Symbolisches Rechnen
Vorlesung
Wintersemester 2006, 2014
Sommersemester 2021, 2025

Johannes Waldmann, HTWK Leipzig

29. April 2025

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)

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

  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) Aufgaben*klassen*
  (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 un 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`):

```
D(t) -> 1          D(constant) -> 0
D(+(x,y)) -> +(D(x),D(y))
D(*(x,y)) -> +(*(y,D(x)),*(x,D(y)))
D(-(x,y)) -> -(D(x),D(y))
```

Robert Floyd 1967, zitiert in: Nachum Dershowitz: *33 Examples of Termination*,
`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
                                   4      2
K satisfies the polynomial, X  - 4 X  + 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
- Hausaugabe: Wiki anmelden, Issue: diskutieren, ggf. MR
- Prüfungszulassung: Hausaufgaben, autotool (empfohlen, nicht erzwungen)
- Prüfung: Klausur, ggf. Hilfspunkte aus Projekt (= ausgearbeitete Hausaufgabe o.ä.)

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
  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
  $(p + qB)(r + sB) = pr + (ps + qr)B + qsB^2$
  Bsp. $B = 100$, $(12 + 34 \cdot 100)(56 + 78 \cdot 100) = 12 \cdot 56 + \ldots$
  $(1 + 2 \cdot 10)(5 + 6 \cdot 10) = 1 \cdot 2 + \ldots$
- $M(w) =$ Anzahl elementarer Operationen (Plus, Mal) für Multiplikation w-stelliger Zahlen nach diesem Verfahren
  $M(1) = 1, M(w) = 4 \cdot M(w/2) + 2 \cdot w$
  $M(2^k) = 3 \cdot 4^k - 2^{k+1}$, 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
- $(p + qB)(r + sB) = pr + (ps + qr)B + qsB^2$
  $= pr + ((p + q)(r + s) - pr - qs)B + qsB^2$
- $K(w) =$ 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 $K(w) \approx C \cdot w^e$
  $C \cdot w^e = 3Cw^e/2^e + 5w \Rightarrow 1 \approx 3/2^e$, also $e = \log_2 3 \approx 1.58$
- explizite Werte für $w = 2^k$,
  ab wann besser als Schulmethode?

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 `Q` nicht exportieren (sonst `let q = Q 8 12`)

Explosion der Stellenanzahl (Beispiel)

- Funktion $f : \mathbb{Q} \to \mathbb{Q} : x \mapsto x/2 + 1/x$
- $x_k = f^k(1)$, $x_0 = 1, x_1 = 3/2, x_2 = 17/12, x_3 = 577/408$
- 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 $\{x \mid 0 \le x < 1\}$
  mit Basis $1/B$ und Ziffern $\in \{0 \ldots B-1\}$
- 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 $0 \le r < 1$ heißt *berechenbar* (zur Basis B), wenn die Funktion $d : \mathbb{N} \to \{0, \ldots, B-1\}$, 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 $\exists 0 \le d' \le d : \Delta_n = f^{(n)}(x_0 + d') d^n/n!$
- Bsp: $f(x) = \exp(x)$, dann $f = f' = f'' = f^{(3)} = \ldots$ und
  $\exp(0 + d) = 1 + d + d^2/2 + d^3/6 + \ldots$
- ```
  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:
  `diff(sqrt(x),x,5);` $105/32 \cdot x^{-9/2}$
  `diff(sqrt(x),x,6);` $-945/64 \cdot x^{-11/2}$
  Vermutung $f^{(n)}(1) = (-1)^{n+1} \cdot (2n-3)!! \cdot 2^{-n}$
- $\sqrt{1+d} = 1 + \sum_{k>0} (-1)^{k+1} \frac{(2k-3)!!}{2k!!} d^k$
  $= 1 + \frac{1}{2} \cdot d - \frac{1 \cdot 1}{2 \cdot 4} \cdot d^2 + \frac{1 \cdot 1 \cdot 3}{2 \cdot 4 \cdot 6} \cdot d^3 - \frac{1 \cdot 1 \cdot 3 \cdot 5}{2 \cdot 4 \cdot 6 \cdot 8} \cdot d^4 \pm \ldots$
- für $\sqrt{2}$: Berechnung als $\sqrt{1+1}$ konvergiert langsam
  besser: $\sqrt{2} = 7/5\sqrt{1 + 1/49}$

Logarithmen

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

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

$$\begin{aligned} a = \log(16/15) &= 4\log 2 - 1\log 3 - \log 5, \\ b = \log(25/24) &= \ldots \\ c = \log(81/80) &= \ldots \end{aligned}$$

Umstellung ergibt $\log 2 = 7a + 5b + 3c, \ldots$ 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 $\arctan x$

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

- 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.)

Hausaufgaben

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 Messu
  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. diskutieren (Zusatz: implementieren) Sie die Darstellung von ganzen Zahlen mit negativer Basis $B \le -2$
   (und nichtnegativen Ziffern $\in \{0, \ldots, |B|-1\}$ 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)
```

3. 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 π aus und vergleichen Sie mit einer verläßlichen Quelle.
   Kann man π nach diesem Verfahren, aber mit anderen Parametern, besser bestimmen? (mehr Stellen bei gleichem Aufwand)
4. 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.

