Symbolisches Rechnen
Vorlesung
Wintersemester 2006

Johannes Waldmann, HTWK Leipzig

30. Januar 2007

Einleitung

Überblick

- Rechnen mit Zahlen (beliebig groß, beliebig genau)
- Rechnen mit Polynomen und Funktions-Ausdrücken (klassische Computeralgebra, z. B. Differentiation, Summation, Integration)
- Rechnen mit Figuren (geometrische Konstruktionen und Beweise)
- Rechnen mit Programmen (Programmtransformationen, Refactoring)
- Rechnen mit logischen Formeln (automatische Beweiser und Beweis-Überprüfer)

Literatur

- Wolfram Koepf: Computeralgebra, Springer, 2006. `http://www.mathematik.uni-kassel.de/~koepf/CA/`
- Christopher Creutzig, Walter Oevel, Jürgen Gerhard: Das MuPAD Tutorium, Springer, 2004
  `http://www.amazon.de/exec/obidos/ASIN/3540435735/wurev`

  Hinweis: diese Buch kann deutlich verbilligt erworben werden (bei Sammelbestellung über die Uni Leipzig)
- Hans-Gert Gräbe: Einführung in das Symbolische Rechnen, Skript, Universität Leipzig
  `http://www.informatik.uni-leipzig.de/`

`~graebe/skripte/esr06.pdf`

Computer-Algebra-Systeme

(kostenpflichtig) Mupad `http://www.sciface.com/`

- ist im Pool (Z423) installiert.
- wegen Home-Use-Lizenzen siehe
  `http://www.informatik.uni-leipzig.de/~graebe/ComputerAlgebra/mupad-lizenz.html`
- wer die originale Suse-Linux-CD 9.1 besitzt, darf das dort enthaltene mupad-2.5.2 benutzen.

(frei)

- Axiom
  `http://wiki.axiom-developer.org/FrontPage`
  ist im Pool (Z423) installiert.

- (wx)Maxima `http://maxima.sourceforge.net/` `http://wxmaxima.sourceforge.net/`
- Geonext `http://geonext.uni-bayreuth.de/`
- Coq/Coqide `http://coq.inria.fr/` ist im Pool (Z423) installiert

Zahlen

Themen

- addieren
- multiplizieren (Karatsuba)
- potenzieren („russisch“)
- (erweiterter) Euklid (gcd)
- Primzahltests, Faktorisierung

Quadratwurzeln

Wie bestimmt man schnell viele Ziffern von $\sqrt{2}$?
Idee: Potenzreihe (Satz von Taylor)

$$f(x_0+d) \approx \sum_{n\geq 0} \frac{f^{(n)}(x_0)}{n!} d^n$$

$$\sqrt{(}1+d) = 1 + d/2 - d^2/8 + 5d^3/2^4 + 5d^4/2^7 - \dots$$

konvergiert aber zu langsam für $d=1$
Trick: $5^2 * 2 \approx 7^2$, also
$\sqrt{2} = 7/5 \cdot \sqrt{(}1+1/49)$
(Aufgabe: für $\sqrt{3}$)
(wo kommen diese Brüche her? Kettenbrüche!)

Logarithmen

Taylor-Entwicklung

$$\log(1+x) = x - x^2/2 + x^3/3 - \dots$$

konvergiert viel zu langsam für $x = 1$ und gar nicht für größere x.

Trick (alt, J.R. Young, 1835, zitiert in: v. Mangoldt/Knopp, Bd. 2, S. 127)
betrachte $\log\frac{16}{15}, \log\frac{25}{24}, \log\frac{81}{80}$

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

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)
- $\pi/4 = \arctan x + \arctan y$

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

Kettenbrüche

- Ein Bruch der Form

  $$a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \dots}}$$

  mit ganzen Zahlen a_0 beliebig, $a_1 \geq 1, a_2 \geq 1, \dots$ heißt Kettenbruch.

- Bsp: endlicher Kettenbruch

  $$3 + \frac{1}{7 + \frac{1}{16}} = 3 + \frac{1}{113/16} = 3 + \frac{16}{113} = \frac{355}{113}$$

- Bsp: unendlicher periodischer Kettenbruch (Wert ist Limes der Folge der Partialbrüche)

  $$x = 1 + \frac{1}{2 + \frac{1}{2 + \frac{1}{2 + \dots}}}$$

  Welche Zahl x ist das?

