Teil I
Die ganzen Zahlen hat der liebe Gott gemacht

[1] Leopold Kronecker sagte in einem Vortrag vor der Berliner Naturforscher-Versammlung im Jahre 1886 „Die ganzen Zahlen hat der liebe Gott gemacht, alles andere ist Menschenwerk.“ (zitiert nach [38, S. 19]). Siehe auch Kroneckers Arbeit *Über den Zahlbegriff.* [24].

Kapitel 1
Zahlen und Ziffern

Den Umgang mit natürlichen Zahlen haben wir in unseren frühsten Kindertagen gelernt. Kieselsteine, Murmeln oder Jahre („Wie alt bist Du denn, mein Kleiner?“) dienten dabei als Anschauungsmaterial. Warum sollten wir uns noch einmal mit diesem Thema beschäftigen? Weil wir Computer für uns rechnen lassen wollen!

Im Gegensatz zu Menschen verfügen Computer über keinerlei Anschauung. Wenn wir sie für unsere Zwecke einspannen wollen, müssen wir also unsere Vorstellungen von Zahlen in einer Sprache formulieren, die völlig ohne Anschauung auskommt. Das ist die Sprache der Mathematik.[1] In diesem Sinne sind Zahlen durch Axiome präzise charakterisierte, aber völlig abstrakte mathematische Objekte. Wir können gewisse Rechenregeln herleiten und anwenden, aber nichts ausrechnen, nicht einmal $1+1$. Denn dazu benötigen wir einen eigenen Namen für jede einzelne Zahl, zum Beispiel 1 oder 2. Auf den ersten Blick wirkt diese Trennung zwischen Zahlen und Zahlendarstellung etwas künstlich. Tatsächlich beschert sie uns aber eine Vielzahl an neuen Möglichkeiten. So erlaubt die Dualdarstellung natürlicher Zahlen einfache technische Realisierungen im Computer, und die Speicherung negativer ganzer Zahlen im Zweierkomplement macht das Subtrahieren überflüssig.

1.1 Natürliche Zahlen

1.1.1 Rechnen und Ausrechnen

Die Menge $\mathbb{N}$ der natürlichen Zahlen ist folgendermaßen charakterisiert [11, Kap. 1]: Es gibt ein ausgezeichnetes Element $0 \in \mathbb{N}$ und eine Abbildung $S : \mathbb{N} \to \mathbb{N}$ mit den Eigenschaften:

(S1) S ist injektiv.
(S2) $0 \notin S(\mathbb{N})$.
(S3) Ist $M \subset \mathbb{N}$ und sowohl $0 \in M$ als auch $S(M) \subset M$ so gilt $M = \mathbb{N}$.

Die Funktion S (engl. *successor*) ordnet jeder natürlichen Zahl $n \in \mathbb{N}$ ihren Nachfolger $S(n)$ zu. Axiom (S1) besagt, daß dabei jede natürliche Zahl höchstens einmal vorkommt. Aus (S2) folgt, daß dieser Zählprozess bei 0 beginnt.[2] Axiom (S3) begründet die vollständige Induktion. Wir wollen die Addition zweier natürlicher Zahlen definieren. Dazu sei $n \in \mathbb{N}$ beliebig, aber fest gewählt. Wir setzen $n+0 = n$ und fordern außerdem, daß die Rekursionsformel $n+S(m) = S(n+m)$ gelten soll. Damit sind wir fertig. Bezeichnet nämlich $M \subset \mathbb{N}$ die Menge aller $m \in \mathbb{N}$, für die $n+m \in \mathbb{N}$ nun wohldefiniert ist, so gilt $0 \in M$, wie vereinbart, und $S(M) \subset M$ aufgrund der Rekursionsformel. Das bedeutet aber $M = \mathbb{N}$ wegen (S3). Setzt man übrigens $1 = S(0)$, so ergibt sich die Rechenregel $n+1 = S(n)$. Das Produkt $n \cdot m$ sowie Kommutativität,

[1] Bertrand Russel (1872-1970), Mathematiker und Nobelpreisträger für Literatur, schreibt in seinem Essay *Mathematics and the Metaphysicians* [32, S. 77]: “Obviousness is always the enemy of correctness. Hence, we invent some new and difficult symbolism, in which nothing seems obvious. Then we set up certain rules for operating on the symbols, and the whole thing becomes mechanical.”

