Computergrundlagen
Boolesche Logik, Zahlensysteme und Arithmetik

Institut für Computerphysik
Universität Stuttgart

Wintersemester 2012/13

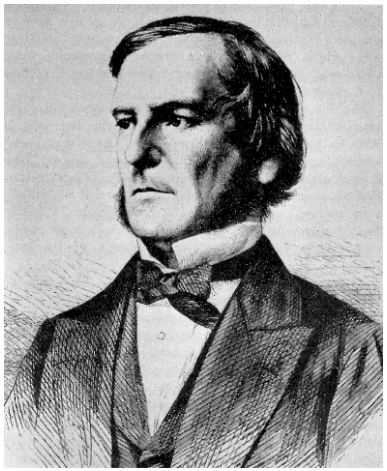
Wie rechnet ein Computer?

Ein Mikroprozessor

- ist ein Netz von Transistoren, Widerständen und Kondensatoren
- Leitungen kennen nur zwei Zustände: Spannung oder nicht
- Interpretation als ja/nein, 0/1, an/aus, richtig/falsch...
- Schaltungen entsprechen logischen Operationen

Die richtige Algebra (das richtige Zahlensystem) musst verwendet werden um damit rechnen zu können.

Aussagenlogik



G. Boole,
1815 - 1864

- erlaubt formale Beweise über logische Aussagen
- Grundlage der Computerlogik
- kann mathematisch als Algebra formalisiert werden (*boolesche Algebra*)
- Anwendung auch in Computerprogrammieren
- die Operatoren dienen zur Beschreibung der boolschen Funktion digitaler Schaltungen (hohe bzw. niedrige Spannung)

Die zweielementige boolesche Algebra

Wir betrachten eine Menge

- mit zwei Elementen 1 („wahr“) und 0 („falsch“)
- mit zwei Verknüpfungen $\vee$ („oder“) und $\wedge$ („und“)
- mit einer einstelligen Verknüpfung $\neg$ („nicht“, Negation)

Ferner gilt für beliebige $a, b, c \in \{0, 1\}$:

1. $a \vee (b \vee c) = (a \vee b) \vee c$,
   $a \wedge (b \wedge c) = (a \wedge b) \wedge c$ (Assoziativität)
2. $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$,
   $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$ (Distributivität)
3. $a \vee b = b \vee a$, $a \wedge b = b \wedge a$ (Kommutativität)
4. $a \vee (a \wedge b) = a$, $a \wedge (a \vee b) = a$ (Adsorption)
5. $a \vee \neg a = 1$, $a \wedge \neg a = 0$ (Komplemente)

Abgeleitete Gesetze

- Neutralität:
  $a \vee 0 = a, \quad a \wedge 1 = a$
- Idempotenz:
  $a \vee a = a, \quad a \wedge a = a$
- Extremalgesetze:
  $a \vee 1 = 1, \quad a \wedge 0 = 0$
- Doppelnegation: $\neg(\neg a) = a$
- Dualität:
  $\neg 0 = 1, \quad \neg 1 = 0$
- De Morgansche Gesetze:
  $\neg(a \vee b) = \neg a \wedge \neg b, \quad \neg(a \wedge b) = \neg a \vee \neg b$

Die zweielementige boolesche Algebra entspricht genau unserem Verständnis von Logik.

Was sind Zahlensysteme?

- Ein Zahlensystem wird zur Darstellung von Zahlen verwendet.
- Jede ganze Zahl B $\geq$ 2 kann als Basis(B) verwendet werden.
- Die Stelle bestimmt den Wert der Ziffer (die „niederwertigste“ Position steht dabei im Allgemeinen rechts, z.B. die erste 3 in 373 hat einen anderen Wert als die zweite 3 dreihundert nicht drei).
- Der Ziffernvorrat ist 0 bis B-1.
- Die Zifferposition bestimmt den Stellenwert B^n, entspricht eine Potenz der Basis.
- Zwei benachbarte Stellenwerte unterscheiden sich um den Faktor B.
- Die gängigsten Basen sind 2 (Dualsystem), 8 (Oktalsystem), 10 (Alltags Rechnen) oder 16 (das in der Informatik wichtige Hexadezimalsystem).

Was sind Zahlensysteme?

