektur:
  - durch Übersetzung Text länger so, dass
  - u. U. Fehlererkennung oder gar Fehlerkorrektur möglich
  - siehe Vorlesungen über Codierungstheorie

Codierungen — injektive Übersetzungen

$$\begin{tikzcd} L_A \arrow[rr, "f"] \arrow[dr, "\mathrm{sem}_A"'] & \arrow[d, phantom, "\circ"] & L_B \arrow[dl, "\mathrm{sem}_B"] \\ & \mathrm{Sem} & \end{tikzcd}$$

- $\mathrm{sem}_A(w) = \mathrm{sem}_B(f(w))$ manchmal kein Problem
  - Verschlüsselung, manche Kompressionsverfahren
- wenn f injektiv
  - von $f(x)$ *eindeutig* zurück zu x
  - dann sem_B *definierbar* durch $\mathrm{sem}_B(f(x)) = \mathrm{sem}_A(x)$
    - für $y \in L_B \setminus f(L_A)$ ist $\mathrm{sem}_B(y)$ «gleichgültig»
- Codierung: Übersetzung mit injektivem f
- Codewörter: die $f(w)$
- Code: $\{f(w) \mid w \in L_A\}$

Wie spezifiziert man eine Übersetzung?

- wenn L_A unendlich
  - kann man nicht alle Möglichkeiten aufzählen …
- Auswege:
  - Homomorphismen
  - Block-Codierungen

Wo sind wir?

Homomorphismen —
mit Konkatenation verträgliche Abbildungen

Karlsruher Institut für Technologie

- gegeben Abbildung $h : A^* \longrightarrow B^*$
  - für Alphabete A und B
- h Homomorphismus, falls für alle $w_1, w_2 \in A^*$ gilt:

$$h(w_1 \cdot w_2) = h(w_1) \cdot h(w_2)$$

- Beispiel:
  - es sei h Homomorphismus, $h(\mathtt{a}) = \mathtt{10}$ und $h(\mathtt{b}) = \mathtt{001}$
  - dann $h(\mathtt{bab}) = h(\mathtt{ba}) \cdot h(\mathtt{b}) = h(\mathtt{b}) \cdot h(\mathtt{a}) \cdot h(\mathtt{b})$

    $= \mathtt{001} \cdot \mathtt{10} \cdot \mathtt{001}$
- Homomorphismus ε-frei, wenn für jedes $x \in A : h(x) \neq \varepsilon$.

Homomorphismen lassen das leere Wort unverändert

- Homomorphismus $h : A^* \longrightarrow B^*$,
  also $\forall w_1, w_2 \in A^*$:
  $$h(w_1 \cdot w_2) = h(w_1) \cdot h(w_2) \ .$$
- kurze Überlegung
  - $h(\varepsilon) = h(\varepsilon\varepsilon) = h(\varepsilon)h(\varepsilon)$
  - also $|h(\varepsilon)| = |h(\varepsilon)| + |h(\varepsilon)|$
  - also $|h(\varepsilon)| = 0$
  - also $h(\varepsilon) = \varepsilon$

Homomorphismen —
die Bilder einzelner Symbole legen alles fest

Karlsruher Institut für Technologie

Lemma

- Es seien A und B Alphabete und
  $h : A^* \longrightarrow B^*$ und $g : A^* \longrightarrow B^*$ Homomorphismen.
  Wenn für jedes $x \in A$ gilt, dass $h(x) = g(x)$ ist,
  dann gilt für jedes $w \in A^*$, dass $h(w) = g(w)$ ist.

Beweis

- vollständige Induktion über die Wortlänge

- *Induktionsanfang:* $w = \varepsilon$ $\qquad (|w| = 0)$
  - $h(\varepsilon) = \varepsilon = g(\varepsilon)$
- *Induktionsschritt:* $\qquad (|w| = n \leadsto |w| = n + 1)$
  Es seien $w \in A^*$ und $x \in A$ beliebig, nachfolgend fest
- gelte *IV:* $h(w) = g(w)$.

Homomorphismen — die Bilder einzelner Symbole legen alles fest (2)

- seien $w \in A^*$ und $x \in A$ und gelte IV $h(w) = g(w)$
- zeige IB: $h(wx) = g(wx)$
- so:

$$\begin{aligned} h(wx) &= h(w)h(x) && \text{da } h \text{ Homomorphismus} \\ &= g(w)h(x) && \text{Induktionsvoraussetzung} \\ &= g(w)g(x) && \text{Voraussetzung des Lemmas} \\ &= g(wx) && \text{da } g \text{ Homomorphismus} \end{aligned}$$

Homomorphismen —
so legen die Bilder einzelner Symbole alles fest

- Es seien A und B Alphabete und $f : A \to B^*$.
- definiere $f^{**} : A^* \to B^*$ vermöge

$$f^{**}(\varepsilon) = \varepsilon$$

$$\text{für jedes } w \in A^*, \text{für jedes } x \in A : f^{**}(wx) = f^{**}(w) f(x)$$

Lemma

- f^{**} ist ein Homomorphismus.
  - Beweis durch vollständige Induktion …
- f^{**} heißt der durch f induzierte Homomorphismus

Ein Präfix eines Wortes ist ein Anfangsstück

- Es sei A ein Alphabet und $w \in A^*$.
- Ein Wort $v \in A^*$ ist ein Präfix von w, wenn es ein $u \in A^*$ gibt so, dass $vu = w$ ist.
- Ein Wort $u \in A^*$ ist ein Suffix von w, wenn es ein $v \in A^*$ gibt so, dass $vu = w$ ist.