[2] Die hier beschriebene Charakterisierung von $\mathbb{N}$ geht auf die Axiome von Peano (1858 - 1932) zurück. Andere Definitionen, beispielsweise von Cantor (1845 - 1918) oder Dedekind (1831 - 1916), ziehen den Beginn mit 1 vor und betrachten somit die 0 nicht als natürliche Zahl.

Assoziativität und Distributivität erhält man auf ähnliche Weise [25, Kap. 1, § 4]. Nun können wir also mit natürlichen Zahlen rechnen.

Allerdings gibt es eine wesentlichen Einschränkung: *Symbolisch* können wir die Summe $s = n + m$ zweier Zahlen $n, m \in \mathbb{N}$ zwar bilden, *numerisch* ausrechnen können wir sie aber noch nicht. Dazu müsste erst jede natürliche Zahl ihren eigenen, unverwechselbaren Namen haben. Ausrechnen von $s = n + m \in \mathbb{N}$ bedeutet dann, zu den gegebenen Namen der Summanden n, m den gesuchten Namen der Summe s zu finden. Damit man sowohl Aufgabe als auch Ergebnis aufschreiben und anderen mitteilen kann, dürfen solche Namen nur aus endlich vielen Zeichen oder Symbolen bestehen. In diesem Sinne bilden Ziffernsysteme die Grundlage des numerischen Rechnens.

1.1.2 Ziffernsysteme.

Sei $\mathscr{Z}$ eine endliche Menge von Ziffern. Daraus bilden wir endliche Ketten

$$z_1\, z_2\, \cdots\, z_k, \qquad z_i \in \mathscr{Z},\ i = 1, \ldots, k,$$

wie Worte aus den Buchstaben eines Alphabets. Wir wollen jeder Zahl $n \in \mathbb{N}$ einen Namen aus der Menge

$$\mathscr{D}(\mathscr{Z}) = \{z_1\, z_2\, \cdots\, z_k \mid k \in \mathbb{N}, z_i \in \mathscr{Z},\ i = 1, \ldots, k\}, \tag{1.1}$$

aller Ziffernketten zuordnen. Dem entspricht eine Abbildung $\varphi : \mathbb{N} \to \mathscr{D}(\mathscr{Z})$. Eine solche Abbildung φ heisst *injektiv*, falls $n = m$ aus $\varphi(n) = \varphi(m)$ folgt, *surjektiv*, falls es zu jedem $z \in \mathscr{D}(\mathscr{Z})$ ein $n \in \mathbb{N}$ mit $\varphi(n) = z$ gibt und *bijektiv*, falls sie injektiv und surjektiv ist. Wir zeigen nun, daß die Menge $\mathscr{D}(\mathscr{Z})$ zur Darstellung der natürlichen Zahlen geeignet ist.

Satz 1.1. *Es existiert eine bijektive Abbildung*

$$\varphi : \mathbb{N} \to \mathscr{D}(\mathscr{Z}). \tag{1.2}$$

Beweis. Es reicht zu zeigen, daß sich alle Elemente von $\mathscr{D}(\mathscr{Z})$ nacheinander anordnen lassen. Gleichbedeutend damit ist die Existenz einer Nachfolger-Abbildung $\sigma : \mathscr{D}(\mathscr{Z}) \to \mathscr{D}(\mathscr{Z})$ mit den gleichen Eigenschaften wie S aus den Axiomen (S1)–(S3). Ausgehend von dem Anfangselement $d_0 \in \mathscr{D}(\mathscr{Z})$ können wir dann φ einfach wie folgt definieren

$$\varphi(0) = d_0, \qquad \varphi(S(n)) = \sigma(\varphi(n))\,.$$

Zur Anordnung von $\mathscr{D}(\mathscr{Z})$ bemerken wir zunächst, daß

$$\mathscr{D}(\mathscr{Z}) = \bigcup_{k=1}^{\infty} \mathscr{D}_k, \qquad \mathscr{D}_k = \{z_1\, z_2\, \cdots\, z_k \mid z_i \in \mathscr{Z},\ i = 1, \ldots, k\}\,.$$

Die Reihenfolge der Mengen $\mathscr{D}_k$ untereinander folgt den natürlichen Zahlen, und die endlich vielen Elemente innerhalb jeder einzelnen Menge $\mathscr{D}_k$ lassen sich natürlich ihrerseits in irgendeine Reihenfolge bringen. □

Es lohnt sich, der Eigenschaft (1.2) einen Namen zu geben.

Definition 1.2 (Abzählbarkeit). Eine Menge M mit unendlich vielen Elementen heißt *abzählbar unendlich* oder kurz abzählbar, wenn eine bijektive Abbildung $\varphi : \mathbb{N} \to M$ existiert.

