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

Johannes Waldmann, HTWK Leipzig

9. November 2026

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?

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) $P_2 = X_1 Y_1, P_0 = X_0 Y_0$,
  $H = (X_1 + X_0)(Y_1 + Y_0)$, 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 `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 $\in \{0 \dots B-1\}$
- 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 $0 \le r < 1$ heißt *berechenbar* (zur Basis B), wenn die Funktion $d : \mathbb{N} \to \{0, \dots, B-1\}$, 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 $\exists 0 \le d' \le d : \Delta_n = f^{(n)}(x_0 + d') d^n/n!$
- Bsp: $f(x) = \exp(x)$, dann $f = f' = f'' = f^{(3)} = \dots$ 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:
  `diff(sqrt(x),x,5);` $105/32 \cdot x^{-9/2}$
  `diff(sqrt(x),x,6);` $-945/64 \cdot x^{-11/2}$
  Vermutung $f^{(n)}(1) = (-1)^{n+1} \cdot (2n-3)!! \cdot 2^{-n}$
- $\sqrt{1+d} = 1 + \sum_{k>0} (-1)^{k+1} \frac{(2k-3)!!}{2k!!} d^k$
  $= 1 + \frac{1}{2} \cdot d - \frac{1 \cdot 1}{2 \cdot 4} \cdot d^2 + \frac{1 \cdot 1 \cdot 3}{2 \cdot 4 \cdot 6} \cdot d^3 - \frac{1 \cdot 1 \cdot 3 \cdot 5}{2 \cdot 4 \cdot 6 \cdot 8} \cdot d^4 \pm \dots$
- 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

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

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-ROM.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 $\in \{0, \ldots, |B| - 1\}$ 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)

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 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)

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

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`

Polynome

Motivation (I): Polynom-Interpretationen

- Anwendung: wfmA $A = (\mathbb{N}, >, [\cdot])$, jedes $[f]_A$ ist Polynom.
  Bsp: $[f](x, y) = x^2 \cdot y$ (Vorsicht), $[g](x, y) = x^2 + y$
  Bsp: kompatibel mit $g(g(x, y), z) \to g(x, g(y, z))$?
- es müssen Aussagen der Form $\forall x, y, z : P(x, y, z) > 0$ (automatisch) nachgewiesen werden
- welche Approximation in der autotool-Aufgabe dazu?
- Zeige $x, y \geq 0 \Rightarrow (x + y)/2 \geq \sqrt[2]{xy}$
  Ü: Zeige $x, y, z \geq 0 \Rightarrow (x + y + z)/3 \geq \sqrt[3]{xyz}$
  Ü: Hilberts 17. Problem (nichtneg. P, das kein SOS ist)

Motivation (II): Algebraische Zahlen

Bsp: gesucht ist Minimalpolynom für $y = \sqrt{2} + \sqrt[3]{3}$

	1	$\sqrt[3]{3}$	$\sqrt{2}$	$\sqrt{2}\sqrt[3]{3}$	$\sqrt[3]{3^2}$	$\sqrt{2}\sqrt[3]{3^2}$
y^0	1	0	0	0	0	0
y^1	0	1	1	0	0	0
y^2	2	0	0	2	1	0
y^3	3	6	2	0	0	3
y^4	4	3	12	8	12	0
y^5	60	20	4	15	3	20
y^6	17	90	120	24	60	18

die letzte Zeile ist linear abhängig von den vorigen (Dimension des Vektorraumes ist ≤ 6), ergibt Darstellung von y^6 als rationale Linearkombination von $y^0, \ldots, y^5$.

Motivation (III): Geometrische Örter

- Satz: in jedem Dreieck liegen
  - die Seitenmitten
  - die Höhenfußpunkte
  - die Mitten der oberen Höhenabschnitte

  auf einem Kreis (Feuerbach-Kreis, 9-Punkte-Kreis).
- Beweis durch symbolisches Rechnen:
  - Koordinaten der Ecken $(A_1, A_2), (B_1, B_2), (C_1, C_2)$
  - ex. Polynom P, so daß: (X_1, X_2) liegt auf Umkreis der Seitenmitten gdw $P(A_1, A_1, B_1, B_2, C_1, C_2, X_1, X_2) = 0$
  - (X_1, X_2) ist ein Höhenfußpunkt: $Q(\ldots) = 0$
  - zu zeigen ist $Q(\ldots) = 0 \Rightarrow P(\ldots) = 0$
    das ist (später) eine Aussage über Polynom-Ideale