5. 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:
   $f'(x) = 3x^2, x \mapsto x - f(x)/f'(x) = \ldots$
6. 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`

Automatische Differentiation

Motivation, Anwendung

- Optimierungs-Aufgaben, Bsp: Koeffizienten eines neuronalen Netzes (gegebener Topologie) zur bestmöglichen Approximation einer gegebenen Funktion
- oder Aufgaben, die sich als O-A formulieren lassen,
  Bsp: Lösung eines (nicht linearen) Ungleichungs-Systems $\bigwedge_i L_i(x) \geq R_i(x)$ für $x \in \mathbb{R}^n$
  äq: $0 \stackrel{?}{=} \min_x \sum_i \max(-(L_i(x) - R_i(x)), 0)$
- lösen durch Folge $x^{(k+1)} = x^{(k)} + \alpha \cdot d$, mit $\alpha \in \mathbb{R}, d \in \mathbb{R}^n$
  wobei Schritt-Richtung $d = -(\nabla f)(x^{(k)})$, Gradient $(\nabla f)x := [\partial f/\partial x_i \mid 1 \leq i \leq n]$ = Vektor der partiellen Ablt.

Verfahren zur Gradientenbestimmung

- Aufgabenstellung (Spezifikation):
  - gegeben: Vektor $x \in \mathbb{R}^d$, arithmetische Fkt. $f : \mathbb{R}^d \to \mathbb{R}$ als symbolischer Ausdruck (Term, DAG)
  - gesucht: $(\nabla f)(x)$, Wert des Gradienten an Stelle x
- verschiedene Lösungsverfahren:
  - *symbolische* Differentiation: Term für ∇f
    ... der kann aber sehr groß werden
  - *numerische* Diff.: berechnet $(f(x + h \cdot \vec{e_i}) - f(x))/h$
    ... anfällig für Rundungs- und Auslöschungs-Fehler
  - *automatische* Diff.: symbolische Rechng. auf DAG von f
- Baydin, Pearlmutter, Radul, Siskind: *Automatic differentiation in machine learning: a survey*, 2015–2018
  `https://arxiv.org/abs/1502.05767`

Automatische Differentiation (AD)

- Implementierung: repräsentiere Funktionswert *und* Gradient in *einem* Objekt, dessen Typ impl. `Num`

```
data N v z = N {abs :: z, lin :: M.Map v z}

var::v->z->N v z; var v z = N z (M.singleton v 1)

instance (Ord v, Num z) => Num (N v z) where
  fromInteger i = N (fromInteger i) M.empty
  N a1 l1+N a2 l2 = N(a1+a2)(M.unionWith (+) l1 l2)
```

- Anwendung: $\sqrt{2}$ durch $\arg\min f$ mit $fx = (x^2 - 2)^2$

```
f :: Num a => a -> a ; f x = (x^2 - 2)^2
f @(Fixed E6) 1.4  ==> 0.001600
f @(N () (Fixed E6) ) (var () 1.4) ==>
  N 0.001600 (fromList [((),-0.224000)])
```


Stochastischer Gradientenabstieg

- Bsp: gesucht ist Netz mit zwei Schichten für XOR.
- volles Optimierungsproblem: minimiere Fehlerquadratsumme über alle (4) Eingabe/Ausgabe-Paare (benutze Gradient dieser Fkt.)
- Variante: in jedem Schritt ein E/A-Paar zufällig auswählen
  - weniger Rechenaufwand pro Schritt
  - Ausbruch aus lokalen Minima (voller Gradient $= 0$), die global schlecht sind
- Variante: je Schritt: betrachte eine zufällige Teilmenge der Parameter als konstant (Gradient nur für die anderen)
  - weniger Rechenaufwand pro Schritt
- Details: siehe VL Numerik für Maschinelles Lernen

Anwendung: Matrix-Interpretationen

- das Wort-Ersetzungs-System $R = \{ab \to ba\}$ terminiert (jede Ableitung ist endlich),
  denn $i : a \mapsto \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}, b \mapsto \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$
  ist zu R kompatible Interpretation in wohlfundiertes monotones Monoid $(\begin{pmatrix} \geq 1 & \geq 0 \\ \geq 0 & \geq 1 \end{pmatrix}, \circ, >_M)$
  mit $x >_M y := x_{1,2} > y_{1,2} \wedge \forall i, j : x_{i,j} \geq y_{i,j}$
  kompatibel: $i(ab) = \begin{pmatrix} 2 & 2 \\ 0 & 1 \end{pmatrix} >_M \begin{pmatrix} 2 & 1 \\ 0 & 1 \end{pmatrix} = i(ba)$
- solche Interpretation für $\{aa \to aba\}$, $\{a^2b^2 \to b^3a^3\}$?