- z. B.
  `Verdi` ist Präfix von `Verdichtung`
  ε ist Präfix von `Klausur`

  `pferde` ist Suffix von `Blumentopferde`
  ε ist Suffix von `Klausur`

Präfixfreie Codes

$h(0) = 100$
$h(1) = 1010$

$\cancel{h(0) = 10}$
$\cancel{h(1) = 1011}$

- gegeben Homomorphismus $h\colon A^* \longrightarrow B^*$
- Frage: Ist h injektiv?
- im allgemeinen nicht ganz einfach zu sehen
- einfacher Spezialfall: h ist präfixfrei, d. h.
  - für *keine zwei verschiedene* Symbole $x_1, x_2 \in A$ mit $x_1 \neq x_2$ gilt: $h(x_1)$ ist ein Präfix von $h(x_2)$.
  - äquivalent: wenn $h(x_1)$ Präfix von $h(x_2)$, dann $x_1 = x_2$, also sogar $h(x_1) = h(x_2)$
- gleich: präfixfreie Codes *sind* injektiv

Präfixfreie Codes: Decodierung

Karlsruher Institut für Technologie

Problem

- nicht alle $w \in B^*$ sind Codewort
  - d. h. h ist im allgemeinen nicht surjektiv

Ausweg

- damit Decodierung u trotzdem total
  - definiere $u : B^* \to (A \cup \{\perp\})^*$.
  - wenn $w \in B^*$ nicht Codewort,
    dann soll in $u(w)$ das Symbol $\perp$ vorkommen: „undefiniert“

Beispiel

- Homomorphismus $h : \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}^* \to \{\mathtt{0}, \mathtt{1}\}^*$ mit
  $h(\mathtt{a}) = \mathtt{1}$, $h(\mathtt{b}) = \mathtt{01}$, $h(\mathtt{c}) = \mathtt{001}$ präfixfrei!

$\forall v \in A^* :$
$u(h(v)) = v$

- gesucht $u : \{\mathtt{0}, \mathtt{1}\}^* \to \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \perp\}^*$ linksinvers zu h
- es soll z. B. gelten:
  - $u(\mathtt{001}) = \mathtt{c}$
  - $u(\mathtt{0101}) = \mathtt{bb}$
  - $u(\mathtt{0}) = \perp$ etc.

Präfixfreie Codes: Decodierung (2)

Karlsruher Institut für Technologie

- wir schreiben mal hin (und diskutieren das anschließend):

$h(\mathtt{a}) = \mathtt{1}$
$h(\mathtt{b}) = \mathtt{01}$
$h(\mathtt{c}) = \mathtt{001}$

$$u(w) = \begin{cases} \varepsilon, & \text{falls } w = \varepsilon \\ \mathtt{a}u(w'), & \text{falls } w = \mathtt{1}w' \\ \mathtt{b}u(w'), & \text{falls } w = \mathtt{01}w' \\ \mathtt{c}u(w'), & \text{falls } w = \mathtt{001}w' \\ \bot, & \text{sonst} \end{cases}$$

- sei $w = \mathtt{100101} = h(\mathtt{acb})$; dann

$$\begin{aligned} u(\mathtt{100101}) &= \mathtt{a}u(\mathtt{00101}) \\ &= \mathtt{ac}u(\mathtt{01}) \\ &= \mathtt{acb}u(\varepsilon) \\ &= \mathtt{acb} \end{aligned}$$

Präfixfreie Codes: Decodierung (3)

$h(\mathtt{a}) = \mathtt{1}$
$h(\mathtt{b}) = \mathtt{01}$
$h(\mathtt{c}) = \mathtt{001}$

- $u(w) = \begin{cases} \varepsilon, & \text{falls } w = \varepsilon \\ \mathtt{a}u(w'), & \text{falls } w = \mathtt{1}w' \\ \mathtt{b}u(w'), & \text{falls } w = \mathtt{01}w' \\ \mathtt{c}u(w'), & \text{falls } w = \mathtt{001}w' \\ \perp, & \text{sonst} \end{cases}$
- $u(\mathtt{100101}) = \mathtt{a}u(\mathtt{00101}) = \mathtt{ac}u(\mathtt{01}) = \mathtt{acb}u(\varepsilon) = \mathtt{acb}$
- Warum hat das geklappt?

Präfixfreie Codes: Decodierung (3)

$h(\mathtt{a}) = \mathtt{1}$
$h(\mathtt{b}) = \mathtt{01}$
$h(\mathtt{c}) = \mathtt{001}$

- $u(w) = \begin{cases} \varepsilon, & \text{falls } w = \varepsilon \\ \mathtt{a}u(w'), & \text{falls } w = \mathtt{1}w' \\ \mathtt{b}u(w'), & \text{falls } w = \mathtt{01}w' \\ \mathtt{c}u(w'), & \text{falls } w = \mathtt{001}w' \\ \perp, & \text{sonst} \end{cases}$
- $u(\mathtt{100101}) = \mathtt{a}u(\mathtt{00101}) = \mathtt{ac}u(\mathtt{01}) = \mathtt{acb}u(\varepsilon) = \mathtt{acb}$
- Warum hat das geklappt?
- In jedem Schritt war klar,
  welche Zeile der Definition von u anzuwenden war …

Präfixfreie Codes: Decodierung (4)

- Man spricht hier von Wohldefiniertheit
- Problem, wenn Funktionswert potenziell „auf mehreren Wegen“ festgelegt
- dann klar machen:
  - entweder nur ein Weg „gangbar“
  - oder auf allen Wegen gleicher Funktionswert
- *präfixfreien* Code kann man so decodieren:

$$u(w) = \begin{cases} \varepsilon & \text{falls } w = \varepsilon \\ x\,u(w') & \text{falls } w = h(x)w' \text{ für ein } x \in A \\ \perp & \text{sonst} \end{cases}$$

- „so einfach“ geht das nur für *präfixfreie* Codes

Wo sind wir?

