

# Symbolisches Rechnen Vorlesung Wintersemester 2006, 2014, 2026 Sommersemester 2021, 2025

Johannes Waldmann, HTWK Leipzig

19. Oktober 2026

## 1 Einleitung

### Symbolisches Rechnen: Beispiele: Zahlen

- numerisches Rechnen mit Maschinenzahlen

```
sqrt 2 + sqrt 3 ==> 3.1462643699419726  
(sqrt 2 + sqrt 3)*(sqrt 2 - sqrt 3) ==> ...
```

- exaktes Rechnen (mit algebraischen Ausdrücken)

$$(\sqrt{2} + \sqrt{3}) \cdot (\sqrt{2} - \sqrt{3}) = \dots,$$

```
maxima: expand(%)
```

### Symbolisches Rechnen: Beisp.: Funktionen

- auf konkreten Daten: `let f x = (x+1)^2 in f 3.1 - f 3`
- auf symbolischen Daten: `diff((x+1)^2, x)`
- `subst([x=3], diff((x+1)^2, x))`
- eigentlich `diff(\x -> (x+1)^2)`  
mit `diff :: (R -> R) -> (R -> R)`,

## Symbolisches Rechnen: Motivation

hat weitreichende Anwendungen:

- Lösen von (parametrisierten) Aufgabenklassen  
(für numerisches Rechnen muß Parameter fixiert werden)
- *exaktes* Lösen von Aufgaben  
(numer. R. mit Maschinenzahlen: nur Approximation)
- experimentelle, explorative, exakte Mathematik und Programmierung

ist nützlich im Studium, verwendet und vertieft:

- Mathematik (Analysis, Algebra)
- Algorithmen-Entwurf, -Analyse
- Prinzipien von Programmiersprachen

## Überblick

- Zahlen (große, genaue)
- Vektoren (Gitterbasen)
- Polynome
- Terme, Term-Ersetzungs-Systeme  
(Anwendung: Differentiation, Vereinfachung)
- Termination und Vervollständigung v. Reduktionssystemen
  - für Polynom-Ideale, Anw.: Geometrische Beweise  
für Gleichheits-Theorien auf Termen (Knuth-Bendix)
- maschinell unterstütztes interaktives Programmieren und Beweisen in konstruktiver Typ-Theorie (Agda)

## Literatur

- Wolfram Koepf: *Computeralgebra*, Springer, 2006. <https://www.mathematik.uni-kassel.de/~koepf/CA/>

- Hans-Gert Gräbe: *Einführung in das Symbolische Rechnen, Gröbnerbasen und Anwendungen*, Skripte, Universität Leipzig <https://www.informatik.uni-leipzig.de/~graebe/skripte/>
- Franz Baader and Tobias Nipkow: *Term Rewriting and All That*, Cambridge, 1998. <https://www21.in.tum.de/~nipkow/TRaAT/>
- weitere Literatur siehe z.B. <https://portal.risc.jku.at/Members/hemmecke/teaching/ppscs>

## Software

- wir benutzen
  - Maxima <https://maxima.sourceforge.net/>
  - FriCAS <https://github.com/fricas/fricas/>
  - Geonext <https://geonext.uni-bayreuth.de/>
  - GHC <http://www.haskell.org/ghc/>
  - Agda <https://wiki.portal.chalmers.se/agda/pmwiki.php>
- ist alles im Pool installiert (ssh, tmux, x2go)
- allgemeine Hinweise, auch zum Selbstbauen <https://www.imn.htwk-leipzig.de/~waldmann/etc/cas/>

## Beispiel: S.R. und Term-Ersetzung

Regeln für symbolisches Differenzieren (nach  $t$ ):

$$\begin{aligned} D(t) &\rightarrow 1 & D(\text{constant}) &\rightarrow 0 \\ D(+ (x, y)) &\rightarrow + (D(x), D(y)) \\ D(* (x, y)) &\rightarrow + (* (y, D(x)), * (x, D(y))) \\ D(-(x, y)) &\rightarrow - (D(x), D(y)) \end{aligned}$$

Robert Floyd 1967, zitiert in: Nachum Dershowitz: *33 Examples of Termination*, [https://doi.org/10.1007/3-540-59340-3\\_2](https://doi.org/10.1007/3-540-59340-3_2)

- Korrektheit? Termination? Komplexität?
- Strategie (Auswahl von Regel und Position)?
- ausreichend? angemessen?