Semantik und Syntax von Polynomen

- Polynom (semantisch) als Funktion: $\lambda x.x^3 - 3x^2 + 3x - 1$
- Polynom (syntaktisch)
  als Menge von Monomen $\{1x^3, -3x^2, 3x^1, -1x^0\}$,
  als Folge von Koeffizienten $[1, -3, 3, -1]$
- symbolisches Rechnen mit Polynomen (elementare Arithmetik)
- als Grundlage für (später) Algorithmus von Buchberger zur Bestimmung von Gröberbasen
- Wdhlg. (Zahlen)bereiche (Gruppen, Ringe, Körper), Anwendung: Polynome in mehreren Variablen

Gruppen, Ringe, Körper

- Definitionen (Wiederholung)
  - Monoid: Addition, Null
  - Gruppe: wie Monoid, und Subtraktion
  - Ring: wie Gruppe und zusätzlich Multiplikation, Eins
  - Körper: wie Ring und zusätzlich Division
  - Vektorraum (ü. Körper): Gruppe, skalare Multiplikation
- Bsp: $\mathbb{N}, \mathbb{Z}, \mathbb{Z}/6\mathbb{Z}, \mathbb{Z}/7\mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{R}^2, \mathbb{C}, \mathbb{Z}[X]$
- zusätzliche Eigenschaften (Beispiele)
  - Ring heißt nullteilerfrei: $\forall a, b : a \neq 0 \wedge b \neq 0 \Rightarrow (ab) \neq 0$
  - Ring heißt euklidisch (bzgl. Funktion $|\cdot| : R \to \mathbb{N}$), wenn $\forall a, b \neq 0 : \exists q, r : a = q \cdot b + r$ mit $|r| < |b|$

Polynome in einer Variablen

- Repräsentation:
  - dicht: Folge (Data.Sequence) der Koeffizienten
  - dünn: endliche Abbildung (Data.Map)
    Schlüssel: Grad, Wert: Koeffizient
    normalisierte Darstellung: nur Koeffizienten $\neq 0$ werden repräsentiert (dann einfacher Test auf $P = 0$)
- Auswertung: mit Horner-Schema
- Addition
  - in dichter Darstellung: `Data.List.zipWith (+)`
  - in dünner Darstellung: `Data.Map.unionWith (+)`
- Multiplikation: naiv: quadratisch, mit FFT: $n \log n$

Polynom-Division (über $\mathbb{Q}$)

- Beispiel: $A = (x^5 - 1), B = (x^2 - 1), \quad A : B =$ Quotient $Q = x^3 + x$, Rest $R = x - 1$
- Spezifikation: $A : B = Q$ Rest R gdw. $A = B \cdot Q + R \wedge \deg R < \deg B$.
- Bezeichnung: $\deg P$ (Grad) = der höchste Exponent,
  $\deg(x^5 - 1) = 5, \deg(4) = 0, \deg(0) = -\infty$.
- Implementierung $A : B$: falls $\deg A \geq \deg B$, dann $(c_1 \cdot x^{e_1} = \operatorname{lead} A$ (höchstes Monom), $(c_2 \cdot x^{e_2} = \operatorname{lead} B$ (höchstes Monom), $q = (c_1/c_2)x^{e_1 - e_2)}$, $A' = A - qB$, $(R', Q') = A' : B$, Resultat ...

Polynome in mehreren Variablen

- das wird in den allermeisten Anwendungen gebraucht
- mögliche Repräsentationen für $\mathbb{Z}[X_1, X_2, \ldots]$
  - geschachtelt: $= (\mathbb{Z}[X_1])[X_2, \ldots]$
    als 1-Var-Polynom, Koeffizienten sind $(n-1)$-Var-Pol.
  - flach: als endliche Abbildung
    Schlüssel: Monom (Bsp: $X_2^8 X_4^5$), Wert: Koeffizient
    Monom: als endliche Abbildung
    Schlüssel: Variable (Bsp: X_2, X_4) Wert: Exponent
- die geschachtelte Darstellung ist elegant (und man muß nichts neu programmieren)
  kann aber unpraktisch sein (wenn man „von oben“ eine innere Variable benutzt)

Polynom-Darstellung als geordnete Liste

- Mult. von Polynom mit Zahl ist strukturerhaltend

  Struktur `M.Map Natural c` wird hier nicht benötigt, könnte auch `[(Natural,c)]` sein (Liste, Invariante: aufsteigend nach Schlüssel, besser: absteigend, wg. Zugriff auf head)

  dafür effiziente Addition, Multiplikation?

- Addition: wie `merge`. Dabei Normalisierung

- Multiplikation: wiederholte Addition mit Verschiebung

Listen-Darstellung f. mehr Var.