Von Wörtern zu Zahlen und zurück

Notationen für Funktionen

Von einem Alphabet zum anderen
- Übersetzung von Zahldarstellungen
- Homomorphismen
- Beispiel Unicode: UTF-8

Huffman-Codierung

Zahldarstellung auch für negative Zahlen

UTF-8 Codierung von Unicode — ein Homomorphismus

- Codierung einzelner Zeichen: kommt gleich
- Wörter werden zeichenweise codiert.
- UTF-8 ist präfixfrei

UTF-8 — Auszug aus RFC 3629

Char. number range (hexadecimal)	UTF-8 octet sequence
0000 0000 - 0000 007F	0$xxxxxxx$
0000 0080 - 0000 07FF	110$xxxxx$ 10$xxxxxx$
0000 0800 - 0000 FFFF	1110$xxxx$ 10$xxxxxx$ 10$xxxxxx$
0001 0000 - 0010 FFFF	11110xxx 10$xxxxxx$ 10$xxxxxx$ 10$xxxxxx$

präfixfrei

- Determine the number of octets required …
- Prepare the high-order bits of the octets …
- Fill in the bits marked x …
  - Start by putting the lowest-order bit of the character number in the lowest-order position of the last octet of the sequence,
  - then …
  - When last octet filled in, move on to the next to last octet, …

Beispiel: UTF-8 Codierung des Integralzeichens $\int$

- U+222B: das Zeichen aus dem Unicode-Alphabet, dessen Codepoint hexadezimal repräsentiert 222B ist
- UTF-8 Codierung: benutze die Zeile

Char. number range	UTF-8 octet seq.
0000 0800 - 0000 FFFF	1110$xxxx$ 10$xxxxxx$ 10$xxxxxx$

- 0x222B in Bits 0010 0010 0010 1011
- also 0010 0010 0010 1011 = 0010 001000 101011
- also UTF-8 Codierung 11100010 10001000 10101011

Das ist wichtig

- Das sollten Sie mitnehmen:
  - Übersetzungen sind in verschiedenen Situationen nützlich
  - Homomorphismen
  - Codes
  - UTF-8
- Das sollten Sie üben:
  - Homomorphismen anwenden
  - Decodieren
  - (vielleicht mal ein) Zeichen in UTF-8 codieren

Wo sind wir?

Von Wörtern zu Zahlen und zurück

Notationen für Funktionen

Von einem Alphabet zum anderen

Huffman-Codierung

Zahldarstellung auch für negative Zahlen

Huffman-Codierung — ein Überblick

- gegeben: Alphabet A und Wort $w \in A^*$
- Eigenschaften der zu w gehörigen Huffman-Codierung
  - eine Abbildung $h : A^* \to \{0, 1\}^*$,
  - ε-freier Homomorphismus
  - injektiv
  - h typischerweise auf w angewendet
- Bestandteil z. B. von Kompressionsverfahren gzip, bzip2
- bei Huffman-Codierungen werden
  - häufigere Symbole durch kürzere Wörter codiert und
  - seltenere Symbole durch längere

Wo sind wir?

Wie oft kommt Symbol x in Wort w vor?

- $$N_x(\varepsilon) = 0$$
  $$\forall y \in A\colon \forall w \in A^*\colon N_x(yw) = \begin{cases} 1 + N_x(w) & \text{falls } y = x \\ N_x(w) & \text{falls } y \neq x \end{cases}$$

Beispiel

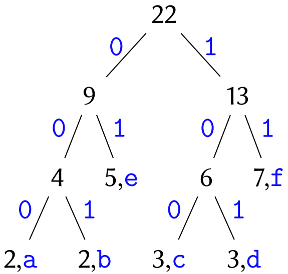
- $$\begin{aligned} N_{\mathtt{a}}(\mathtt{baaba}) &= N_{\mathtt{a}}(\mathtt{aaba}) \\ &= 1 + N_{\mathtt{a}}(\mathtt{aba}) \\ &= 2 + N_{\mathtt{a}}(\mathtt{ba}) \\ &= 2 + N_{\mathtt{a}}(\mathtt{a}) \\ &= 3 + N_{\mathtt{a}}(\varepsilon) \\ &= 3 \end{aligned}$$

Huffman-Codierung: der Plan

- Gegeben
  - $w \in A^*$ und
  - die Anzahlen $N_x(w)$ der Vorkommen aller $x \in A$ in w
  - o. B. d. A. alle $N_x(w) > 0$
    - «o. B. d. A. » bedeutet «ohne Beschränkung der Allgemeinheit»
- Bestimmung eines Huffman-Codes in zwei Phasen
  1. Konstruktion eines «Baumes»
     - Blätter entsprechen den $x \in A$, die in w vorkommen, und
     - Kanten mit 0 und 1 beschriftet
  2. Ablesen der Codes aus dem Baum (Pfadbeschriftungen)

Algorithmus für Huffman-Codes

- Beispiel: w = `afebfecaffdeddccefbeff`
- Baum am Ende:



- Homomorphismus (präfixfrei!)

x	a	b	c	d	e	f
h(x)	000	001	100	101	01	11

Konstruktion des Huffman-Baumes (1)

- zu jedem Zeitpunkt
  - Menge M_i
    „zu betrachtender Symbolmengen mit ihren Häufigkeiten“
  - ebenso große Menge schon konstruierter Teilbäume
- Initialisierung:
  - M_0 ist die Menge aller $\{(N_x(w), \{x\})\}$ für $x \in A$,
  - Als Anfang für die Konstruktion des Baumes zeichnet man für jedes Symbol einen Knoten mit Markierung $(N_x(w), \{x\})$.
- Beispiel

$$M_0 = \{\ (2, \{\texttt{a}\}),\ (2, \{\texttt{b}\}),\ (3, \{\texttt{c}\}),\ (3, \{\texttt{d}\}),\ (5, \{\texttt{e}\}),\ (7, \{\texttt{f}\})\ \}$$

Konstruktion des Huffman-Baumes (2)

- Anfang im Beispiel:

5,e 7,f

2,a 2,b 3,c 3,d

Konstruktion des Huffman-Baumes (3)

- Iterationsschritt des Algorithmus:
  - Solange M_i noch mindestens zwei Elemente enthält, bestimme M_{i+1} wie folgt:
    - wähle zwei Paare (k_1, X_1) und (k_2, X_2) mit kleinsten Häufigkeiten
    - ersetze diese Paare durch $(k_1 + k_2, X_1 \cup X_2)$

$$
\begin{aligned}
M_{i+1} = M_i \setminus \{\ (k_1, X_1),\ (k_2, X_2)\ \} \\
\cup \{\ (k_1 + k_2, X_1 \cup X_2)\ \}
\end{aligned}
$$

  - im Graphen
    - neuer Knoten
    - markiert mit Häufigkeit $k_1 + k_2$ und
    - Kanten zu den Knoten für (k_1, X_1) und (k_2, X_2)