- Das System zur B=2 enthält nur 0 und 1.und wird für Computerrechnen und Datenspeichern verwendet.
- Die kleinste Informationseinheit (Bit) ist die Information über die Möglichkeiten 1 oder 0.
- In der Computertechnik wird oft das Hexadezimalsystem (Basis B=16) verwendet (0-9,A-F)
- Die Standardeinheit der Informationsgröße ist ein Byte (=8 Bit): die Information über eine aus 256 Möglichkeiten (die je zwei Zustände der acht Bits ermöglichen insgesamt 2^8=256 Möglichkeiten).
- In dezimaler Darstellung: 0, 1, 2, ... bis 255, im Binärsystem: 00000000, 00000001, 00000010, ... bis 11111111.)
- Mit dem 16er-System (immer genau 2 Ziffern entsprechen einem Byte) viel besser handhaben als mit dem Dezimalsystem.

Zahlensysteme: Eigenschaften

- Wie kann ich mit nur zwei Elementen Zahlen darstellen?

Sei $B > 0$ eine natürliche Zahl. Dann kann jede natürliche Zahl z *eindeutig* dargestellt werden als

$$z = \sum_{i=0}^{\infty} B^n z_k,$$

wobei für alle k $0 \leq z_k < B$ und nur endliche viele $z_k \neq 0$.

Beispiel

$B = 10$ entspricht unserem Dezimalsystem:

$$1042 = 10^0 \cdot 2 + 10^1 \cdot 4 + 10^3 \cdot 1 = 1042_{16}$$

$B = 8$ ergibt das Oktalsystem:

$$1042 = 8^0 \cdot 2 + 8^1 \cdot 2 + 8^3 \cdot 2 = 2022_8$$

Binärsystem

- Wie kann ich mit nur zwei Elementen Zahlen darstellen?

Wir benutzen das *Binärsystem* mit $B = 2$ und Ziffern 0 und 1 (Bits)

Beispiele

$$1042 = 2^{10} + 2^4 + 2^1 = 10.000.010.010_2$$

- Umrechnung von Binär- auf Dezimalzahlen ist umständlich
- Binär$\leftrightarrow$ oktal ist einfach:

$$10.000.010.010_2 = 2022_8$$

- *Hexadezimal* ($B = 16$, Ziffern 1–9, A–F) auch:

$$10.0000.0010.010_2 = 412_{16}$$

$$1010.1111.1111.1110_2 = \mathrm{AFFE}_{16}$$

Umwandeln von Binär in Hexadezimalzahlen und umgekehrt

- Jeweils 4 Binärstellen entsprechen einer Hexadezimalstelle ($16 = 2^4$).
- Umwandlung:
  - **Vom Binär- ins Hexadezimalsystem**: Unterteile die Binärzahl (rechts nach links) in 4er-Päckchen; wandle jedes Päckchen nach nebenstehender Tabelle in die entsprechende Hexadezimalziffer um.
  - **Vom Hexadezimal- ins Binärsystem**: Wandle die Hexadezimalziffern der Reihe nach in die entsprechenden vierstelligen Binärzahlen um.

Dezimal	Hexadezimal	Binär
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Quelle:mathe-lexikon.at

Umwandeln von Binär in Oktalzahlen

- Jeweils 3 Binärstellen entsprechen einer Oktalstelle ($8 = 2^3$).
- Die Ziffern in Dreierblöcke gruppieren (von rechts nach links),
- die entsprechenden oktalen Ziffern aus der Tabelle ablesen,
- falls Stellen über bleiben, die keine vollständige Dreier-Gruppe ergeben, kann man sich statt dessen 'unsichtbare Nullen' vorstellen (z.B. 10 ist mit 010 gleichbedeutend und umgekehrt die Dreiergruppe 001 ist gleichbedeutend mit 1).