- dünne Darstellung ist
  `type Mono v = M.Map v Natural,`
  `type Poly v c = M.Map (Mono v) c`
- jeweils `Map s w` ersetzen durch `[(s, w)]` mit Invariante: absteigend nach `s`
  effiziente Implementierung arithmetischer Op.?
- Addition von Polynomen (benutzt `merge`)
- Multiplikation von Polynomen: benutzt: von Monomen die Monom-Ordnung muß monoton sein bzgl. Multiplikation
  wenn $m_1 > m_2$, dann $m \cdot m_1 > m \cdot m_2$.
  Beispiele: lexikografisch; Exponentensumme, dann lex.

Ideale

- Def: in Ring $R = (M, 0, +, 1, \cdot)$: Menge $I \subseteq M$ heißt *Ideal*,
  – wenn abgeschlossen unter Summe (mit sich)
  $\forall x \in I, y \in I : (x + y) \in I$.
  – und abgeschlossen bzgl. Multiplikation mit M (mit allen)
  $\forall x \in I, y \in M : (xy) \in I$.
- Def: das von $B \subseteq M$ *erzeugte* Ideal, Notation $\langle B \rangle$,
  ist das kleinste (bzgl. Inklusion) Ideal I mit $B \subseteq I$
  Satz: $\mathrm{Ideal}(B) = \langle B \rangle = \{\sum c_i b_i \mid c_i \in M, b_i \in B\}$
  Bsp: in $\mathbb{Z}$: $\mathrm{Ideal}(\{8, 12\}) = \langle 8, 12 \rangle = ?$
  Bsp: in $\mathbb{Q}[X, Y]$: $\mathrm{Ideal}(1)$, $\mathrm{Ideal}(0)$, $\mathrm{Ideal}(\{X + Y, X - Y\})$
- Bsp: gilt $(X^5 - Y^2) \in \mathrm{Ideal}(\{X^2Y - 1, XY^2 - 1\})$?
  nächste VL: Algorithmus für Polynom-Ideal-Mitgliedschaft

Hausaufgaben

1. Bezeichnungen:
   arithmetisches Mittel $A(x_1, \ldots, x_n) = (\sum x_i)/n$,
   geometrisches Mittel $G(x_1, \ldots, x_n) = \sqrt[n]{\prod x_i}$.
   Aufgabe: Stelle $A(x_1, \ldots, x_4)^4 - G(x_1, \ldots, x_4)^4$ als Summe von Quadraten (SOS) von Polynomen dar.
   Hinweis: $A(x_1, x_2)^2 - G(x_1, x_2)^2 = (x_1 - x_2)^2/4$ und
   $A(x_1, x_2, x_3, x_4) = A(A(x_1, x_2), A(x_3, x_4)) \geq$
   $A(G(x_1, x_2), G(x_3, x_4)) \geq G(G(x_1, x_2), G(x_3, x_4)) =$
   $G(x_1, x_2, x_3, x_4)$.
   Zusatz: $A(x_1, x_2, x_3)^3 - G(x_1, x_2, x_3)^3$ ist kein SOS, kann aber als Bruch mit SOS im Zähler und $(x + y + z)$ im Nenner geschrieben werden. Dazu die SOS-Darstellung

   der A-G-Ungleichung für die 4 Werte
   $x_1, x_2, x_3, t = A(x_1, x_2, x_3)$ benutzen, denn
   $A(x_1, x_2, x_3, t) = t$ und $G(x_1, x_2, x_3, t)^4 = G(x_1, x_2, x_3)^3 \cdot t$.
   Hintergrund: Bruce Reznick: *Some Concrete Aspects of Hilbert's 17th Problem*, 2000, `https://faculty.math.illinois.edu/~reznick/`.

2. das Minimalpolynom für $\sqrt{2} + \sqrt[3]{3}$ nach angegebenem Verfahren ausrechnen und überprüfen.
   Ähnlich für $\sqrt{3} + \sqrt{5}$ oder (z.B.) $\sqrt[5]{3} + \sqrt[5]{5}$

3. Das Polynom P für „liegt auf Umkreis der Seitenmitten“ angeben (oder ähnlicher geometrischer Ort im Dreieck).
   o.B.d.A $A_1 = A_2 = B_1 = 0$ annehmen.
   Warum wäre zusätzlich $C_2 = 0$ doch eine B.d.A.?

4. für die in VL angegebene Implementierung von Polynomen:
   eine anderen Koeffizientenbereich (als `Rational`) benutzen, z.B. `Complex Rational`
   nichttriviale Rechnungen durchführen (z.B. Division von $X^{pq} - 1$ durch $X^p - 1$), Ergebnis prüfen,
   Laufzeit messen, Ausführung profilieren, teure Funktionen feststellen, ggf. verbessern
   Vergleichen mit derselben Rechnung in einem richtigen Computer-Algebra-System (maxima, fricas). (Nicht irgendwo online, sondern lokal, damit man messen und vergleichen kann.)