Konstruktion des Huffman-Baumes (4)

- Beispiel

$$M_0 = \{\ (2, \{\mathtt{a}\}),\ (2, \{\mathtt{b}\}),\ (3, \{\mathtt{c}\}),\ (3, \{\mathtt{d}\}),\ (5, \{\mathtt{e}\}),\ (7, \{\mathtt{f}\})\ \}$$

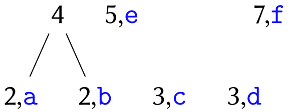
5,e 7,f

2,a 2,b 3,c 3,d

Konstruktion des Huffman-Baumes (4)

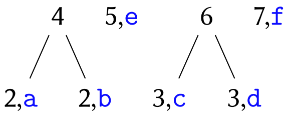
- Beispiel

$$M_1 = \{\ (4, \{\texttt{a}, \texttt{b}\}),\ (3, \{\texttt{c}\}),\ (3, \{\texttt{d}\}),\ (5, \{\texttt{e}\}),\ (7, \{\texttt{f}\})\ \}$$



- Beispiel

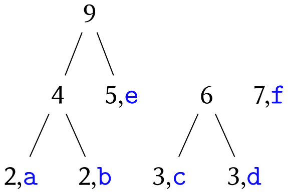
$$M_2 = \{\ (4, \{\texttt{a}, \texttt{b}\}),\ (6, \{\texttt{c}, \texttt{d}\}),\ (5, \{\texttt{e}\}),\ (7, \{\texttt{f}\})\ \}$$



Konstruktion des Huffman-Baumes (6)

- Beispiel

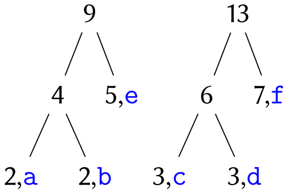
$$M_3 = \{\ (9, \{\mathtt{a}, \mathtt{b}, \mathtt{e}\}),\ (6, \{\mathtt{c}, \mathtt{d}\}),\ (7, \{\mathtt{f}\})\ \}$$



Konstruktion des Huffman-Baumes (7)

- Beispiel

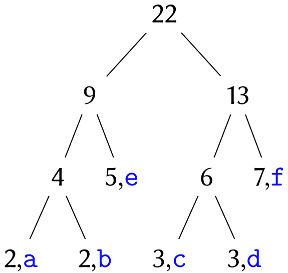
$$M_4 = \{ \ (9, \{\mathtt{a}, \mathtt{b}, \mathtt{e}\}), \ \ (13, \{\mathtt{c}, \mathtt{d}, \mathtt{f}\}) \ \}$$



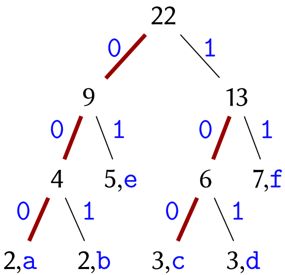
Konstruktion des Huffman-Baumes (8)

- Beispiel

$$M_5 = \{\ (22, \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}, \texttt{e}, \texttt{f}\})\ \}$$

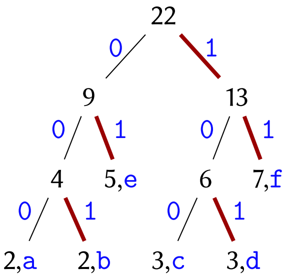


Beschriftung der Kanten



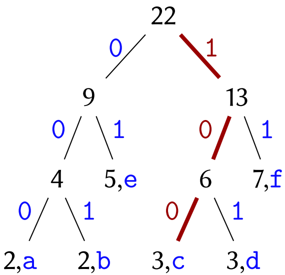
- nach links führende Kanten mit 0 beschriftet
- nach rechts führende Kanten mit 1 beschriftet

Beschriftung der Kanten



- nach links führende Kanten mit 0 beschriftet
- nach rechts führende Kanten mit 1 beschriftet

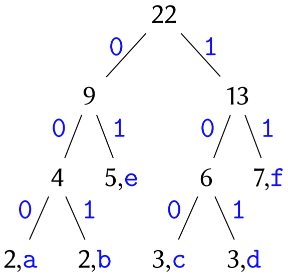
Ablesen der Codierungen



- gehe auf kürzestem Weg
  - von der «Wurzel» des Baumes (ganz oben)
  - zu dem Blatt für x
- konkateniere der Reihe nach
  - alle Symbole an den
  - Kanten auf diesem Weg
- $h(\mathtt{c}) = \mathtt{100}$

Algorithmus für Huffman-Codes

- Beispiel: w = afebfecaffdeddccefbeff
- Baum am Ende:



- Homomorphismus (präfixfrei!)

x	a	b	c	d	e	f
h(x)	000	001	100	101	01	11

Wo sind wir?

Von Wörtern zu Zahlen und zurück

Notationen für Funktionen

Von einem Alphabet zum anderen

Huffman-Codierung

- Algorithmus zur Berechnung von Huffman-Codes
- Weiteres zu Huffman-Codes

Zahldarstellung auch für negative Zahlen

Eigenschaften von Huffman-Codes

- Huffman-Code nicht eindeutig
  - im allgemeinen mehrere Möglichkeiten, welche zwei Knoten vereinigt werden
  - im Baum links und rechts vertauschbar
- aber alle sind „gleich gut“:
  - Unter allen präfixfreien Codes führen Huffman-Codes zu kürzesten Codierungen *des Wortes, für das die Huffman-Codierung konstruiert wurde.*

Block-Codierungen

- Verallgemeinerung des obigen Verfahrens:
  - Betrachte nicht Häufigkeiten einzelner Symbole,
  - sondern für Teilwörter einer festen Länge $b > 1$.
  - einziger Unterschied: an den Blättern des Huffman-Baumes stehen Wörter der Länge b.
- So etwas nennt man eine Block-Codierung.
  - Statt für einzelne $x \in A$ eine Codierung festzulegen,
  - legt man $f(x)$ für alle Blöcke $x \in A^b$ fest, und
  - erweitert dies zu einem Homomorphismus $f^{**} : (A^b)^* \longrightarrow B^*$ wie weiter vorne definiert