### Beispiel: Termersetzung (cont.)

```
data E = Zero | One | T
       | Plus E E | Times E E deriving Show

e :: E
e = let b = Plus T One in Times b b

d :: E -> E
d e = case e of
  Zero -> Zero ; One -> Zero ; T -> One
  Plus x y -> Plus (d x) (d y)
  Times x y ->
    Plus (Times y (d x)) (Times x (d y))
```

### Beispiel: Inverse Symbolic Calculator

- <http://wayback.cecm.sfu.ca/projects/ISC/ISCmain.html> (kaputt)  
zur Bestimmung ganzzahliger Relationen (z.B. zwischen Potenzen einer numerisch gegebenen Zahl)

- `sqrt(2+sqrt 3) ==> 1.9318516525781366`

```
integer relations algorithm, run:
K = 1.9318516525781366
```

K satisfies the polynomial,  $X^4 - 4X^2 + 1$

mit LLL-Algorithmus (Lenstra, Lenstra, and Lovasz, 1982), der kurzen Vektor in geeignetem Gitter bestimmt.

### Hausaufgaben KW 15, Organisatorisches

1. zum Haskell-Programm zum Symb. Differenzieren:

- füge Syntax und Regel für Quotienten hinzu
- schlage Regeln zur Vereinfachung vor

2. ISC Simple Lookup and Browser sagt für  $\sqrt{2 + \sqrt{3}}$ :

Mixed constants with 5 operations  
 $1931851652578136 = 1/2/\sin(\pi/12)$

begründen Sie das (geometrisch oder schriftlich)

3. ein Polynom mit Nullstelle  $\sqrt[2]{2} + \sqrt[3]{3}$  bestimmen, nachrechnen.
4. Geonext: Satz von Napoleon illustrieren (gleichseitige Dreiecke über den Seiten eines beliebigen Dreiecks)
5. eigener Rechner: `rlwrap maxima` installieren,  
Rechner im Pool: `ssh` und `tmux` ausprobieren, auch Management von Sessions, Windows, Panes (split horizontal, vertikal), vgl. <https://news.ycombinator.com/item?id=26670708>

Organisatorisches:

- in Gitlab.Dit-Projekt einschreiben
- Hausaufgabe: Wiki anmelden, Issue: diskutieren, ggf. MR
- Prüfungszulassung: Hausaufgaben, autotool (empfohlen, nicht erzwungen)
- Prüfung: Klausur, ggf. Hilfspunkte aus Projekt (= ausgearbeitete Hausaufgabe o.ä.)

## 2 Zahlen

### Überblick

- exakte Zahlen: natürlich, ganz, rational  
Darstellungen: Positionssystem, Bruch  
Rechnungen: Addition, Multiplikation
- beliebig genau genäherte Zahlen  
(berechenbare reelle Zahlen)  
Darstellungen: Positions-System, Kettenbruch  
Rechnungen: arithmetische und irrationale Funktionen
- später:  
exaktes Rechnen mit algebraischen Zahlen

## Darstellung natürlicher Zahlen

- die Zahl  $n \in \mathbb{N}$  im Positionssystem zur Basis  $B$ :

$$n = \sum_{k \geq 0} x_k B^k$$

mit  $\forall i : 0 \leq x_i < B$  und  $\{i \mid x_i \neq 0\}$  endlich

$$\text{Bsp: } 25 = 1 \cdot 3^0 + 2 \cdot 3^1 + 2 \cdot 3^2 = [1, 2, 2]_3$$

- Darstellung ist eindeutig, kann durch fortgesetzte Division mit Rest bestimmt werden  
 $25 = 1 + 3 \cdot 8, 8 = 2 + 3 \cdot 2$
- praktisch wählt man die Basis
  - 10 für schriftliches Rechnen
  - historisch auch 60 (Zeit- und Winkelteilung)
  - 2 (oder  $2^w$ ) binäre Hardware, maschinennahe Software

## Natürliche Zahlen, Addition

- Darstellung

```
type Digit = Word64 ; data N = Z | C Digit N
```

- Semantik

```
value :: N -> Natural
value Z = 0 ; value (C x xs) = x + 2^64 * value xs
```