5. für $P = X^2Y + XY^2 + Y^2, Q = XY - 1, R = Y^2 - 1$:
   ist $P \in \mathrm{Ideal}(Q, R)$?
   Hinweis: nein. Hinweis: Betrachten Sie Werte von P für die gemeinsamen Nullstellen von Q, R.

6. gilt $X^5 - Y^2 \in \mathrm{Ideal}(X^2Y - 1, XY^2 - 1)$?
   Welches Resultat liefert der Nullstellentest?

7. Zeigen Sie $\mathrm{Ideal}(X + XY, Y + XY, X^2, Y^2) = \mathrm{Ideal}(X, Y)$.

8. Entscheiden Sie $X^2 + 1 \in \mathrm{Ideal}(X + 1)$,
   $X^2 - y^2 \in \mathrm{Ideal}(X + Y)$, geben Sie Algorithmus für $P \in \mathrm{Ideal}(Q)$ an.

9. der arktische Halbring $\mathbb{A} = (\{-\infty\} \cup \mathbb{N}, \max, -\infty, +, 0)$.
   Ist $\deg$ homomorph (strukturerhaltend) von $(\mathbb{Z}[X], +, 0, \cdot, 1)$ nach $\mathbb{A}$?

10. Def: *Monom-Ordnung*: total, monoton bzgl. Multiplikation, terminierend.
    Implementieren und Eigenschaften überprüfen/beweisen für *graded reverse lexicographic*: Ordnen nach absteigender Exponentensumme, bei Gleichheit nach aufsteigender Exponentenfolge
    Bsp. (für $X > Y$) $X^3Y^2 > X^1Y^3$, $X^3Y^2 > X^4Y^1$.
    Welche Eigenschaft(en) gehen dabei ohne Gradierung verloren?

1. Für Monom-Ordnung $>$ bezeichnet mdeg(M) für Monome: den Exponentenvektor (in der Reihenfolge der Variablen).
   und für Polynome ($\neq 0$): den größten solchen Vektor (ist eindeutig, weil die Ordnung total ist).
   Bsp.: für $X > Y$ ist mdeg$(X^2Y^5) = [2, 5]$,
   für lex-Ordnung ist mdeg$(X^2Y^5 - X^8) = [8, 0]$.
   Ausprobieren, beweisen, ergänzen: für Polynome $f, g \neq 0$ ist mdeg$(f \cdot g) =$ mdeg$(f) +$ mdeg(g) (komponentenweise Addition)
   Welche Beziehung zw. mdeg$(f + g)$ und mdeg(f), mdeg(g)? Ggf. unter welchen Randbedingungen?

Gröbnerbasen

Begriff, Motivation

- Def: Polynom-Ideal-Element-Problem: gegeben Polynome $P, B_1, \ldots, B_n$, gefragt: $P \in \mathrm{Ideal}(\{B_1, \ldots, B_n\})$.
  Bsp: $X^5 - Y^2 \stackrel{?}{\in} \mathrm{Ideal}(X^2Y - 1, XY^2 - 1)$
- Beispiel-Anwendung: Beweis geometrischer Aussagen, die sich algebraisch formulieren lassen.
- Algorithmus von Bruno Buchberger (1965) konstruiert besondere Basis G mit $\mathrm{Ideal}(G) = \mathrm{Ideal}(\{B_1, \ldots, B_n\})$
- benannt nach seinem Doktorvater Wolfgang Gröbner, (`https://genealogy.math.ndsu.nodak.edu/id.php?id=18956`, C. F. Gauß $\to^*$ F. Klein $\to^* \ldots$)

Literatur

- B. Buchberger: Ein algorithmisches Kriterium für die Lösbarkeit eines algebraischen Gleichungssystems (1970) `https://www3.risc.jku.at/people/buchberg/refereed_publications.html`
  (in 1970-00-00-A: Verweis auf Implementierung im „Formelcode der ZUSE Z 23“)
- F. Baader und T. Nipkow: Term Rewriting and all that (Kapitel 8), Cambridge Univ. Press 1998.
- H.-G. Gräbe: Gröbnerbasen und Anwendungen (Skript, Uni Leipzig); `http://www.informatik.uni-leipzig.de/~graebe/skripte/`