Block-Codierungen

- Verallgemeinerung des obigen Verfahrens:
  - Betrachte nicht Häufigkeiten einzelner Symbole,
  - sondern für Teilwörter einer festen Länge $b > 1$.
  - einziger Unterschied: an den Blättern des Huffman-Baumes stehen Wörter der Länge b.
- So etwas nennt man eine Block-Codierung.
  - Statt für einzelne $x \in A$ eine Codierung festzulegen,
  - legt man $f(x)$ für alle Blöcke $x \in A^b$ fest, und
  - erweitert dies zu einem Homomorphismus $f^{**} : (A^b)^* \longrightarrow B^*$ wie weiter vorne definiert

Beispiel

- $w = \texttt{abcdabcddcbaabcddcbadcbaabcddcba}$ und $b = 4$
  - Blockcodierung $h(\texttt{abcd}) = \texttt{0}$ und $h(\texttt{dcba}) = \texttt{1}$
  - damit $h(w) = \texttt{00101101}$
  - besser als z. B. $h'(\texttt{a}) = \texttt{00}$, $h'(\texttt{b}) = \texttt{01}$, $h'(\texttt{c}) = \texttt{10}$, $h'(\texttt{d}) = \texttt{11}$

Das ist wichtig

- Das sollten Sie mitnehmen:
  - Huffman-Codierung liefert kürzest mögliche präfixfreie Codes
  - „Algorithmus“ zur Bestimmung des Huffman-Baumes
- Das sollten Sie üben:
  - Huffman-Codes berechnen

Wo sind wir?

Von Wörtern zu Zahlen und zurück

Notationen für Funktionen

Von einem Alphabet zum anderen

Huffman-Codierung

Zahldarstellung auch für negative Zahlen

Repräsentation negativer Zahlen

Erinnerung

- $$\mathrm{Num}_b(\varepsilon) = 0$$
  $$\forall w \in Z_b^* \; \forall x \in Z_b : \mathrm{Num}_b(wx) = b \cdot \mathrm{Num}_b(w) + \mathrm{num}_b(x)$$

Repräsentation negativer Zahlen

Erinnerung

- $$\mathrm{Num}_b(\varepsilon) = 0$$
  $$\forall w \in Z_b^* \; \forall x \in Z_b : \mathrm{Num}_b(wx) = b \cdot \mathrm{Num}_b(w) + \mathrm{num}_b(x)$$

Ideen

- für negative Zahlen (und nichtnegative!)

Repräsentation negativer Zahlen

Erinnerung

- $$\mathrm{Num}_b(\varepsilon) = 0$$
  $$\forall w \in Z_b^* \; \forall x \in Z_b : \mathrm{Num}_b(wx) = b \cdot \mathrm{Num}_b(w) + \mathrm{num}_b(x)$$

Ideen

- für negative Zahlen (und nichtnegative!)
  - negative $\mathrm{num}_b(x)$: kleines Beispiel gleich
  - negatives b: hier nicht

Unübliche Methode für negative Zahlen — mit der Ziffer «meins»

- Ziffernmenge $Z = \{\bar{1}, 0, 1\}$
- definiere $\text{num} : Z \to \mathbb{Z}$
$$\begin{aligned} \bar{1} &\mapsto -1 \\ 0 &\mapsto 0 \\ 1 &\mapsto 1 \end{aligned}$$
- definiere $\text{Num} : Z^* \to \mathbb{Z}$ «wie üblich»:
$$\text{Num}(\varepsilon) = 0$$
$$\forall w \in Z^* \; \forall x \in Z : \text{Num}(wx) = 3 \cdot \text{Num}(w) + \text{num}(x)$$
- dann z. B.
  - $\text{Num}(1\bar{1}) = 1 \cdot 3^1 + (-1) \cdot 3^0 = 2$
  - $\text{Num}(\bar{1}01) = (-1) \cdot 3^2 + 0 \cdot 3^1 + 1 \cdot 3^0 = -8$
  - $\text{Num}(1\bar{1}01) = +3^3 - 3^2 + 0 + 3^0 = 19$

Unübliche Methode für negative Zahlen — Vorzeichenwechsel und Addition/Subtraktion sind ganz einfach

Karlsruher Institut für Technologie

- definere Homomorphismus $\mathrm{inv} : Z^* \to Z^*$ durch
  - $\mathrm{inv}(\mathtt{1}) = \bar{\mathtt{1}}$, $\mathrm{inv}(\bar{\mathtt{1}}) = \mathtt{1}$ und $\mathrm{inv}(\mathtt{0}) = \mathtt{0}$.

  also z. B. $\mathrm{inv}(\mathtt{1}\bar{\mathtt{1}}\mathtt{01}) = \bar{\mathtt{1}}\mathtt{10}\bar{\mathtt{1}}$

Behauptung
- für alle $w \in Z^*$ ist $\mathrm{Num}(\mathrm{inv}(w)) = -\mathrm{Num}(w)$.

Beobachtung
- die 4 Grundrechenarten funktionieren «wie in der Schule»

Unübliche Methode für negative Zahlen — Vorzeichenwechsel und Addition/Subtraktion sind ganz einfach

Karlsruher Institut für Technologie

- definere Homomorphismus $\mathrm{inv} : Z^* \to Z^*$ durch
  - $\mathrm{inv}(\mathtt{1}) = \bar{\mathtt{1}}$, $\mathrm{inv}(\bar{\mathtt{1}}) = \mathtt{1}$ und $\mathrm{inv}(\mathtt{0}) = \mathtt{0}$.

  also z. B. $\mathrm{inv}(\mathtt{1}\bar{\mathtt{1}}\mathtt{01}) = \bar{\mathtt{1}}\mathtt{10}\bar{\mathtt{1}}$

Behauptung
- für alle $w \in Z^*$ ist $\mathrm{Num}(\mathrm{inv}(w)) = -\mathrm{Num}(w)$.