Satz 1.1 besagt also:

Die Menge aller endlicher Ziffernketten $\mathscr{D}(\mathscr{Z})$ aus einer endlichen Ziffernmenge $\mathscr{Z}$ ist abzählbar.

Wir wollen, daß jede Zahl $n \in \mathbb{N}$ genau einen Namen hat, aber es ist nicht nötig, daß alle Ziffernketten $z \in \mathscr{D}(\mathscr{Z})$ dabei verwendet werden (siehe Abschnitt 1.1.3).

Definition 1.3 (Ziffernsystem). Eine endlichen Ziffernmenge $\mathscr{Z}$ und eine injektive Abbildung $\varphi : \mathbb{N} \to \mathscr{D}(\mathscr{Z})$ bilden ein *Ziffernsystem* zur Darstellung von $\mathbb{N}$.

Umgangssprachlich wird zwischen verschiedenen Ziffernsystemen und den natürlichen Zahlen selbst nicht unterschieden. So spricht man von Dezimalzahlen, Dualzahlen und so weiter.

Eine wegen ihrer Einfachheit beliebte Wahl ist

$$\mathscr{Z} = \{\,|\,\}, \quad \mathscr{D}(\mathscr{Z}) = \{\,|, ||, |||, \ldots\}, \qquad \varphi(1) = |, \quad \varphi(n+1) = \varphi(n)\,|\,. \tag{1.3}$$

Die 0 bildet dabei eine Ausnahme. Das resultierende *Unärsystem* findet seit Jahrtausenden unter anderem auf Knochen, Kerbhölzern oder Bierdeckeln Verwendung (siehe Abbildung 1.1). Mehr zu diesem Thema

Abb. 1.1 Frühe Ziffernsysteme: Ishango-Knochen und Bierdeckel

erfährt man bei Ifrah [20, Kapitel 4]. Allerdings erweist sich solch eine Darstellung im Falle großer Zahlen schnell als unübersichtlich (vgl. erste Anklänge eines Quinärsystems in Abb. 1.1 rechts).

1.1.3 Positionssysteme.

Sei $q \in \mathbb{N}$, $q > 1$, fest gewählt. Dann erhält man zu jeder natürlichen Zahl n durch sukzessive Division mit Rest die Zerlegung

$$n = \sum_{i=0}^{k} r_i q^i. \tag{1.4}$$

mit $r_k \neq 0$ und eindeutig bestimmten Koeffizienten $r_i \in \{0, \ldots, q-1\} \subset \mathbb{N}$. Die Grundidee eines Positionssystems ist es nun, jede Zahl $r_i \in \{0, \ldots, q-1\}$ mit einer Ziffer zu bezeichnen und daraus den Namen von n zusammenzusetzen.

Definition 1.4 (Positionssystem zur Basis q). Ein *Positionssystem zur Basis q* ist ein Ziffernsystem, welches aus einer Ziffernmenge $\mathscr{Z}$ mit q Elementen und der Abbildung

$$\varphi(n) = \varphi\left(\sum_{i=0}^{k} r_i q^i\right) = \varphi(r_k)\varphi(r_{k-1})\cdots\varphi(r_0) \tag{1.5}$$

basierend auf der Zerlegung (1.4) und einer bijektiven Zuordnung $\varphi : \{0, 1, \ldots, q-1\} \to \mathscr{Z}$ gebildet ist. Die so erzeugte Zifferndarstellung (1.5) heißt *q-adische Darstellung*.

Die Eindeutigkeit der Koeffizienten r_i in der Zerlegung (1.4) liefert dabei die Injektivität von φ. Übrigens gilt $\varphi(\mathbb{N}) \neq \mathscr{D}(\mathscr{Z})$, denn wegen der Bedingung $r_k \neq 0$ in (1.4) kommt beispielsweise die Kette $\varphi(0)\varphi(1) \in \mathscr{D}(\mathscr{Z})$ nicht vor.

Wir sind seit Kindesbeinen an das Dezimalsystem, also an das Positionssystem zur Basis $q = 10$ gewöhnt. Dabei werden die ersten zehn natürlichen Zahlen $\{0, S(0), S\circ S(0), \ldots, S\circ\cdots\circ S(0)\}$ bijektiv auf die Ziffern $\mathscr{Z} = \{0,1,2,\ldots,9\}$ abgebildet. Dann erhält man zum Beispiel

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

Wir identifizieren von nun an die natürlichen Zahlen mit ihrer Darstellung im Dezimalsystem.