- Rechnung

```
instance Num N where (+) = plus_carry False
plus_carry :: Bool -> N -> N -> N
plus_carry cin (C x xs) (C y ys) =
  let z = bool 0 1 cin + x + y
      cout = z < x || z < y || ...
  in C z (plus_carry cout xs ys)
plus_carry ...
```

## Rekursive Multiplikation

- anstatt „Schulmethode“: rekursive Multiplikation

$$(p + qB)(r + sB) = pr + (ps + qr)B + qsB^2$$

$$\text{Bsp. } B = 100, (12 + 34 \cdot 100)(56 + 78 \cdot 100) = 12 \cdot 56 + \dots$$

$$(1 + 2 \cdot 10)(5 + 6 \cdot 10) = 1 \cdot 2 + \dots$$

- $M(w)$  = Anzahl elementarer Operationen (Plus, Mal) für Multiplikation  $w$ -stelliger Zahlen nach diesem Verfahren

$$M(1) = 1, M(w) = 4 \cdot M(w/2) + 2 \cdot w$$

$$M(2^k) = 3 \cdot 4^k - 2^{k+1}, \text{ also } M(w) \in \Theta(w^2)$$

- also wie bei Schulmethode, aber das läßt sich verbessern, und darauf muß man erstmal kommen

## Karatsuba-Multiplikation

- Anatoli Karatsuba, Juri Ofman 1962

$$\begin{aligned} (p + qB)(r + sB) &= pr + (ps + qr)B + qsB^2 \\ &= pr + ((p + q)(r + s) - pr - qs)B + qsB^2 \end{aligned}$$

- $K(w)$  = Anzahl elementarer Operationen (Plus, Mal) für Multiplikation  $w$ -stelliger Zahlen nach diesem Verfahren

$$K(1) = 1, K(w) = 3 \cdot K(w/2) + 5 \cdot w$$

(eine Multiplikation weniger, drei Additionen mehr)

- asymptotisch mit Ansatz  $K(w) \approx C \cdot w^e$

$$C \cdot w^e = 3Cw^e/2^e + 5w \Rightarrow 1 \approx 3/2^e, \text{ also } e = \log_2 3 \approx 1.58$$

- explizite Werte für  $w = 2^k$ ,  
ab wann besser als Schulmethode?

## Asymptotisch schnellere Multiplikation

- für  $X = \sum_k X_k B^k, Y = \sum_k Y_k B^k$ ,  
bestimme  $XY = \sum_k P_k B^k$  mit  $P_k = \sum_{i+j=k} X_i Y_j$

- Karatsuba (je zwei Blöcke)  $P_2 = X_1Y_1, P_0 = X_0Y_0, H = (X_1 + X_0)(Y_1 + Y_0)$ , dann  $P_1 = H - (P_0 + P_2)$
- Andrei Toom 1963: drei Blöcke, fünf Produkte:  
 $P_4, P_0, A = (X_2 + X_1 + X_0)(Y_2 + Y_1 + Y_0), B = (X_2 - X_1 + X_0)(Y_2 - Y_1 + Y_0),$   
 $C = ?$  (Übung)  
 Linearkombinationen (Bsp.)  $2P_2 = A + B - 2P_4 - 2P_0$
- Zeit  $T(n) = 5T(n/3) + \Theta(n), T \in \Theta(n^{\log_3 5}) = \Theta(n^{1.465})$
- $k$  Blöcke: Exponent  $\log_k(2k - 1)$ , aber steigende Faktoren
- Schönhage, Strassen 1971:  $O(n \log n \log \log n)$
- Harvey, van der Hoeven 2019:  $\Omega(n \log n)$

## Ganze Zahlen

- Darstellung: Bsp GMP (GNU Multi Precision Library)  
 Natürliche Zahl (magnitude) mit Vorzeichen (sign)  
 nicht Zweierkomplement, denn das ist nur für Rechnungen modulo  $2^w$  sinnvoll
- GHC (integer-gmp)  

```
data Integer = S Int | Jp BigNat | Jn BigNat
data BigNat = BN ByteArray
```
- NB: ganzzahlige Zahlbereiche in Haskell/GHC:
  - Int (ist immer die falsche Wahl)
  - Maschinenzahlen (mod  $2^{64}$ ): Int64, Word64,
  - beliebig große: Integer, Numeric.Natural