Beobachtung
- die 4 Grundrechenarten funktionieren «wie in der Schule», z. B.:

$$\begin{array}{cccc} \mathtt{1} & \bar{\mathtt{1}} & \mathtt{0} & \mathtt{1} \\ & \bar{\mathtt{1}} & \mathtt{0} & \mathtt{1} \\ \hline \end{array}$$

Unübliche Methode für negative Zahlen — Vorzeichenwechsel und Addition/Subtraktion sind ganz einfach

- definere Homomorphismus inv : $Z^* \to Z^*$ durch
  - $\mathrm{inv}(1) = \bar{1}$, $\mathrm{inv}(\bar{1}) = 1$ und $\mathrm{inv}(0) = 0$.

  also z. B. $\mathrm{inv}(1\bar{1}01) = \bar{1}10\bar{1}$

Behauptung

- für alle $w \in Z^*$ ist $\mathrm{Num}(\mathrm{inv}(w)) = -\mathrm{Num}(w)$.

Beobachtung

- die 4 Grundrechenarten funktionieren «wie in der Schule», z. B.:

$$
\begin{array}{cccc}
1 & \bar{1} & 0 & 1 \\
 & \bar{1} & 0 & 1 \\
 & & 1 & \\
\hline
 & & & \bar{1}
\end{array}
$$

Unübliche Methode für negative Zahlen — Vorzeichenwechsel und Addition/Subtraktion sind ganz einfach

Karlsruher Institut für Technologie

- definere Homomorphismus $\mathrm{inv} : Z^* \to Z^*$ durch
  - $\mathrm{inv}(\mathtt{1}) = \mathtt{\bar{1}}$, $\mathrm{inv}(\mathtt{\bar{1}}) = \mathtt{1}$ und $\mathrm{inv}(\mathtt{0}) = \mathtt{0}$.

  also z. B. $\mathrm{inv}(\mathtt{1\bar{1}01}) = \mathtt{\bar{1}10\bar{1}}$

Behauptung

- für alle $w \in Z^*$ ist $\mathrm{Num}(\mathrm{inv}(w)) = -\mathrm{Num}(w)$.

Beobachtung

- die 4 Grundrechenarten funktionieren «wie in der Schule», z. B.:

$$
\begin{array}{cccc}
\mathtt{1} & \mathtt{\bar{1}} & \mathtt{0} & \mathtt{1} \\
 & \mathtt{\bar{1}} & \mathtt{0} & \mathtt{1} \\
 & \mathtt{0} & \mathtt{1} & \\
\hline
 & & \mathtt{1} & \mathtt{\bar{1}}
\end{array}
$$

Unübliche Methode für negative Zahlen — Vorzeichenwechsel und Addition/Subtraktion sind ganz einfach

- definere Homomorphismus $\text{inv} : Z^* \to Z^*$ durch
  - $\text{inv}(1) = \bar{1}$, $\text{inv}(\bar{1}) = 1$ und $\text{inv}(0) = 0$.

  also z. B. $\text{inv}(1\bar{1}01) = \bar{1}10\bar{1}$

Behauptung

- für alle $w \in Z^*$ ist $\text{Num}(\text{inv}(w)) = -\text{Num}(w)$.

Beobachtung

- die 4 Grundrechenarten funktionieren «wie in der Schule», z. B.:

$$
\begin{array}{cccc}
1 & \bar{1} & 0 & 1 \\
 & \bar{1} & 0 & 1 \\
\bar{1} & 0 & 1 & \\
\hline
 & 1 & 1 & \bar{1}
\end{array}
$$

Unübliche Methode für negative Zahlen — Vorzeichenwechsel und Addition/Subtraktion sind ganz einfach

- definere Homomorphismus $\mathrm{inv} : Z^* \to Z^*$ durch
  - $\mathrm{inv}(\mathtt{1}) = \overline{\mathtt{1}}$, $\mathrm{inv}(\overline{\mathtt{1}}) = \mathtt{1}$ und $\mathrm{inv}(\mathtt{0}) = \mathtt{0}$.

  also z. B. $\mathrm{inv}(\mathtt{1}\overline{\mathtt{1}}\mathtt{0}\mathtt{1}) = \overline{\mathtt{1}}\mathtt{1}\mathtt{0}\overline{\mathtt{1}}$

Behauptung

- für alle $w \in Z^*$ ist $\mathrm{Num}(\mathrm{inv}(w)) = -\mathrm{Num}(w)$.

Beobachtung

- die 4 Grundrechenarten funktionieren «wie in der Schule», z. B.:

$$\begin{array}{cccc}
\mathtt{1} & \overline{\mathtt{1}} & \mathtt{0} & \mathtt{1} \\
 & \overline{\mathtt{1}} & \mathtt{0} & \mathtt{1} \\
\overline{\mathtt{1}} & \mathtt{0} & \mathtt{1} & \\
\hline
\mathtt{0} & \mathtt{1} & \mathtt{1} & \overline{\mathtt{1}}
\end{array}$$

Unübliche Methode für negative Zahlen — Vorzeichenwechsel und Addition/Subtraktion sind ganz einfach

Karlsruher Institut für Technologie

- definere Homomorphismus $\mathrm{inv} : Z^* \to Z^*$ durch
  - $\mathrm{inv}(1) = \bar{1}$, $\mathrm{inv}(\bar{1}) = 1$ und $\mathrm{inv}(0) = 0$.

  also z. B. $\mathrm{inv}(1\bar{1}01) = \bar{1}10\bar{1}$

Behauptung

- für alle $w \in Z^*$ ist $\mathrm{Num}(\mathrm{inv}(w)) = -\mathrm{Num}(w)$.

Beobachtung

- die 4 Grundrechenarten funktionieren «wie in der Schule», z. B.:

$$\begin{array}{cccc|r}
1 & \bar{1} & 0 & 1 & 19 \\
 & \bar{1} & 0 & 1 & -8 \\
\bar{1} & 0 & 1 & & \\
\hline
0 & 1 & 1 & \bar{1} & 11
\end{array}$$

Rechnen in $\mathbb{Z}_k$ (1)

- Es sei $k \in \mathbb{N}_0$ mit $k \geq 2$.