Auch bei anderen q-adischen Darstellungen unterscheiden wir nicht mehr zwischen den Ziffern $z_i \in \mathscr{Z}$ und den natürlichen Zahlen $i \in \{0,1,\ldots,q-1\}$. Wir schreiben also einfach

$$z_k\, z_{k-1} \ldots z_{0_q} = \sum_{i=0}^{k} z_i q^i, \qquad z_i \in \mathscr{Z} = \{0,1,2,\ldots,q-1\}\,.$$

Falls keine Missverständnisse auftreten können, verzichten wir auf den Index q.

Im Falle $q > 10$ müssten wir uns allerdings noch weitere Ziffern ausdenken. Oft werden zunächst die lateinischen Großbuchstaben verwendet, also $A, B, \ldots, Z$, für 10, 11, 12, …, 35. Unsere heutige Dezimalnotation entwickelte sich in Indien in der Zeit von 600 - 300 v. Chr. und fand über arabische Astronomen den Weg nach Europa. Das erste Positionssystem tauchte aber schon zu Beginn des zweiten Jahrtausends v. Chr. in Mesopotamien auf. Die babylonischen Mathematiker verwendeten ein Sexagesimalsystem, also die Basis $q = 60$. Im Sexagesimalsystem bedeutet

$$123_{60} = 1\cdot 60^2 + 2\cdot 60^1 + 3\cdot 60^0\,,$$

und man rechnet schnell nach, daß dieselbe Zahl im Dezimalsystem den Namen 3723 trägt. Interessanterweise machten die Babylonier bei der Entwicklung der benötigten 60 Ziffern vom Dezimalsystem Gebrauch. Die Mayas und Azteken verwendeten ein Vigesimalsystem, also die Basis $q = 20$. Das dekadische Ziffernsystem der Griechen ist kein Positionssystem. Eine Fülle weiterer Informationen zu Ursprung und Geschichte der Ziffern- und Positionssysteme findet sich bei Ebbinghaus et al. [11, Kap. 1] und Ifrah [20].

1.1.4 Dualdarstellung

Moderne Rechenmaschinen arbeiten sämtlich mit dem Dualsystem, also dem Positionssystem zur Basis $q = 2$. Der Grund ist, daß der benötigte, binäre Ziffernvorrat $\mathscr{Z} = \{0,1\}$ mit minimalem Aufwand technisch umgesetzt werden kann, sei es mechanisch mit Hebeln[3] und Relais oder elektronisch mit Röhren oder Halbleitern. Aber schon G. W. Leibniz (1646 - 1716) schätzte das Dualsystem sehr, wenn auch aus ganz anderen Gründen.[4]

Die Darstellung der Dezimalzahl 10 im Dualsystem ist

$$1010_2 \;=\; 1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 0\cdot 2^0 = 10\,.$$

Im Dualsystem kann man mit maximal N Stellen alle Zahlen $n \in \mathbb{N}$ mit der Eigenschaft

$$0 \leq n \leq 2^N - 1$$

[3] Die von Konrad Zuse (1910 - 1995) zwischen 1936 und 1938 im Wohnzimmer seiner Eltern in der Methfesselstraße 10 in Berlin-Kreuzberg gebaute Z1 war der erste frei programmierbare Rechner der Welt. Er besaß ein Ein- Ausgabewerk, ein Rechenwerk, ein Speicherwerk und ein Programmwerk, das die Programme von gelochten Filmstreifen ablas. Leider hat er nie funktioniert. Das technisch unvermeidbare, minimale Spiel der einzelnen mechnischen Bauteile wurde durch deren Hintereinanderschaltung dermaßen verstärkt, daß am Ende alles verhakt war. Solch lawinenartigen Fehlerverstärkungen können auch beim numerischen Rechnen auftreten (siehe Kapitel 7).

[4] Laplace (1749-1827), ein berühmter französischer Mathematiker und Astronom, berichtet: „Leibniz sah in der dyadischen Arithmetik das Bild der Schöpfung. Er stellte sich vor, die Einheit stelle Gott dar und die Null das Nichts; das höchste Wesen habe alle Dinge aus dem Nichts erschaffen, ebenso wie die Einheit und die Null alle Zahlen seines Zahlensystems ausdrücken.“ Zitiert nach Courant und Robbins [7, Kap. § 1]

darstellen. Dies folgt direkt aus der Summenformel für die geometrische Reihe

$$\sum_{i=0}^{N-1} q^i = \frac{q^N - 1}{q - 1}$$

und Einsetzen von $q = 2$.