Reduktion von Polynomen

- Bezeichnungen: für $p \neq 0$ (Bsp: $p = 3X^2Y - 5XZ^2$)
  $H_>(p)$, der Kopf von p: das größte (bzgl. $>$) Monom (mit Koeffz.) (Bsp: $H_>(p) = 3X^2Y$)
  $R_>(p)$, der Rest von p, so daß $p = H(p) + R(p)$
- $p \to_f q$, falls p ein Monom $a \cdot m$ enthält, das durch $H(f)$ teilbar ist (exist. m' mit $m = H(f)m'$) und $q = p - am'f$.
- für endliche Menge F von Polynomen: $(\to_F) = \bigcup_{f \in F} (\to_f)$
- Bsp. $F = \{f_1, f_2\}$ mit $f_1 = x^2y - x^2$, $f_2 = xy^2 - y^2$.
  Ordnung $>$ auf Monomen: gradiert lexikografisch
  $x^2y^2 \to_{f_1} x^2y^2 - y \cdot f_1 = x^2y \to_{f_1} x^2y - 1 \cdot f_1 = x^2$
- Satz: $(p \to_F^* q) \Rightarrow (p - q) \in \mathrm{Ideal}(F)$.
  Folgerung: $p \to_F^* 0 \Rightarrow p \in \mathrm{Ideal}(F)$.

Eigenschaften der Redukton: Termination

- $p \to_f q$, falls p ein Monom $a \cdot m$ enthält, das durch $H(f)$ teilbar ist (exist. m' mit $m = H(f)m'$) und $q = p - am'f$.
- Bsp. $x^2y^2 \to_{f_1} x^2y^2 - y \cdot f_1 = x^2y \to_{f_1} x^2y - 1 \cdot f_1 = x^2$
- Satz: $\to_F$ terminiert. (mit $(\to_F) = \bigcup_{f \in F} (\to_f)$)
- Def: Relation $\to$ terminiert, wenn es keine unendliche Kette $e_0 \to e_1 \to e_2 \to \ldots$ gibt.
- Monom-Ordnung $>$ (Monotonie, Termination)
  daraus abgeleitete Ordnung $\gg$ auf Polynomen (als absteigende Folge von Monomen, Koeffz. ignorieren)
  Satz: $>$ ist Monom-Ordnung $\Rightarrow \gg$ terminiert
  Satz: $p \to_f q \implies p \gg q$, Folgerung: $\to_F$ terminiert.

Eigenschaften d. Reduktion: Konfluenz?

- $F = \{f_1, f_2\}$ mit $f_1 = x^2y - x^2$, $f_2 = xy^2 - y^2$.
  $x^2y^2 \to_F^* x^2$ und $x^2y^2 \to_{f_2}? \to_{f_2} y^2$.
  x^2y^2 hat (wenigstens) zwei $\to_F$-Normalformen.
- Wir wollen (Existenz und) Eindeutigkeit von Nf. Dann $P \to_B^* N$ ausrechnen und $N = 0$ testen.
- Def. Relation $\to$ heißt *konfluent*, wenn
  $\forall a, b, c : (a \to^* b \wedge a \to^* c) \Rightarrow \exists d : b \to^* d \wedge c \to^* d$.
- Satz: wenn $\to$ terminiert und konfluent ist, dann besitzt jedes a genau eine Normalform
  Def: a ist $\to$-Normalform, falls $\neg \exists b : a \to b$.
- $\to_F$ ist nicht notwendig konfluent. BB-Algor.: berechnet $G \supseteq F$ mit $\mathrm{Ideal}(G) = \mathrm{Ideal}(F)$ und $\to_G$ konfluent.
  Def: solches G heißt *Gröbner-Basis* für $\mathrm{Ideal}(F)$.

S-Polynome

- Konstruktion von G für F durch Vervollständigung (Hinzufügen von Polynomen $s \in \mathrm{Ideal}(F)$)
- Def: $S(f_1, f_2) := f_1 \cdot m/H(f_1) - f_2 \cdot m/H(f_2)$.
  mit $m = \mathrm{lcm}(H(f_1), H(f_2))$ (kleinstes gemeinsames Vielf.)
  Satz: $S(f_1, f_2) \in \mathrm{Ideal}(F)$.
- Beispiel $F = \{f_1, f_2\}$ mit $f_1 = x^2y - x^2$, $f_2 = xy^2 - y^2$
  $S(f_1, f_2) = f_1y - f_2x = x^2y - xy^2$
- Satz (hier ohne Bew.) Wenn für alle $f_1, f_2 \in F : S(f_1, f_2) \to_F 0$, dann ist $\to_F$ konfluent.
- Beispiel: $x^2y - xy^2 \to_F x^2 - y^2 \not\to_F 0$.