Constraint-Lösen durch Optimieren

- gesucht sind $a, b \in \mathbb{R}^{3\times3}$ mit $a, b \in M$, $a^2 >_M aba$
- Constraints für $2 \times 3 \times 3$ Unbekannte: $a_{i,j} \geq 0, , b_{i,j} \geq 0$
  ersetze durch Fkt. von unbeschränkten $x_{i,j} \in \mathbb{R}$
  Bsp.: $a_{i,j} = x_{i,j}^2$, $a_{i,j} = \max(0, x_{i,j})$, $a_{i,j} = \max(x_{i,j}, -x_{i,j})$
  AD funktioniert auch für stückweise Funktionen
  `instance Ord (N v z) where ...`
- 3×3 Constraints für Produkte: $(a^2)_{i,j} \geq (aba)_{i,j}$
  Ungl. $f \geq g$ realisieren durch Strafterm $\max(-(f - g), 0)$
  zu minimieren ist Summe der Strafen, min soll 0 sein
- Modellierung so, daß der Gradient nützlich ist (also Strafterm *nicht*: if $f < g$ then 1 else 0)
- Constraints (Ungl.) *exakt* erfüllen (in $\mathbb{Q}$, nicht `Double`)

Literatur und Software (funktionaler) AD

- Jerzy Karczmarczuk *Functional Differentiation of Computer Programs.* ICFP 1998 `https://dl.acm.org/doi/10.1145/289423.289442`
- Faustyna Krawiec, Simon Peyton Jones, Neel Krishnaswami, Tom Ellis, Richard A. Eisenberg, Andrew Fitzgibbon: *Provably correct, asymptotically efficient, higher-order reverse-mode automatic differentiation*, POPL 2022, `https://dl.acm.org/doi/10.1145/3498710` Implementierung: Mikolaj Konarski, `https://github.com/Mikolaj/horde-ad`
- Edward Kmett 2010–, `https://hackage.haskell.org/package/ad`

Aufgaben

1. (H. Rosenbrock 1960) ein Test für Abstiegsverfahren ist $f(x,y) = (x^2 - y)^2 + y^2/100$,
   Veranschaulichen Sie den Werteverlauf mit `rlwrap maxima`

```
f : (x^2-y)^2 + y^2/100;
plot3d(f,[x,-50,50],[y,-1000,3000]);
```

   wo ist das globale Minimum?

   in welche Richtung geht der (entgegengesetze) Gradient im Punkt $(20, 100)$,

   wo verläuft die Folge der Näherungswerte, wird das globale Minimum erreicht? (1. diskutieren, 2. ausprobieren)

2. für den Typ `N v z`: weitere Funktionen implementieren (Division, recip, sqrt, exp, log),
   den so bestimmten Wert des Gradienten mit numerischer Differentiation vergleichen
   diese Fkt. in einer Minimum-Bestimmung verwenden
3. in neuronalen Netzen wird oft die Funktion $S : x \mapsto 1/(1 + \exp(-x))$ verwendet (sie bildet $\mathbb{R}$ monoton steigend auf $[0, 1]$ ab)
   bestimmen Sie deren Ableitung symbolisch (zu Fuß oder Maxima), vergleichen mit AD-Implementierung (vorige Aufgabe)
4. (Fortsetzung) ein k-stelliges Neuron mit Gewichten $w_0, w_1, \ldots, w_k \in \mathbb{R}$ ist die Funktion $x \mapsto S(w_0 + \sum_i w_i x_i)$.

   ein vollständig verbundenes Netz mit e Eingängen und n Neuronen: das Neuron i sieht alle Eingänge sowie die Ausgänge der Neuronen $1 \ldots i-1$.
   Bestimmen Sie (mit stochastischem Gradientenabstieg mit AD) die Gewichte eines vollständigen Netzes (mit möglichst wenig Neuronen) für
   - XOR über alle Eingänge
   - Majoriät (falls s ungerade)
5. für Optimierungsverfahren 2. Ordnung benötigt man die Hesse-Matrix einer Funktion (diese enthält alle zweiten Ableitungen). Modellieren und berechnen Sie diese durch eine Erweiterung des Typs `data N v z`.
   Vergleichen Sie mit exakten Werten (Bsp. maxima: `hessian((x^2-y)^2,[x,y]);`)

6. (Forschungsaufgabe) Matrix-Interpretationen für Wortersetzungssysteme
   - verbesserte Modellierung (der Zielfunktion) und Parameter (Schrittweite)
   - stochastischer Abstieg (1. Ordnung)
   - Abstiegsverfahren 2. Ord. (oder Quasi-Newton)

   Testfälle
   - (leicht) $a \to b$, $ab \to ba$, $ab \to bba$, $aa \to aba$,
   - (schwerer) $a^2b^2 \to b^3a^3$, $\{a^2 \to bc, b^2 \to ac, c^2 \to ab\}$
     System mit 3 Regeln: es genügt $>_M$ für eine Regel, für die anderen $\geq_M$. Wie kann man diese Bedingung als Zielfunktion einer Optimierung realisieren?
     ganzzahlige Lösungen sind bekannt (gefunden durch Bit-Blasting) gibt es kleinere nicht-ganzzahlige?