Im Vergleich mit dem Dezimalsystem ist das Rechnen im Dualsystem sehr einfach. Zur Addition hat man sich nur $1 + 1 = 10_2$ zu merken, und das kleine Einmaleins $1 \cdot 1 = 1$ macht seinem Namen alle Ehre. Die aus der Schule bekannten Methoden zur schriftlichen Addition und Multiplikation bleiben anwendbar:

$$\begin{array}{r} 1\,0\,1\,1 \\ +\ {}_1\,1_1\,0_1\,1 \\ \hline 1\,0\,0\,0\,0 \end{array} \qquad\qquad \begin{array}{r} 1\,0\,0\,1 \cdot\ 1\,1 \\ \hline 1\,0\,0\,1 \\ 1\,0\,0\,1 \\ \hline 1\,1\,0\,1\,1 \end{array}$$

Im Dezimalsystem hätten wir die Aufgaben $11 + 5 = 16$ und $9 \cdot 3 = 27$ allerdings im Kopf erledigt.

1.1.5 Praktische Realisierung

Die kleinste digitale Informationseinheit im Computer kann die Werte 0 oder 1 annehmen und heißt *Bit*. Den Datenspeicher eines Computers kann man sich also als eine sehr lange Aneinanderreihung von Bits vorstellen. Als Grundeinheit für deren Aufteilung verwendet man die aus 8 Bits bestehenden *Bytes*.[5] Mit einem Byte lassen sich $2^8 = 256$ verschiedene Zustände speichern. Mehrere Bytes werden zu einem *Wort* zusammengefasst. Die Anzahl der dabei verwendeten Bits heisst *Wortlänge*. Die Wortlänge ist maschinenabhängig. Sie bestimmt die Menge an Information, welche in jedem einzelnen Rechenschritt (Takt) der jeweiligen CPU (Central Processor Unit) verarbeitet werden kann.

In heutigen Computern werden natürliche Zahlen hauptsächlich zur Adressierung verwendet. Dazu werden die kleinsten adressierbaren Einheiten, etwa die einzelnen Bytes, einfach durchnumeriert, wie die Häuser in einer Straße. Obwohl die dazu verwendeten Dualzahlen im allgemeinen unterschiedlich viele Stellen haben, wird für jede Dualzahl ein Wort mit fester Wortlänge von N Stellen reserviert und jeweils mit Nullen aufgefüllt, um den zusätzlichen Aufwand für die Speicherverwaltung zu sparen. Die Wortlänge N bestimmt dann die Anzahl 2^N von verfügbaren Hausnummern und so die maximale Größe des direkt adressierbaren Speichers. 64-Bit-Rechner erlauben somit Speicherbereiche bis zu einer Größe von etwa 18 Exabyte. Das sind 18 Milliarden Gigabyte.

Beim Lösen von Rechenaufgaben mit dem Computer spielen spezielle Datentypen für natürlichen Zahlen eine untergeordnete Rolle. Zwar verfügen Programmiersprachen wie beispielsweise C mit `insigned short`, `insigned int` und `unsigned long` oder MATLAB mit `unit8`, `unit16` und `unit32` über spezielle Datentypen für natürliche Zahlen mit unterschiedlichen Bitlängen. Diese Darstellungen werden aber nicht von allen arithmetischen Operationen unterstützt, müssen also vor solchen Operationen in einen anderen Datentyp umgewandelt werden. Dem damit verbundenen Zeitaufwand steht im Vergleich zur `integer`-Darstellung nur ein minimaler Speicherplatzgewinn von einem Bit gegenüber (siehe Abschnitt 1.2.4). Abgesehen von Ausnahmefällen macht das die Verwendung spezieller Darstellungen für natürliche Zahlen zum Rechnen unattraktiv.

1.2 Ganze Zahlen

1.2.1 Konstruktion von $\mathbb{Z}$ durch Abschluß von $\mathbb{N}$ unter Subtraktion

Seien n, m natürliche Zahlen. Ist $n \leq m$, so hat die Gleichung

[5] Das Kunstwort *Byte* wurde von IBM geprägt, die bei ihrem 1964 vorgestellten Rechner „System/360“ die Standardisierung der Wortlänge auf 8 Bit einführten. Der Begriff Byte ist an das phonetisch gleiche englische Wort „bite“ (Biss) angelehnt, daß eine klare Steigerung des Wortes „bit“ (Bisschen) darstellt.