Der Buchberger-Algorithmus

- Eing.: endl. Menge F von Polynomen, Monom-Ord. $>$
  Ausgabe: Gröbnerbasis G für $\mathrm{Ideal}(F)$.
  Start: $G_0 = F$. Dann für $i = 1, 2, \ldots$
  falls $\exists f_1, f_2 \in G_i : S(f_1, f_2) \to_{G_i}^! s_i \neq 0$, (Normalf. $\neq 0$)
  dann $G_{i+1} = G_i \cup \{s_i\}$, sonst Halt mit Resultat G_i
- Satz: Dieser Algorithmus ist total und korrekt.
  Beweis: partiell korrekt wg. $\forall i : \mathrm{Ideal}(G_i) = \mathrm{Ideal}(F)$,
  total (terminierend): falls unendl. Folge $s_0, s_1, \ldots$,
  dann existieren $i < j$ mit $H(s_i) \mid H(s_j)$ (warum?)
  das ist ein Widerspruch (zu welcher Eigenschaft?)

Buchberger-Alg., Beispiel

- (Fortsetzung, $F = \{f_1, f_2\}$ wie oben.)
- $G_0 = F$, dann $S(f_1, f_2) \to_F x^2 - y^2 = f_3$, also $G_1 = \{f_1, f_2, f_3\}$.
- Neue Paare $S(f_1, f_3) = f_1 - yf_3 = y^3 - y^2 =: f_4$ ist Normalform,
- $S(f_2, f_3) = xf_2 - y^2 f_3 = -xy^2 + y^4 \to_{f_2} y^4 - y^2 \to_{f_4} y^3 - y^2 \to_{f_4} 0$.
- ```
  load ("grobner");
  poly_grobner([x^2*y-x^2,x*y^2-y^2],[x,y]);
           2      2     2    2   2    2   3    2
         [x  y - x , x y  - y , y  - x , y  - y ]
  ```


Buchberger-Alg., Eigenschaften

- Resultat (G-Basis) und Schrittzahl hängen von der gewählten Ordnung auf Monomen ab.
- Verbesserung: nicht nur die neuen s nach G_i reduzieren, sondern auch die G_i nach s. (siehe Aufgabe)
- Die Schrittzahl kann trotzdem doppelt exponentiell sein.
- Mit manchen Ordnungen geht es „oft“ schneller,
  (empfohlen wird graded reverse lexikografisch)
  da gibt es aber nur Erfahrungswerte.
  Aufgabe: ausprobieren (wie Ordnung in maxima einstellen?)

Hausaufgaben

1. zur Ordnung auf Monomen:
   (a) unterschiedliche Ordnungen ((graded) (reverse) lexikografisch) selbst implementieren.
   `instance Ord (Mono v) where ...`
   (b) stimmt die abgeleitete Ordnung
   `newtype Mono v = M [(v, Natural)] deriving`
   mit einer dieser Ordnungen überein?
2. zur Ordnung auf Polynomen:
   für Polynome in Variablen $X > Y > Z$: geben Sie möglichst lange $\gg$-Ketten an, die bei $X^3 + Y^2 Z$ beginnen
   - bzgl. der lexikografische Ordnung auf Monomen
   - bzgl. der grad-lexikografische Ordnung auf Monomen

3. Kann $S(f_1, f_2)$ Monome mit höherem Grad enthalten als f_1 und f_2?
4. bestimmen Sie nach dem Buchberger-Algorithmus eine Gröbner-Basis B für $I = \mathrm{Ideal}(X^2Y - 1, XY^2 - 1)$.
   (auf Papier, mit maxima, mit eigener Impl.)
5. Benutzen Sie dieses B, um $X^5 - Y^2 \in I$ zu entscheiden.
   (Papier, maxima (`poly_grobner_member`), eigen)
6. Geben Sie Polynome c_1, c_2 mit $X^5 - Y^2 = c_1(X^2Y - 1) + c_2(XY^2 - 1)$ an.
   Erweitern Sie dazu den Buchberger-Alg. und das Reduktionsverfahren $\to_B$.
   (zum Vergleich: Euklid bestimmt $\gcd(a, b)$, erweiterter Euklid bestimmt c, d mit $ac + bd = \gcd(a, b)$.)