## Rationale Zahlen

- rationale Zahl  $q$  ist Paar von ganzen Zahlen  $(z, n)$   

```
data Q = Q Integer Integer
```
- normalisiert:  $n \geq 1$  und  $\gcd(|z|, |n|) = 1$

- normalisierte Darstellung ist eindeutig
- Vorteil:  
semantische Gleichheit (dieselbe Zahl wird bezeichnet) *ist* syntaktische Gleichheit (der Komponenten)
- Nachteil (?)  
nach jeder arithmetischen Operation normalisieren
- nicht normale Zahlen verhindern: Konstruktor  $\mathbb{Q}$  nicht exportieren (sonst `let q = Q 8 12`)

### Explosion der Stellenanzahl (Beispiel)

- Funktion  $f : \mathbb{Q} \rightarrow \mathbb{Q} : x \mapsto x/2 + 1/x$
- $x_k = f^k(1)$ ,  $x_0 = 1$ ,  $x_1 = 3/2$ ,  $x_2 = 17/12$ ,  $x_3 = 577/408$
- Zähler (und Nenner) in jedem Schritt (wenigstens) quadriert, d.h., Stellenzahl verdoppelt (Bsp:  $x_{10}$ )  
 $p/q \mapsto p/(2q) + q/p = (p^2 + 2q^2)/(2pq)$
- das ist typisch für Folgen arithmetischer Op.,  
Normalisierung wirkt nur selten verkleinernd.
- die Folge  $x_k$  nähert  $\sqrt{2}$  an:  $(p_k/q_k)^2 - 2 = 1/q_k^2$   
Konvergenz ist quadratisch (jeder Schritt quadriert den Fehler, verdoppelt Anzahl gültiger Stellen)

### Endliche und unendliche „Dezimal“-brüche

- Darstellug von Zahlen in  $\{x \mid 0 \leq x < 1\}$   
mit Basis  $1/B$  und Ziffern  $\in \{0 \dots B-1\}$
- Bsp:  $B = 10$ ,  $0.314 = 0 \cdot B^0 + 3 \cdot B^{-1} + 1 \cdot B^{-2} + 4 \cdot B^{-3}$
- hier sind auch unendliche Ziffernfolgen sinnvoll,  
bezeichnet Limes der Werte der endlichen Partialfolgen
- Konversion

```
decimal :: Rational -> [ Nat ]
decimal x = let q = truncate (fromRational x)
            in  q : decimal (10 * (x-q))
```

Anwendung: vorige Näherung von  $\sqrt{2}$

## Berechenbare reelle Zahlen

- beschrieben wird hier nicht die exakte (symbolische) Rechnung, sondern eine konvergente Näherung
- Def. reelle Zahl  $r$  mit  $0 \leq r < 1$  heißt *berechenbar* (zur Basis  $B$ ), wenn die Funktion  $d : \mathbb{N} \rightarrow \{0, \dots, B-1\}$ , welche die Darstellung von  $x$  zur Basis  $1/B$  bestimmt, berechenbar ist (z.B. durch eine Turingmaschine)
- jede rationale Zahl ist berechenbar  
(Beweis: Dezimalbruch ist endlich oder periodisch)
- $\sqrt{2}$  ist berechenbar (Beweis: vorige Folge  $x_k$ )
- Menge  $\mathbb{B}$  der berechenbaren Zahlen ist abgeschlossen unter arithmetischen Operationen (und weiteren)

## Potenzreihen, Exponentialfunktion

- Satz von Taylor: wenn  $f$  oft genug diff-bar,  
dann  $f(x_0 + d) = \sum_{k=0}^{n-1} f^{(k)}(x_0) d^k / k! + \Delta_n$   
mit  $\exists 0 \leq d' \leq d : \Delta_n = f^{(n)}(x_0 + d') d^n / n!$
- Bsp:  $f(x) = \exp(x)$ , dann  $f = f' = f'' = f^{(3)} = \dots$  und  
 $\exp(0 + d) = 1 + d + d^2/2 + d^3/6 + \dots$
- ```
take 100 $ decimal
$ sum $ take 100 $ scanl (/) (1::Rational) [1..]
==> [2,7,1,8,2,8,1,8,2,8,4,5,9,0,4 ... ]
```