$$x+n=m \tag{1.6}$$

die Lösung $x=m-n\in\mathbb{N}$. Wir wollen $\mathbb{N}$ so erweitern, daß (1.6) auch für $n>m$ eine eindeutig bestimmte Lösung hat. Da die Aufgabe (1.6) und damit auch die Lösung x durch die naürlichen Zahlen n, m charakterisiert ist, könnte man als zusätzlich benötigte Zahl einfach das Paar $x=(m,n)\in\mathbb{N}^2$ nehmen. Das würde allerdings sofort auf einen Widerspruch führen. Offenbar hat nämlich die Aufgabe $x+n+d=m+d$ für jedes beliebige $d\in\mathbb{N}$ dieselbe Lösung x, aber die Paare (m,n) und $(m',n')=(m+d,n+d)$ sind verschieden. Eliminiert man d aus den Gleichungen $m'=m+d$ und $n'=n+d$, so erhält man $m'-m=n'-n$, oder gleichbedeutend

$$n+m'=n'+m. \tag{1.7}$$

Die Gleichung (1.7) hat den Charme, daß sie für alle natürlichen Zahlen wohldefiniert ist, egal ob $n>m$ oder $n<m$ vorliegt. Um Eindeutigkeit zu erzwingen, wollen wir einfach alle Paare (m,n) und (m',n') mit der Eigenschaft (1.7) identifizieren. Dazu erklären wir auf $\mathbb{N}^2$ die folgende Äquivalenzrelation (vgl. A.1)

$$(m,n)\cong(m',n') \quad\Longleftrightarrow\quad n+m'=n'+m$$

und definieren die ganzen Zahlen $\mathbb{Z}$ als die Menge der resultierenden Äquivalenzklassen $[m,n]$. Formal ist damit $x=[m,n]$ die Lösung von (1.6). Anschaulich entspricht jeder Äquivalenzklasse $[m,n]$ ein Pfeil der Länge $|m-n|$, der im Falle $n<m$ nach rechts und im Fall $n>m$ nach links zeigt (vgl. Vektoren im Euklidischen Raum).

Addition und Multiplikation nebst den üblichen Rechenregeln lassen sich von $\mathbb{N}$ auf $\mathbb{Z}$ übertragen [11, Kap. 1] . Als Repräsentant von $[m,n]$ wählt man $(m-n,0)$, falls $n\le m$ und $(0,n-m)$, falls $n>m$ ist. Statt $(n,0)$ schreibt man n, und $(0,n)$ wird mit $-n$ abgekürzt.

1.2.2 Zifferndarstellung

Gegeben sei ein Ziffernsystem zur Darstellung von $\mathbb{N}$, welches aus der Ziffernmenge $\mathscr{Z}$ und der injektiven Abbildung $\varphi:\mathbb{N}\to\mathscr{D}(\mathscr{Z})$ besteht. Erweitert man $\mathscr{Z}$ um die zusätzliche Ziffer $-$ und setzt man $\varphi(-n)=-\varphi(n)$ für alle positiven $n\in\mathbb{N}$, so erhält man ein Ziffernsystem für die Darstellung von $\mathbb{Z}$. Es gilt also folgender Satz.

Satz 1.5. *Jedes Ziffernsystem zur Darstellung von* $\mathbb{N}$ *induziert ein Ziffernsystem zur Darstellung von* $\mathbb{Z}$.

Verwenden wir für die natürlichen Zahlen das Dezimalsystem, so erhalten wir auf diese Weise für die ganzen Zahlen die gewohnte Darstellung

$$\ldots,\ -3,\ -2,\ -1,\ 0,\ 1,\ 2,\ 3,\ \ldots\ .$$

Nun ist offenbar das Vorzeichen eine digitale Größe: Positive Zahlen kann man durch ein vorangestelltes „+“ oder auch eine vorangestellte „0“ kennzeichnen, und „-“ kann durch „1“ ersetzt werden. Die resultierende *Dualdarstellung der ganzen Zahlen* ist dann

$$\ldots,\ 111,\ 110,\ 11,\ 0,\ 01,\ 010,\ 011,\ \ldots\ . \tag{1.8}$$

Die erste Stelle wird jeweils als Vorzeichen interpretiert. Man spricht von einem *Vorzeichenbit*.

1.2.3 Speicherung im N-Bit-Zweierkomplement

Wir gehen davon aus, daß für die Speicherung ganzer Zahlen nur eine feste Anzahl von N Bits zur Verfügung steht, von denen das erste die Rolle des Vorzeichenbits übernimmt. Natürlich können wir dann auch nur endlich viele Zahlen darstellen. Wie bisher werden natürliche, also nicht-negative ganze Zahlen im Dualsystem dargestellt (vgl. Abschnitt 1.1.4). Die größte darstellbare ganze Zahl ist dann

$$z_{\max}=2^{N-1}-1,$$

denn das Vorzeichenbit steht ja als Dualstelle nicht zur Verfügung.