Rechnen «modulo k»

- binäre Operationen auf $\mathbb{Z}_k$; für alle $x, y \in \mathbb{Z}_k$ sei

  $x +_k y = (x + y) \mod k$

  $x -_k y = (x - y) \mod k$

  - üblicherweise schreibt man $+$ statt $+_k$ (und $-$ statt $-_k$)

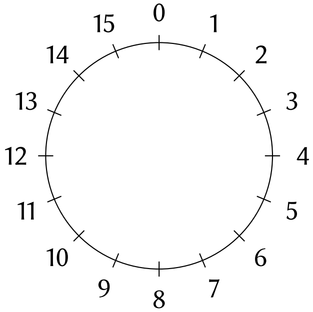
Lemma

- $\forall x, y \in \mathbb{N}_0$ ist

$$(x \pm y) \mod k = (x \mod k) \pm_k (y \mod k) .$$

Rechnen in $\mathbb{Z}_k$ (2) — graphisch: Zahlen als Pfeile

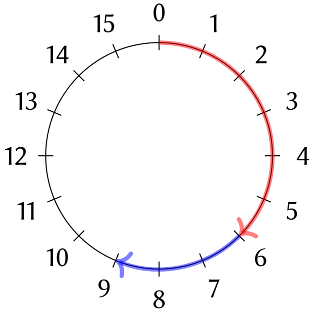
- kreisförmige Darstellung von $\mathbb{Z}_{16}$



- Addition $x +_k y$
  - Pfeile für x und y „hintereinander setzen“ (Anfang an Spitze)

Rechnen in $\mathbb{Z}_k$ (2) — graphisch: Zahlen als Pfeile

- kreisförmige Darstellung von $\mathbb{Z}_{16}$

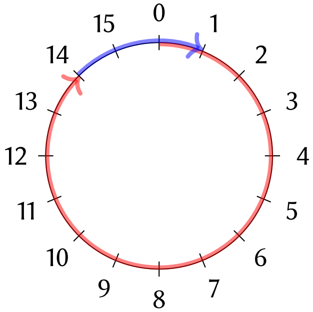


$6 +_{16} 3 = 9$

- Addition $x +_k y$
  - Pfeile für x und y „hintereinander setzen“ (Anfang an Spitze)

Rechnen in $\mathbb{Z}_k$ (2) — graphisch: Zahlen als Pfeile

- kreisförmige Darstellung von $\mathbb{Z}_{16}$

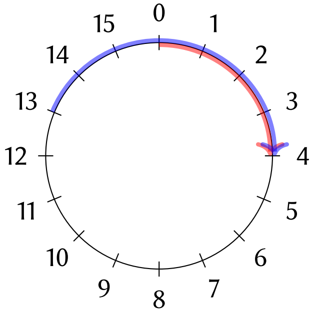


$14 +_{16} 3 = 1$

- Addition $x +_k y$
  - Pfeile für x und y „hintereinander setzen“ (Anfang an Spitze)

Rechnen in $\mathbb{Z}_k$ (2) — graphisch: Zahlen als Pfeile

- kreisförmige Darstellung von $\mathbb{Z}_{16}$



$4 -_{16} 7 = 13$

vgl. Minuten in Uhrzeiten,

mod 60

- Addition $x +_k y$
  - Pfeile für x und y „hintereinander setzen“ (Anfang an Spitze)
- Subtraktion $x -_k y$:
  - Pfeile für x und y „nebeneinander setzen“ (Spitze an Spitze)

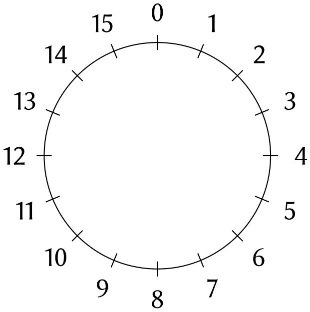
Zahldarstellungen mit beschränkter fester Länge

- in Rechnern üblich
  - MIMA benutzt 24 Bits (siehe Kapitel 10)
- Länge $\ell \in \mathbb{N}_+$, $\ell \geq 2$
- (zu) einfache Idee:
  - $\mathrm{bin}_\ell : \mathbb{Z}_{2^\ell} \to \{0, 1\}^\ell$

$$\mathrm{bin}_\ell(n) = 0^{\ell - |\mathrm{Repr}_2(n)|}\mathrm{Repr}_2(n)$$

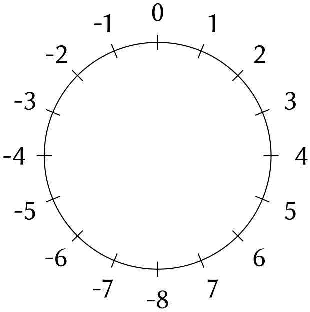
  - $|\mathrm{bin}_\ell(n)| = \ell$ und $\mathrm{Num}_2(\mathrm{bin}_\ell(n)) = n$
- völliges Fehlen negativer Zahlen unpraktisch

Negative Zahlen — Pfeile in die entgegengesetzte Richtung



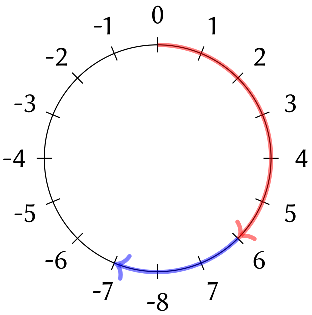
- bisher
  kreisförmige Darstellung von $\mathbb{Z}_{16}$

Negative Zahlen —
Pfeile in die entgegengesetzte Richtung



- nun
  kreisförmige Darstellung von $\mathbb{K}_4$ (Definition in Kürze)

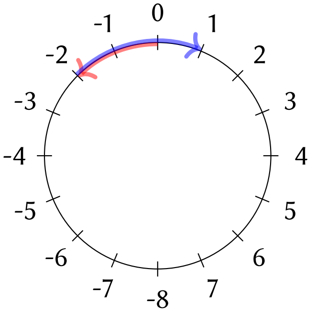
Negative Zahlen — Pfeile in die entgegengesetzte Richtung



- nun
  kreisförmige Darstellung von $\mathbb{K}_4$ (Definition in Kürze)
- Addition x „+“ y und Subtraktion x „−“ y
  analog zum Fall $\mathbb{Z}_k$