  
Fehlerabschätzung? (reicht die zweite 100 für die erste?)
- $\exp(1) = \exp(1/2)^2$ , diese Reihe konvergiert schneller

## Potenzreihe für Wurzelfunktion

- Taylor-Reihe von  $\sqrt{x}$  an der Stelle 1

Ableitungen mit maxima:

$$\text{diff}(\text{sqrt}(x), x, 5); 105/32 \cdot x^{-9/2}$$

$$\text{diff}(\text{sqrt}(x), x, 6); -945/64 \cdot x^{-11/2}$$

$$\text{Vermutung } f^{(n)}(1) = (-1)^{n+1} \cdot (2n-3)!! \cdot 2^{-n}$$

- $\sqrt{1+d} = 1 + \sum_{k>0} (-1)^{k+1} \frac{(2k-3)!!}{2k!!} d^k$   
 $= 1 + \frac{1}{2} \cdot d - \frac{1 \cdot 1}{2 \cdot 4} \cdot d^2 + \frac{1 \cdot 1 \cdot 3}{2 \cdot 4 \cdot 6} \cdot d^3 - \frac{1 \cdot 1 \cdot 3 \cdot 5}{2 \cdot 4 \cdot 6 \cdot 8} \cdot d^4 \pm \dots$
- für  $\sqrt{2}$ : Berechnung als  $\sqrt{1+1}$  konvergiert langsam  
besser:  $\sqrt{2} = 7/5 \sqrt{1+1/49}$

## Logarithmen

Taylor-Entwicklung  $\log(1+x) = x - x^2/2 + x^3/3 - \dots$   
konvergiert sehr langsam für  $x = 1$ , gar nicht für größere  $x$ .

J.R. Young, 1835, siehe v. Mangoldt/Knopp, Bd. 2, S. 127

$$a = \log(16/15) = 4 \log 2 - 1 \log 3 - \log 5,$$

$$b = \log(25/24) = \dots$$

$$c = \log(81/80) = \dots$$

Umstellung ergibt  $\log 2 = 7a + 5b + 3c, \dots$  Aufgaben:

- alle Koeffizienten ausrechnen
- wie genau sind die Werte für  $\log 2$  usw., wenn man nur die erste Näherung benutzt, also  $\log(1+x) \approx x$
- woher bekommt man geeignete Brüche (z. B. für  $\log 7$ )?

## Pi

darüber gibt es ganze Bücher (Aufgabe: finde Beispiele)

Ansatz: Taylor-Entwicklung von  $\arctan x$

$$\arctan x = x - x^3/3 + x^5/5 - \dots$$

- betrachte  $x = 1/5$ ,  $\alpha = \arctan x$ ,  
bestimme  $\tan(4\alpha)$  (ist nahe bei 1)
- bestimme  $\beta = 4\alpha - \pi/4$   
und  $y$  mit  $\beta = \arctan y$  (ist nahe bei 0)
- $\pi/4 = 4 \arctan x - \arctan y$

(J. Machin, 1706, 100 Stellen; W. Shanks, 1873, 707 St.)

## Approximiertes Rechnen

- auf Gleitkomma-Zahlen begrenzter Stellenzahl
- in CPU/FPU, durch Microcode, Polynom-Approximation  
Ken Shirriff *Pi in the Pentium* <https://www.righto.com/2025/01/pentium-floating-point.html>
- in spezieller Hardware (Wertetabelle)  
Ken Shirriff *Reverse Engineering the Yamaha DX7 synthesizer's sound chip* <https://www.righto.com/2021/11/reverse-engineering-yamaha-dx7.html>

## Hausaufgaben

WS26 empfohlen: 2, 4, 6 (andere sind nicht verboten)

1. Multiplikation in GMP (GNU Multiprecision Library) <https://gmplib.org/manual/Multiplication-Algorithms>

- (a) Karatsuba-Rechnung ist dort etwas anders als hier auf der Folie, warum?
- (b) GHC verwendet GMP für den Typ `Integer`.  
Bestimmen Sie experimentell den Anstieg der Rechenzeit bei Verdopplung der Stellenzahl, z.B.

```
:set +s
x = 10^10^8 :: Integer
odd x -- damit x ausgewertet wird
odd ((x-1)*(x+1)) -- die eigentliche Messung
True
(3.73 secs, 166,159,792 bytes)
y = x*x -- hat doppelte Stellenzahl
```

Ist die Anzahl der Bytes plausibel?

