Anhang A

Computerarithmetik (∗)

A.1 Zahlendarstellung im Rechner und Computerarithmetik

Prinzipiell ist die Menge der im Computer darstellbaren Zahlen endlich. Wie „groß“ diese Menge ist, hängt von der Rechnerarchitektur ab. So sind bei einer 32Bit-Architektur zunächst maximal $2^{32} = 4294967296$ ganze Zahlen (ohne Vorzeichen) darstellbar. Innerhalb dieser Grenzen stellt die Addition und Multiplikation von ganzen Zahlen kein Problem dar. Ganz anders verhält es sich mit rationalen und erst recht irrationalen Zahlen: Jede Zahl $x \neq 0$ lässt sich bekanntlich darstellen als

$$x = \operatorname{sgn}(x) B^N \sum_{n=1}^{\infty} x_{-n} B^{-n}, \tag{A.1}$$

wobei $2 \leq B \in \mathbb{N}, N \in \mathbb{Z}$ und $x_n \in \{0, 1, \ldots, B-1\}$ für alle $n \in \mathbb{N}$ (B-adische Entwicklung von x). Diese ist eindeutig, wenn gilt:

- $x_{-1} \neq 0$, falls $x \neq 0$,
- es existiert ein $n \in \mathbb{N}$ mit $x_{-n} \neq B - 1$.

Eine Maschinenzahl dagegen hat die Form

$$\tilde{x} = \operatorname{sgn}(x) B^N \sum_{n=1}^{l} x_{-n} B^{-n},$$

wobei die feste Größe $l \in \mathbb{N}$ die Mantissenlänge ist. Als Mantisse bezeichnet man den Ausdruck

$$m(x) = \sum_{n=1}^{l} x_{-n} B^{-n}. \tag{A.2}$$

Hieraus folgt sofort, dass irrationale Zahlen überhaupt nicht und von den rationalen Zahlen nur ein Teil im Rechner dargestellt werden. Man unterscheidet zwei Arten der Darstellung:

1. *Festkommadarstellung.* Sowohl die Anzahl N der zur Darstellung verfügbaren Ziffern, als auch die Anzahl N_1 der Vorkommastellen ist fixiert. Die Anzahl der maximal möglichen Nachkommastellen N_2 erfüllt notwendigerweise

$$N = N_1 + N_2.$$

2. *Gleitkommadarstellung.* In (A.2) ist die Mantissenlänge l fixiert und der Exponent N ist begrenzt durch

$$N_- \leq N \leq N_+, \qquad N_-, N_+ \in \mathbb{Z}.$$

Alle Maschinenzahlen liegen dabei vom Betrag her im Intervall $(B^{N_- - 1}, B^{N_+})$. Zur Normalisierung der Darstellung verlangt man, dass $x_{-1} \neq 0$, wenn $x \neq 0$. Zahlen, die kleiner als $B^{N_- - 1}$ sind, werden vom Computer als 0 angesehen. Zahlen, die größer als B^{N_+} sind, können nicht verarbeitet werden (Exponentenüberlauf).

Die Darstellung von irrationalen Zahlen erfordert eine Projektion auf die Menge der Maschinenzahlen, die so genannte Rundung. Man unterscheidet vier Rundungsarten:

1. *Aufrundung.* Zu $x \in \mathbb{R}$ wählt man die nächsthöhere Maschinenzahl $\tilde{x}$.

2. *Abrundung.* Zu $x \in \mathbb{R}$ wählt man die nächstniedrigere Maschinenzahl $\tilde{x}$.

3. *Rundung (im engeren Sinne).* Ausgehend von der Darstellung (A.1), setzt man

$$\tilde{x} := \operatorname{sgn}(x) B^N \begin{cases} \sum_{n=1}^{l} x_{-n} B^{-n}, & \text{falls } x_{-(l+1)} < B/2, \\ \sum_{n=1}^{l} x_{-n} B^{-n} + B^{-l}, & \text{falls } x_{-(l+1)} \geq B/2. \end{cases}$$

4. *Abschneiden.* Verwerfen aller x_{-n} mit $n > l$ in der Darstellung A.1.

Die Größe

$$|\tilde{x} - x|$$

heißt absoluter, die Größe

$$\frac{|\tilde{x} - x|}{|x|}$$

heißt relativer Rundungsfehler.

Man wird erwarten, dass die Rundung im engeren Sinne die beste ist. In der Tat weist sie statistisch gesehen die geringsten Rundungsfehler auf. Für die Rundung (im engeren Sinne) gilt:

a) $\tilde{x}$ ist wieder eine Maschinenzahl.

b) Der absolute Fehler erfüllt

$$|\tilde{x} - x| \leq \frac{1}{2} B^{N-l}.$$

c) Der relative Fehler erfüllt

$$\frac{|\tilde{x} - x|}{|x|} \leq \frac{1}{2} B^{-l+1} =: eps.$$

Die Zahl *eps* wird auch als Maschinengenauigkeit bezeichnet.

A.2 IEEE Gleitkommazahlen

Die IEEE (Institute of Electrical and Electronic Engineers) ist eine internationale Organisation, die binäre Formate für Gleitkommazahlen festlegt. Diese Formate werden von den meisten (aber nicht allen) heutigen Computern benutzt. Die IEEE definiert zwei unterschiedliche Formate mit unterschiedlicher Präzision: Einzel- und Doppelpräzision (single and double precision). Single precision wird in C von float Variablen und double precision von double Variablen benutzt.