7. Eine Gröbner-Basis B heißt *reduziert*, wenn $\forall b \in B : \neg\exists c : b \to_{B\setminus\{b\}} c$.
   (b kann nicht durch die jeweils anderen Basis-Elemente reduziert werden)
   Ist die Basis auf Folie *Buchberger-Bsp.* reduziert? Wenn nein, bestimmen sie eine reduzierte B'.
   Ergänzen Sie die Eigenbau-Implementierung.
   Probieren Sie verschiedene Monom-Ordnungen.
   Die reduzierte Gröbner-Basis eines Polynom-Ideals ist im wesentlichen eindeutig.
8. Arjen Cohen: *Gröbner Bases, an Introduction* (in: *Some Tapas of Computer Algebra*, Springer, 1999):
   it is very hard to provide a good explicit upper bound on [die Laufzeit des Buchberger-Algorithmus], bad examples are known.
   Finden Sie solche *bad examples*
   (a) selbst ausdenken
   (b) mit Suchmaschine (welche Testfälle werden in Publikationen verwendet, die verbesserte Algorithmen beschreiben)
   (c) brute force
   und messen Sie den Ressourcenverbrauch (maxima, Eigenbau)
  ```

Gröberbasen in der Geometrie

Beispiel

- Quadrat $ABCD$, Punkt P auf Parallele zu BD durch C mit $BP = BD$. Schnittpunkt von CD und BP heißt Q. Zeige $DP = DQ$.
- algebraische Formulierung: Koordinaten: $A = (0,0), B = (1,0), C = (1,1), D = (0,1), P = (x,y), Q = (z,1)$.
  Lage: $CP \| BD$: $(P_x - C_y)(D_y - C_y) = (D_x - B_x)(P_y - C_y)$,
  Q auf CD: $CQ \| CD$, gleiche Länge: $|DP|^2 = |DQ|^2$.
- Quelle: Hans-Gert Gräbe: *Geometrie mit dem Computer (5. Geometrische Sätze vom Gleichungstyp)* 2018, `http://www.informatik.uni-leipzig.de/~graebe/skripte/`

Aufgaben und Anschauung Geometrie

1. Coxeter und Greitzer: *Geometry Revisited*, 1967
   Bsp. 9-Punkte-Kreis (Feuerbachkreis) Kap. 1.8
2. Intl. Math. Olympiade, Bsp: Aufgabe 2024/4
   `https://www.imo-official.org/problems.aspx`
3. dynamische Geometrie-Software Geonext (Univ. Bayreuth, Mathematikdidaktik)
   `https://web.archive.org/web/20130605045826/http://geonext.uni-bayreuth.de/`

   ```
   /opt/jdk/jdk8u442-b06/bin/java -cp /opt/geonext/geo
   ```

   Export der Zeichenvorschrift im Format `i2g`

Methode

- geometrische Lage von Punkten $[(x_1, y_1), \ldots, (x_k, y_k)]$ beschrieben durch Aussage $p = 0$ für Polynom p
- geometrische Sätze beschrieben durch Implikation
  für alle $x_1, y_1, \cdots : (p_1 = 0 \wedge \cdots \wedge p_n = 0) \Rightarrow (q = 0)$
- das folgt aus $q \in \mathrm{Ideal}(p_1, \ldots, p_n)$.
  Beweis: Idealmitgliedschaft: $q = \sum c_i p_i$,
  dann $q(x_1, y_1, \ldots) = \sum c_i p_i(x_1, y_1, \ldots) = \sum c_i \cdot 0 = 0$
- Implementierung (einfache Variante): $q \in \mathrm{Ideal}(F)$,
  gdw. $q \xrightarrow{?}_B 0$ bzgl. einer Gröberbasis B für F.
  Ü: für das Beispiel ausrechnen

Methode (Variante)

- $p_1(x_1, \ldots) = 0 \wedge \cdots \wedge p_n(x_1, \ldots) = 0 \Rightarrow q(x_1, \ldots) = 0$.
- äquivalent: es gibt kein $x = (x_1, \ldots)$
  mit $p_1(x) = 0 \wedge \cdots \wedge p_n(x) = 0 \wedge q(x) \neq 0$.
- (der Trick von Rabinovitch) äquivalent: neue Var. f und $\neg\exists x$ mit $p_1(x) = 0 \wedge \cdots \wedge p_n(x) = 0 \wedge 1 - f \cdot q(x) = 0$.
- äquivalent: $\mathrm{Ideal}(p_1, \ldots, p_n, 1 - fq) = \mathrm{Ideal}(1)$
  Gröbnerbasis dafür kann man evtl. leichter berechnen
- Vgl. Resolution in der Logik: Falls $(A_1 \wedge \ldots \wedge A_n \wedge \neg B)$ widersprüchlich, dann $(A_1 \wedge \ldots \wedge A_n \implies B)$;
  aus widersprüchlicher Formel(menge) folgt jede Formel.

Anwendungen

- diese Notation für Mupad, Ü: übertrage auf andere CAS
- Seitenhalb. schneiden sich in einem Punkt:

```
kollinear:=(x,y,z)->((z.2-y.2)*(y.1-x.1)-(y
mittelpunkt:=(a,m,b)->( (a.1-m.1)-(m.1-b.1)

groebner::gbasis([
 mittelpunkt(a,f,b), mittelpunkt(b,d,c), mi
 kollinear(a,h,d), kollinear(b,h,e),
 1 - y * kollinear(c,h,f)
])
```

- Thaleskreis:


```
qdist:=(a,b)->((a.1-b.1)^2+(a.2-b.2)^2)
gleichweit:=(a,b,c,d)-> (qdist(a,b)-qdist(c
senkrecht:=(a,b,c,d)-> ((a.1-b.1)*(c.1-d.1)

groebner::gbasis([
        senkrecht(a,c,b,c), mittelpunkt(a,m
        1 - y*gleichweit(a,m,c,m)
 ])
```

- Satz von Napoleon (kann so gar nicht gehen, da die Dreiecke „nach außen“ zu errichten sind)

```
schwerpunkt:=(a,b,c,s)->(a.1+b.1+c.1-3*s.1,
gleichseitig:=(a,b,c)->(gleichweit(a,b,b,c)
```



```
groebner::gbasis([
        a.1,a.2,b.2,
        gleichseitig(a,b,p), schwerpunkt(a,
        gleichseitig(b,c,q), schwerpunkt(b,
        gleichseitig(c,a,r), schwerpunkt(c,
        1 - y * gleichseitig(u,v,w)
])
```

- im Projekt *Symbolic Data* haben H.-G. Gräbe et al. solche Beispiele erzeugt und gesammelt
  `https://web.archive.org/web/20120106071657/http://www.symbolicdata.org:80/`
  siehe auch `https://symbolicdata.github.io/`, `https://github.com/symbolicdata/data/tree/master/XMLResources/GeoProofSchemes`

Hausaufgaben

1. Beispiele für geometrische Theoreme aus Skript Gräbe 2018 (Simsonsche Gerade, Miquelscher Punkt, ...)
   - mit Geogebra zeichnen, Theorem überprüfen (durch Betrachtung der Lage bei Verschieben von Punkten)
   - algebraisieren (automatisieren!)
   - nach beschriebenen Verfahren lösen (Gröberbasis mit verschiedenen CAS ausrechnen)
2. ein falsches Theorem beweisen (um zu sehen, wie die Methode die Falsch-Aussage erkennt)
3. noch offene Aufgaben aus vorigen Wochen, insbesondere zur Eigenimplementierung von Polynomen (Division!) und Buchberger-Algorithmus

4. *bad examples* (für Buchberger-Algorithmus) in verschiedenen CAS (viele naive Algebraisierungen von geometrischen Sätzen ergeben aufwendige Rechnungen, aber es gibt auch kleinere schwere Testfälle)

   Beispiel (Verallgemeinerung? Quelle?)

```
cyclic4 =
  let [a,b,c,d] = map (var . V) [1 .. 4]
  in  [ a*b*c*d-1
      , a*b*c+b*c*d+c*d*a+d*a*b
      , a*b+b*c+c*d+d*a
      , a+b+c+d
      ]
```


5. den Rabinovitch-Trick anwenden für den Satz über die Simson-Gerade in allgemeiner Lage (nur ein Punkt ist fixiert)

```
fsimson = prove $ do
  ...
  return $ colinear a' b' c'
```

   die letzte Zeile ersetzen durch

```
  z0 <- number
  assert $ ...
  return $ ...
```

   und Laufzeiten (der naiven Implementierung sowie anderer CAS) vergleichen

6. die Behandlung von Degenerations-Bedingungen, Beispiel: der Satz von Pappus

```
pappus = prove $ do
  a  <- point ; b  <- point; c  <- point_on a  b
  a' <- point ; b' <- point; c' <- point_on a' b'
  p <- intersection b c' b' c
  q <- intersection a c' a' c
  r <- intersection a b' a' b
  return $ -- degenerate cases:
           parallel (b-c') (b'-c)
         * parallel (a-c') (a'-c)
         * parallel (a-b') (a'-b)
           -- actual conclusion:
         * colinear p q r
```


7. bessere Implementierungen des Buchberger-Verfahrens: nicht jedesmal *alle* S-Polynome bestimmen und bezüglich *aller* bisherigen reduzieren.

   Zum Code in `poly/Grobner.hs`:

   - die Invarianten beweisen
   - die Termination begründen
     - für jede einzelne Reduktion eines (S-)Polynoms
     - für das gesamte Buchberger-Verfahren

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