Diskutieren Sie mögliche verkürzte Auswertungen für `odd . . .` Kann GMP/GHC das?

(c) Zusatz: warum ist (oder erscheint)  $(x+1)^2$  schneller als  $(x+1) * (x+1)$ ?

2. zu Multiplikation nach Toom (3 Blöcke): ergänzen Sie den Ausdruck  $C$  sowie die fehlenden Linearkombinationen

3. diskutieren (Zusatz: implementieren) Sie die Darstellung von ganzen Zahlen mit negativer Basis  $B \leq -2$

(und nichtnegativen Ziffern  $\in \{0, \dots, |B| - 1\}$  wie bisher)

Bsp:  $B = -2$ ,

$$-3 = 1 \cdot B^0 + 0 \cdot B^1 + 1 \cdot B^2 + 1 \cdot B^3 = 1 + 0 + 4 - 8$$

(a) Eindeutigkeit, Konstruktion

(b) Arithmetik (Nachfolger, Addition, Multiplikation)

4. Bestimmen Sie die Taylor-Reihe für den Arcustangens an der Stelle 0

wie auf Folie *Potenzreihe für Wurzelfunktion*

Bestimmen Sie damit  $x = \arctan(1/2)$ ,  $y = \arctan(1/3)$  auf (z.B.) 20 Stellen.

Begründen Sie  $x + y = \pi/4$ . Rechnen Sie den Wert für  $\pi$  aus und vergleichen Sie mit einer verlässlichen Quelle.

Kann man  $\pi$  nach diesem Verfahren, aber mit anderen Parametern, besser bestimmen? (mehr Stellen bei gleichem Aufwand)

5. Bestimmen Sie die Taylor-Reihe für den (natürlichen) Logarithmus an der Stelle 1. Bestimmen Sie damit  $a = \log(6/5)$ ,  $b = \log(9/8)$ ,  $c = \log(10/9)$  auf (z.B.) 100 Stellen und daraus  $\log 2$  als eine Linearkombination.

6. wie bestimmt man  $\sqrt{3}$ ,  $\sqrt{5}$ ,  $\sqrt[3]{2}$

Hinweis:  $\sqrt[3]{2}$  als Fixpunkt von  $x \mapsto (2x + 2/x^2)/3$ ,

diese Gewichte ergeben quadratische Konvergenz,

ist äquivalent zu Bestimmung der Nullstelle von  $f(x) = x^3 - 2$  nach Newton-Verfahren:

$$f'(x) = 3x^2, x \mapsto x - f(x)/f'(x) = \dots$$

7. Was macht dieses Programm?

```
int n = 1000;
int [] d = new int [n];
```

```

for (int i=2; i<n; i++) { d[i]=1; }
for (int j=2; j<n; j++) {
    int c = 0;
    for (int i=n-1; i>=1; i--) {
        int h = 10 * d[i] + c;
        d[i] = h % i; c = h / i;
    }
    System.out.print(c); System.out.flush();
}

```

Ist die Ausgabe korrekt? Wie weit darf man n erhöhen?

8. Jerzy Karczmarczuk: *The Most Unreliable Technique in the World to compute pi*, 2003? <https://web.archive.org/web/20051017081559/http://users.info.unicaen.fr/~karczma/arpap/lazypi.ps.gz>

### Plan (vorläufig)

- (15) Einleitung, symbolisches Diff.
- (17) Automatisches Differenzieren
- (19) Polynome (Repräsent., Operationen) Polynomideale
- (20) Monom-Ordnungen, Gröbnerbasen (GB)
- (21) Anwendung GB in Geometrie
- (23) abstrakte Reduktionssysteme (ARS) Term-Ersetzungs-Systeme (TRS)
- (24) Termination
- (25) Unifikation, Vervollständigung
- (26) Curry-Howard-Isomorphie (Programm = Beweis), (daten)abhängige Typen
- (27) interaktives Beweisen
- (28) Zusammenfassung, Ausblick