Zur Darstellung negativer ganzer Zahlen werden nun nicht, wie bisher, die Koeffizienten der Zerlegung in Zweierpotenzen verwendet, sondern deren Komplemente, also 0 statt 1 und 1 statt 0. Am Ende wird noch eine 1 addiert. Die Darstellung einer negativen ganzen Zahl im *N-Bit-Zweierkomplement* sieht also folgendermaßen aus:

$$1|z_{N-2}z_{N-3}\cdots z_0 = -\left(1+\sum_{i=0}^{N-2}(1-z_i)\,2^i\right).$$

Die kleinste auf diese Weise darstellbare Zahl ist offenbar

$$1|00\cdots 0 = -\left(1+\sum_{i=0}^{N-2}(1-0)\,2^i\right) = -2^{N-1}.$$

Zur Berechnung der Darstellung einer negativen Dezimalzahl im N-Bit-Zweierkomplement gibt es ein einfaches Kochrezept: a) Dualdarstellung, b) Umklappen der Bits und c) Addition von 1. Beispielweise erhält man im Falle $N=4$ für die negative Dezimalzahl -3 die Darstellung $1|101$ im Zweierkomplement wie folgt:

$$\underbrace{-3}_{\text{Dezimalzahl}} \;\rightarrow\; \underbrace{-0011}_{\text{Dualdarstellung}} \;\rightarrow\; \underbrace{1100}_{\text{Umklappen}} \;\rightarrow\; \underbrace{1|101}_{\text{Addition von 1}}\;.$$

Die Darstellung im Zweierkomplement wirkt auf den ersten Blick unnötig kompliziert, hat aber zwei wesentliche Vorteile.

Erstens ist die Darstellung der 0 durch $0|000$ eindeutig (der Einfachheit halber für die Bitlänge $N=4$). Während bei naiver Verwendung des Vorzeichenbits nämlich die 0 sowohl durch $0|000$ als auch durch $1|000$ dargestellt wird, liefert das 4-Bit-Zweierkomplement als Darstellung von 0 und von -0 jeweils $0|000$, denn

$$\underbrace{-0}_{\text{Dezimalzahl}} \;\rightarrow\; \underbrace{-0000}_{\text{Dualdarstellung}} \;\rightarrow\; \underbrace{1111}_{\text{Umklappen}} \;\rightarrow\; \underbrace{0|000}_{\text{Addition von 1}}\;.$$

Hier wird deutlich, wie die Speicherung im N-Bit-Zweierkomplement die endliche Anzahl verfügbarer Bits ausnutzt: Der Übertrag nach Addition von 1 kann einfach ignoriert werden.

Zweitens kann man die Subtraktion positiver Zahlen als Addition negativer Zahlen realisieren. Das folgende Beispiel zeigt, was damit gemeint ist. Die Lösung der Aufgabe $3-4=-1$ berechnet man im 4-Bit-Zweierkomplement durch Addition von $3=0|011$ und $-4=1|100$. So erhält man

$$\begin{array}{r} 0|\ 0\ 1\ 1 \\ +\,1|\ 1\ 0\ 0 \\ \hline 1|\ 1\ 1\ 1 \end{array}$$

also mit $1|111 = -(1+\sum_{i=0}^{2} 0\cdot 2^i) = -1$ das richtige Ergebnis. Subtraktionen können also durch Umrechnung ins N-Bit-Zweierkomplement (Umklappen und Addition von 1) und Addition vorgenommen werden. Ein Subtraktionswerk ist nicht nötig.

1.2.4 Praktische Realisierung

Ganze Zahlen sind als grundlegender Datentyp `integer` in nahezu jeder Programmiersprache verfügbar. Dabei hat sich die Speiecherung im Zweierkomplement weitgehend durchgesetzt. Entsprechend der Wortlänge der jeweiligen CPU stehen für jede Integer-Zahl eine Speicherlänge von maximal $N=8$, 16, 32, 64 oder 128 Bits, beziehungsweise 1, 2, 4, 8 oder 16 Bytes, zur Verfügung. Handelsübliche 64-Bit-Rechner erlauben also einen maximalen Zahlenbereich von

$$z_{\min} = -2^{64-1} \approx -9.2\cdot 10^{18}, \qquad z_{\max} = 2^{64-1}-1 \approx 9.2\cdot 10^{18}\;.$$

Manchmal ist von vorneherein klar, daß man mit einem kleineren Zahlenvorrat auskommt. Um in diesem Fall Speicherplatz sparen zu können, bieten die meisten Programmiersprachen unterschiedlich lange Integer-Zahlen an, die beispielsweise in C mit `short int`, `int` oder `long int` bezeichnet werden.