Oktal	Binär
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Quelle:mathe-lexikon.at

Darstellung reeller Zahlen

$$\pm 1,23456789 \cdot 10^{\pm 123}$$

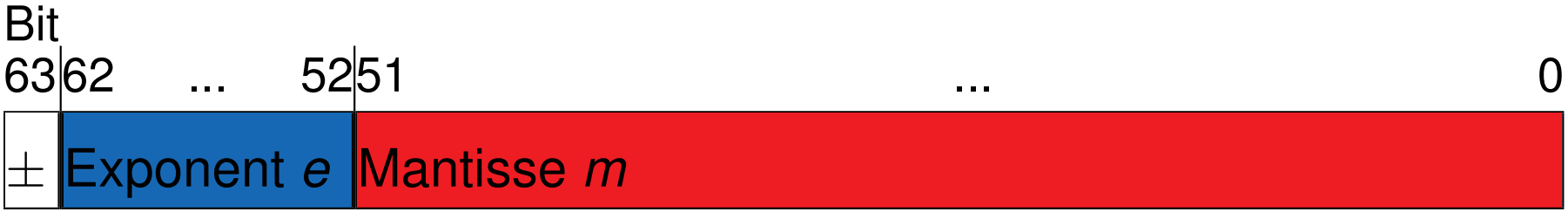
- Eine Fließkommazahl besteht aus:
  - Vorzeichen
  - Mantisse
  - (10er-)Exponent
- Stellen daher nur einen Teil der rationalen Zahlen exakt dar
- Alle anderen Zahlen werden angenähert
- Die Mantissenlänge bestimmt die Genauigkeit der Näherung

Binär entsprechend:

$$\pm 1,00111100000011001010 \cdot 2^{\pm 1111011}$$

IEEE-Fließkommazahlen

Speicherung einer 64-Bit-Fließkommazahl nach IEEE 754-Standard:



- Keine Komplementdarstellung von Mantisse oder Exponent
- Binärsystem: (normalisierte) Mantisse wird ohne führende Stelle gespeichert, die immer 1 ist
- Der 11-bittige Exponent e ist um (Bias=)1023 (127 in 32-Bit-Fließkommazahlen) verschoben gespeichert und nimmt Werte von -1022 ($e = 1$) bis 1023 ($e = 2046$) an
- Der Wert einer Zahl ist also:

$$\pm 1, m \cdot 2^{e-1023}$$

Spezielle Werte

Was ist mit den Exponenten -1023 ($e = 0$) und 1024 ($e = 2047$)?

Diese stellen spezielle Werte dar:

$e = 0$ und $m = 0$	± 0
$e = 0$ und $m \neq 0$	Zahlen der Form $\pm 0, m \cdot 2^{-1022}$
$e = 2047$ und $m = 0$	$\pm\infty$
$e = 2047$ und $m \neq 0$	$\pm$NaN (not a number)

- Da die erste Stelle immer 1 ist, kann 0 nur so dargestellt werden
- $\pm\infty$ ergibt sich z.B. bei Berechnung von $\pm 1/0$
- $\pm$NaN ergibt sich z.B. bei Berechnung von $\sqrt{-1}$
- Python fängt NaNs mit Fehlern ab, nicht aber z.B. C

32-Bit gegen 64-Bit Fließkommazahlen

- 32-Bit: einfach-genaues Format (single precision)
  - [Länge des Exponenten=8 Bit, Länge der Mantisse=23 Bit, Bias des Exponenten=127]
  - Genauigkeit der Darstellung von ca. 7 Dezimalstellen
  - z.B. bei einer Zahl wie 12.345,678 ist die dritte Stelle nach dem Komma schon ungenau
  - in numerischen Berechnungen mit mehreren Rechenoperationen sinkt die Genauigkeit sehr stark $\rightarrow$ einfach genaues Format nicht ausreichent
- 64-Bit: doppelt genaues Format (double precision)
  - [Länge des Exponenten=11 Bit, Länge der Mantisse=52 Bit, Bias des Exponenten=1023]
  - Genauigkeit der Darstellung von ca. 16 Dezimalstellen.

Speicherplatz und Rechenleistung sind so reichlich vorhanden, dass man nur noch das doppelt genaue Format verwendet kann.