6 „+“ $3 = -7$

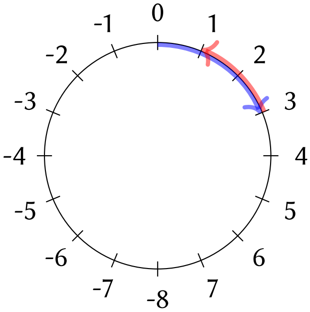
Negative Zahlen — Pfeile in die entgegengesetzte Richtung



- nun
  kreisförmige Darstellung von $\mathbb{K}_4$ (Definition in Kürze)
- Addition x „+“ y und Subtraktion x „−“ y
  analog zum Fall $\mathbb{Z}_k$

-2 „+“ $3 = 1$

Negative Zahlen — Pfeile in die entgegengesetzte Richtung

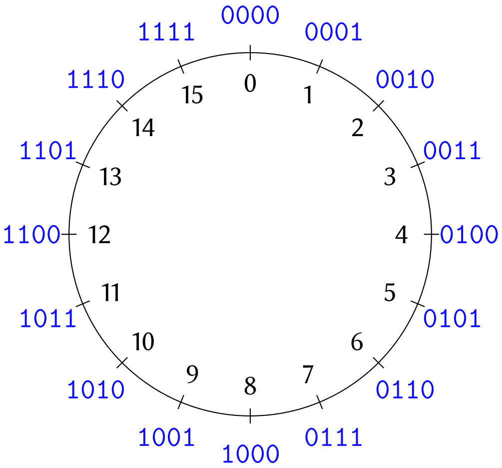


- nun
  kreisförmige Darstellung von $\mathbb{K}_4$ (Definition in Kürze)
- Addition x „+" y und Subtraktion x „−" y analog zum Fall $\mathbb{Z}_k$

3 „+" $-2 = 1$

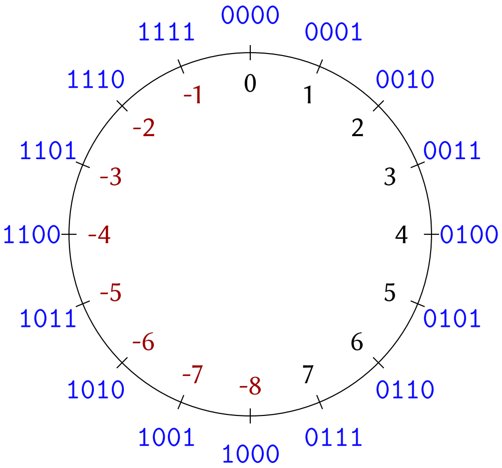
Darstellung auch negativer Zahlen

- $\mathbb{Z}_{16}$ in Binärdarstellung



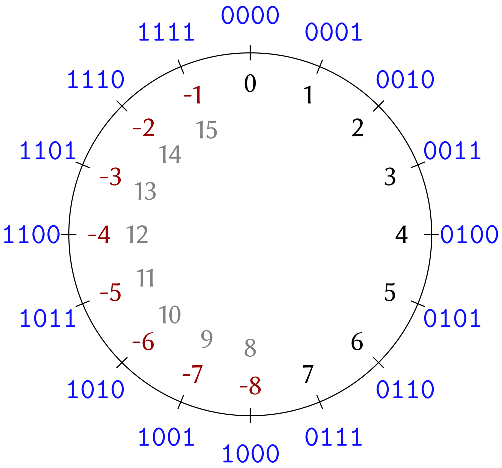
Darstellung auch negativer Zahlen

- $\mathbb{K}_4$ in Zweierkomplementdarstellung



Darstellung auch negativer Zahlen

- $\mathbb{K}_4$ in Zweierkomplement- und $\mathbb{Z}_{16}$ in Binärdarstellung



Zweierkomplement-Darstellung mit ℓ Bits — für negative und nichtnegative Zahlen

es sei $\ell \in \mathbb{N}_+$

- für die Zahlen in

  $\mathbb{K}_\ell = \{x \in \mathbb{Z} \mid -2^{\ell-1} \leq x \leq 2^{\ell-1} - 1\}$

  z. B.

  $\mathbb{K}_2 = \{-2, -1, 0, 1\}$

  $\mathbb{K}_8 = \{-128, -127, \ldots, -1, 0, 1, \ldots, 127\}$

- $\mathrm{Zkpl}_\ell \colon \mathbb{K}_\ell \to \{0, 1\}^\ell$

$$\mathrm{Zkpl}_\ell(x) = \begin{cases} \mathrm{bin}_\ell(x) & \text{falls } x \geq 0 \\ \mathrm{bin}_\ell(2^\ell + x) & \text{falls } x < 0 \end{cases}$$

äquivalent

$$\mathrm{Zkpl}_\ell(x) = \begin{cases} 0\mathrm{bin}_{\ell-1}(x) & \text{falls } x \geq 0 \\ 1\mathrm{bin}_{\ell-1}(2^{\ell-1} + x) & \text{falls } x < 0 \end{cases}$$

Zweierkomplementdarstellung — Rechnen (fast) wie in der Schule

0	1	1	1	7
1	0	1	1	−5
				2

Zweierkomplementdarstellung — Rechnen (fast) wie in der Schule

0	1	1	1	7
1	0	1	1	-5
		1		
			0	2

Zweierkomplementdarstellung — Rechnen (fast) wie in der Schule

0	1	1	1		7
1	0	1	1		-5
	1	1			
		1	0		2

Zweierkomplementdarstellung — Rechnen (fast) wie in der Schule

0	1	1	1	7
1	0	1	1	−5
1	1	1		
	0	1	0	2

Zweierkomplementdarstellung — Rechnen (fast) wie in der Schule

0	1	1	1	7
1	0	1	1	-5
1	1	1		
0	0	1	0	2

- Übertrag an der linkesten Stelle wird ignoriert
  … *das* ist anders als in der Schule

- Dass immer alles funktioniert, muss und kann man beweisen …

Das ist wichtig

- Das sollten Sie mitnehmen:
  - Zweierkomplement-Darstellung
- Das sollten Sie üben:
  - selbst Zahlen verschieden repräsentieren und
  - damit operieren
  - Definitionen auch in Randfällen ausprobieren