  Welches sind die Näherungsbrüche?

Gebrochen rationale Funktionen

- haben allgemeine Form $f(x) = \dfrac{ax+b}{cx+d}$
- Notation als Matrix $[f] = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$
- ist nützlich, denn es gilt $[f \circ g] = [f] \cdot [g]$

betrachte Kettenbruch $k = [a_0; a_1, \dots, a_n]$
als Funktion $x \mapsto [a_0; a_1, \dots, a_n, x]$
$[k] =$
$$\begin{pmatrix} 1 & a_0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & a_1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \cdots \begin{pmatrix} 1 & a_n \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$
$$= \begin{pmatrix} a_0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} a_1 & 1 \\ 1 & 0 \end{pmatrix} \cdots \begin{pmatrix} a_n & 1 \\ 1 & 0 \end{pmatrix}$$

. . . das kann man auch mit mupad ausrechnen:

```
f := (a -> matrix([[a,1],[1,0]]))
f(1) * f(2) * f(2) * f(2)
```

Wert eines periodischen Kettenbruchs ergibt sich als Fixpunkt der gebr. rationalen Funktion, die der Periode entspricht.
Periode 2: $[f(2)](x) = x$

```
linalg::eigenvalues (f(2))
```

Kettenbrüche (II)

- jede reelle Zahl besitzt (im wesentlichen) genau einen Kettenbruch-Entwicklung
- für jede rationale Zahl ist der Kettenbruch endlich
- „im wesentlichen": falls endlich, dann letzte Stelle > 1.
- jede quadratische Irrationalität (nicht rationale Nullstelle eines Polynoms zweiten Grades) besitzt einen periodischen Kettenbruch

(Periode exakt ausrechnen? Algebraische Zahlen!)

Aufgabe: (mupad: contfrac) für π und/oder e oder $\exp(2), \exp(1/2), \dots$ Gesetzmäßigkeiten raten . . . und beweisen! — Vgl. HAKMEM `http://www.inwap.com/pdp10/hbaker/hakmem/cf.htm`

Test-Testat 2. 11.

1. Bestimmen Sie die Zahl $y = \tan\beta$ mit $\tan(4\alpha+\beta) = 1$ für $x = \tan\alpha = 1/5$.

2. Für jede Zahl $n \geq 2$:

   Die Matrix A der Form $n \times n$ enthält auf der Hauptdiagonale immer die 2 und auf den beiden ersten Nebendiagonalen immer die -1.

   Das erhält man in Mupad einfach so:

   ```
   matrix(n,n,[-1,2,-1],Banded)
   ```

   und ausführlich so:

   ```
   A := n -> matrix (n,n, (i,j) ->
   ```



```
    if i = j then 2
    else if i = j-1 or i = j + 1 then -1 e
    end_if  )
```

Beschreiben Sie die Inverse B von A durch ein ähnliche Programm! D. h. `A(n)*B(n)` soll immer die Einheitsmatrix ergeben.

3. Beweisen Sie, daß alle Eigenwerte von A positiv sind. (Quelle: Berkeley Problems in Mathematics, Spring 199 Problem 12,

   `http://math.berkeley.edu/~desouza/pb.html`

   Bemerkung 1: Die Matrix ist symmetrisch, also sind die Eigenwerte reell.

   Bemerkung 2: mit `linalg::eigenvalues` sehen Sie,

daß Mupad das nicht immer merkt. Wir rechnen aber anders:

Wenn der Spaltenvektor x^T ein Eigenvektor von A zum Eigenwert λ ist, dann gilt nach Definition $Ax^T = \lambda x^T$. Betrachten Sie $x \cdot A \cdot x^T$.

Einen geeigneten Vektor erhalten Sie mit

```
X := n -> matrix([[x.i $ i = 1 .. n ]])
```

4. Berechnen Sie mit `numeric::eigenvalues` einige Werte. Für welche n sind die Zahlen $1, 2, 3, 4$ Eigenwerte? Können Sie das beweisen?