Intel's mathematischer Co-Prozessor benutzt eine dritte, höhere Präzision, die sogenannte erweiterte Präzision (extended precision). Alle Daten im Co-Prozessor werden mit dieser erweiterten Präzision behandelt. Werden die Daten aus dem Co-Prozessor im Speicher ausgelagert, so werden sie automatisch umgewandelt in single bzw. double precision. Die erweiterte Präzision benutzt ein etwas anderes Format als die IEEE float- bzw double-Formate und werden in diesem Abschnitt nicht diskutiert.

IEEE single precision.

Single precision Gleitkommazahlen benutzen 32 Bits (4 Byte) um eine Zahl zu verschlüsseln:
Bit 1-23: Hier wird die Mantisse gespeichert, d.h. die Mantissenlänge l beträgt bei single precision 23. Im single precision Format wird zur Mantisse noch Eins addiert.

Beispiel A.1 IEEE Mantisse.

$$\begin{aligned} &\overbrace{10110110100000000000000}^{\text{23 Bit}} \\ =&0.101101101_2 + 1 \\ =&1.101101101_2 \end{aligned}$$

□

Bit 24-31: Hier wird der binäre Exponent N gespeichert. In Wirklichkeit wird der Exponent N plus 127 gespeichert.
Bit 32: Gibt das Vorzeichen der Zahl an.

Beispiel A.2 IEEE single precsion. Welche Dezimalzahl wird duch den Binärcode $\underbrace{01000001}_{\text{Byte 3}}\ \underbrace{11011011}_{\text{Byte 2}}\ \underbrace{01000000}_{\text{Byte 1}}\ \underbrace{00000000}_{\text{Byte 0}}$ dargestellt?

Bit 32	Bit 31-24	Bit 23-1
$\underbrace{0}_{\text{Zahl ist positiv}}$	$\underbrace{10000011}_{N=131-127=4}$	$\underbrace{10110110100000000000000}_{\text{Mantisse}}$

$$\begin{aligned} &= 1.101101101_2 * 2^N \\ &= 11011.01101_2 \\ &= 2^4 + 2^3 + 2^1 + 2^0 + 2^{-2} + 2^{-3} + 2^{-5} \\ &= 27.406250 \end{aligned}$$

Folgendes Beispielprogramm testet ob der Computer die float Zahl 27.040625 im IEEE single precision Format speichert.

```
# include <stdio.h>
# define text1 "Der Computer testet ob die float Zahl 27.0406250 \n"
# define text2 "im IEEE single Format gespeichert wird.\n\n"
# define text3 "          \t   Byte 3  Byte 2  Byte 1  Byte 0  \n\n"
# define text4 "27.0406250 = \t 01000001 11011011 01000000 00000000"
# define text5 " (im IEEE Format)\n"
# define text6 "         = \t      65     219      64       0"
# define text7 " (Byte als unsigned char)\n\n\n"
# define text8 "Ihr Rechner betrachtet die Zahl als : %f \n"

int main()
{
       unsigned char *p;
       float a;

       printf(text1);
       printf(text2);
       printf(text3);
       printf(text4);
       printf(text5);
       printf(text6);
       printf(text7);

       /*            Byte 3        Byte 2        Byte 1        Byte 0  */
       /*                                                              */
       /* bin\"ar :   01000001      11011011      01000000      00000000 */
       /* dezimal :   65            219           64            0       */

       p=(unsigned char *) &a;
       p[0]= 0;
       p[1]= 64;
       p[2]= 219;
       p[3]= 65;

       printf(text8,a);
       return 0;
}
```

□

IEEE double precision

Double precision Gleitkommazahlen benutzen 64 Bits (8 Byte) um eine Zahl zu verschlüsseln:
Bit 1-52: Mantisse
Bit 53-63: Exponent
Bit 64: Vorzeichen
Mehr zu diesem Thema findet man zum Beispiel in [Car05].

A.3 Computerarithmetik

Das Rechnen mit ganzen Zahlen ist (innerhalb der erwähnten Grenzen) kein Problem: Addition und Multiplikation sind kommutativ, assoziativ und das Distributivgesetz gilt.

Beim Rechnen in der Fest- oder Gleitkommadarstellung gelten Assoziativ- und Distributivgesetz jedoch nicht!

Ein weiterer Effekt, der zu beachten ist, ist die Auslöschung von führenden Stellen.

Beispiel A.3 Auslöschnung. Sei $B = 10, l = 3$. Sei $x = 0.9995, y = 0.9984$ und es soll im Computer die Differenz $x - y$ berechnet werden. Wie wirkt sich die Rundung auf das Resultat aus? Zunächst ist $\tilde{x} = 0.1 \cdot 10^1$ und $\tilde{y} = 0.998$. Daher ist

$$\tilde{x} - \tilde{y} = 0.2 \cdot 10^{-2},$$

das exakte Resultat ist

$$x - y = 0.11 \cdot 10^{-2}$$

und der relative Fehler ergibt sich zu 81.8% ! Dieser Effekt der Verstärkung des Rundungsfehlers tritt auf, wenn man fast identische Zahlen voneinander subtrahiert. □

Eine Operation mit ähnlich instabilen Verhalten ist die Division x/y mit $x \ll y$. Addition zweier Zahlen mit gleichem Vorzeichen und Multiplikation sind jedoch gutartige Operationen.