Die jeweiligen Speicherlängen sind nicht standardisiert. In der Praxis findet man oft die folgenden Realisierungen.

Datentyp	**Speicherlänge** N	$z_{\min}$	$z_{\max}$
short int	16	-32.768	32.767
int	16	-32.768	32.767
long int	32	-2.147.483.648	2.147.483.647

Tabelle 1.1 Ganzzahlige Datentypen in C

JAVA bietet mit `short`, `int` und `long` Integer-Zahlen der Länge N=16, 32 und 64 Bit. Auch in MATLAB ist es möglich, ganze Zahlen der Länge von 8, 16 oder 32 Bit mittels `int8`, `int16` und `int32` zu verwenden. Ebenso wie die entsprechenden Datentypen für natürliche Zahlen werden sie aber nicht von allen MATLAB-Funktionen unterstützt.

1.3 Aufgaben

Aufgabe 1.1 *Führen Sie die Umrechnung der gegebene Darstellung von der einen in die andere jeweils im Index gegebene Basis durch:*

a) $72_{10} = x_3$, *b)* $5453_6 = x_2$, *c)* $654_7 = x_9$, *d)* $17HAI_{26} = x_{36}$.

Aufgabe 1.2 *Bilden die römischen Zahlen ein Ziffernsystem? Begründen Sie Ihre Ansicht.*

Aufgabe 1.3 *Gegeben sei die Darstellung*

$$a_{n-1}a_{n-2}\dots a_1 a_{0_r}$$

einer natürlichen Zahl zur Basis $r = q^k$ *mit Ziffern* $a_i \in \mathscr{Z}_r = \{0,1,\dots,r-1\}$ *und* $a_{n-1} \neq 0$*, wobei* $q,k,r \in \mathbb{N}\setminus\{0\}$ *gilt. Wie sieht die Darstellung*

$$b_{m-1}b_{m-2}\dots b_1 b_{0_q}$$

dieser Zahl zur Basis q *mit Ziffern* $b_i \in \mathscr{Z}_q = \{0,1,\dots,q-1\}$ *und* $b_{m-1} \neq 0$ *aus? Begründen Sie Ihre Aussage!*

Hinweis: Überlegen Sie sich, warum $m \leq nk$ *gilt.*

Aufgabe 1.4 *a) Z Geben Sie für* $N = 16$ *und* $N = 32$ *jeweils die größte und kleinste im N-Bit-Zweierkomplement darstellbare ganze Zahl an.*

b) Stellen Sie die Zahlen $i_1 = 893$ *und* $i_2 = -65535$ *im 32-Bit-Zweierkomplement dar.*

Aufgabe 1.5 *Ich bin nun 65 Jahre alt. Bezüglich welcher Basis muss ich mein Alter darstellen, damit ich wieder 50 bin? Kann es auf diese Weise meiner 9-jährigen Tochter gelingen, in einen erst ab 18 freigegebenen Film zu kommen? Oder meinem 17-jährigen Sohn?*

Aufgabe 1.6 *Gegeben sei eine Zahl* $A = a_1 ... a_{n16}$ *im Hexadezimalsystem. Zeigen Sie, daß die Dualdarstellung von A berechnet werden kann, indem man die Ziffern* $a_i \in \{0,\dots F\}$ *jeweils in eine vierstellige Dualzahl* d_i *umrechnet und diese dann inklusive möglicher Nullen aufsteigend für* $i = 1,\dots,n$ *von rechts nach links hintereinander schreibt.*

Aufgabe 1.7 *a) Welches falsche Ergebnis erhalten Sie bei Addition der Zahlen* 7 *und* 1 *im 4-Bit-Zweierkomplement und warum? Kann auch die Addition zweier negativer Zahlen oder einer negativen und einer positiven Zahl zu falschen Ergebnissen führen? Wie kann man solche Fehler erkennen?*

b) Schreiben Sie ein Programm zur Addition und Subtraktion von Dezimalzahlen durch Addition im N-Bit-Zweierkomplement. Das Programm soll entweder die richtige Lösung oder eine Fehlermeldung ausgeben.