Subtraktion/Addition von Fließkommazahlen

Beispiele:

$$1,0 \cdot 10^{0} + 1,0 \cdot 10^{-5}$$
$$= 1,0 \cdot 10^{0} + 0,00001 \cdot 10^{0} = 1,00001 \cdot 10^{0}$$

$$1,000002 \cdot 10^{0} - 1,000001 \cdot 10^{0}$$
$$= 0,000001 \cdot 10^{0} = 1,0 \cdot 10^{-6}$$

- Verschieben der Mantisse der kleineren Zahl, bis beide denselben Exponenten haben
- Dabei gehen Stellen der kleineren Zahl verloren
- Dann gewöhnliche Addition/Subtraktion der Mantissen
- Schließlich den Exponenten verringern, bis die Mantisse keine führenden Nullen hat

Unterlauf/Überlauf

- In der Fließkommadarstellung gibt es eine kleinste(größte) positive Zahl,
- unter(über) diese Zahl kann kein Wert dargestellt werden
- in diesem Bereich wird die Zahl als $0(\infty)$ repräsentiert
- im Falle eines Zwischenergebnisses ist die Information üeber das Ergebnis verloren

Zahlen verschiedener Größenordnung

- Die Addition/Subtraktion einer betragsmäßig viel kleineren Zahl ändert die größere Zahl nicht

  in einer vierstelligen Dezimalarithmetik:

$$\begin{aligned} 1\text{e}3 + 1\text{e}{-2} &= 1,000 \cdot 10^3 + 1,000 \cdot 10^{-2} \\ &= 1,000 \cdot 10^3 + 0,000|010... \cdot 10^2 \\ &= 1,000 \cdot 10^3 + 0,000 \cdot 10^3 = 1\text{e}3 \end{aligned}$$

Auslöschung: falsches Ergebnis bei der Subtraktion fast gleich großer Zahlen

Ist $x < 2^{-l_m}$, wobei l_m die Bitlänge der Mantisse ist, so ist in Fliesskommaarithmetik

$$x \to x + 1 - 1 \to (x + 1) - 1 \to 1 - 1 \to 0$$

- *Auslöschung* der vorderen Stellen bei Subtraktion ungefähr gleich großer Zahlen führt zu fehlenden hinteren Stellen im Ergebnis
- Im Beispiel verschwindet die kleinere Zahl x komplett
- Ist $x \approx 2^{-l_m}$, verliert x fast alle signifikanten Stellen
- Formeln muss man umformen so dass keine Auslöschung passieren kann.

Beispiel: quadratische Gleichung

- Lösen der quadratischen Gleichung $x^2 + ax + b = 0$:

$$x_\pm = -\frac{a}{2} \pm \sqrt{\left(\frac{a}{2}\right)^2 - b}$$

- Bei kleinem b ist $\sqrt{\left(\frac{a}{2}\right)^2 - b} \approx \frac{a}{2}$
- $a > 0$: *Auslöschung* bei der größeren Nullstelle x_+
- $a < 0$: *Auslöschung* bei der kleineren Nullstelle x_-
- Ausweg: Es gilt

$$x_+ x_- = \left(-\frac{a}{2} + \sqrt{\left(\frac{a}{2}\right)^2 - b}\right)\left(-\frac{a}{2} - \sqrt{\left(\frac{a}{2}\right)^2 - b}\right) = b$$

- Berechne also nur die stabile Summe (x_- bei $a > 0$, sonst x_+), und die zweite Nullstelle durch $x_+ = b/x_-$ bzw. $x_- = b/x_+$

Multiplikation von Fließkommazahlen

Beispiel:

$$1,01 \cdot 10^{0} \times 1,01 \cdot 10^{5}$$
$$= 1,0101 \cdot 10^{0+5} = 1,0101 \cdot 10^{5}$$

- Multiplikation und Division sind unkritisch, es kann zu keiner Auslöschung kommmen
- Allerdings zu Über- oder Unterläufen des Exponenten
- In diesem Fall ist das Ergebnis ∞ bzw. 0
- Meist kann man dies aber durch geeignete Umformungen auch vermeiden, z.B. durch logarithmisches Rechnen

Fehlerquellen bei numerischen Rechnungen

- **Numerische Fehler**
  - Auslöschung
  - Über- und Unterläufe
  - Rundungsfehler durch endliche Mantissenlänge
- **Algorithmische (Verfahrens-) Fehler**
  - Abschneidefehler: Unendliche Summen müssen durch endliche Summen ersetzt werden
  - Diskretisierungsfehler: Funktionsauswertung immer nur an endlich vielen Punkten
- **Modellierungsfehler**
  Das der Rechnung zugrundeliegende Modell erfasst wesentliche Details des Experiments nicht
- **Datenfehler**
  Genauigkeit der Anfangswerte (z.B. aus einem Experiment)