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(
}
```

(wieviel weiter darf die j-Schleife noch gehen?)

Multiplikation (Zahlen, Matrizen)

Karatsuba-Multiplikation

$(10^n a + b)(10^n c + d) = 10^{2n} ab + 10^n (ad + bc) + bd$

Idee: Anzahl der Multiplikationen verringern

(auf Kosten der Additionen).

Berechne ac, bd und $(a+b)(c+d) = ac + (ad+bd) + bd$,

ergibt $(ad+bc) = (a+b)(c+d) - ac - bd$.

Statt 4 Mult und 3 Add: 3 Mult. und 7 Add/Sub.

Rechenzeit: $T(n) = 3T(n/2) + c \cdot n$. Ansatz $T(n) = n^x$.

`http://mathworld.wolfram.com/KaratsubaMultiplication.html`

Matrix-Multiplikation nach Strassen

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{pmatrix}$$

8 Mult, 4 Add.

Es genügen 7 Multiplikationen (welche?) und mehrere Additionen/Subtraktionen (welche?)

Dann $T(n) = 7 * T(n/2) + c \cdot n$

`http://mathworld.wolfram.com/StrassenFormulas.html`

Multiplikation Boolescher Matrizen

(im Booleschen Halbring: plus = or, mal = und)

Für $A \cdot B$, beide quadratisch $n \times n$,

zerlege A in vertikale Streifen, B in horizontale, gleicher Breite l, dann $AB = \sum A_i B_i$.

Jeder Streifen A_i besteht aus Zeilen $A_{i,j}$ (der Breite l)

Jedes Produkt $A_{i,j} \cdot B_i$ ist eine Linearkombination der Zeilen von B_i.

Trick: alle 2^l Kombinationen vorher berechnen und speichern! benötigt $2^l n$ Platz (und Zeit)

$A \cdot B$ benötigt dann noch $(n/l)n^2$ Zeit. Geeignetes l?

Der „Vier-Russen-Algorithmus“

`http://www14.in.tum.de/skripten/ead_ws9899_html/node94.html`

Multiplikationsketten

Bei einem Matrixprodukt mit mehreren Faktoren kann ma durch geeignete Klammerung Zeit sparen.

$A_{3\times 2} B_{2\times 8} C_{8\times 4} D_{4\times 1}$

Annahme: Zeit für $A_{x\times y} B_{y\times z}$ ist $\approx xyz$.

Was kostet (z. B.) $(AB)(CD)$?

geeignete Klammerung bestimmt man durch dynamische Optimierung:

$c(i,j) :=$ beste Kosten für $A_i \cdot \dots \cdot A_j$

A_i hat Abmessung $d_{i-1} \times d_i$

$c(i,i) = 0$

$c(i,k) = \min\{c(i,j) + c(j+1,k) + d_{i-1} d_j d_k \mid i \leq j < k\}$

Terme, Ersetzungs-Systeme

Terme, Positionen

```
>> prog::exprtree((2*k+1)!!)
                        fact2
                        |
                        `-- _plus
                            |
                            +-- _mult
                            |   |
                            |   +-- k
                            |   |
                            |   `-- 2
                            |
                            `-- 1
>> op((2*k+1)!!,[1,1,2])          2
```


Folgen

werden durch Komma getrennt, bei Bedarf rund geklammert (nicht eckig!)

```
>> _plus (x,y,z)
                              x + y + z

>> _plus (k $ k=1..100)
                                5050

>> float (_plus ( 1/(i+j) $ j=1..i $ i=1..
```

(welcher Grenzwert? Warum?)

Term-Umformungen

Formel = *Term*,

Operation (Differenzieren, Vereinfachen) = Termumformung, gegeben durch *Regelsystem*.

Regel ist Paar von Termen mit Variablen: $(f(x), g(x,x))$,

Regelanwendung: wenn linke Seite *paßt*, dann *ersetze* durch entsprechende rechte Seite.

Beispiel: $D(\sin(x), x) \to \cos(x)$.

eigentlich: $D(x,x) \to 1$, $D(\sin(t), x) \to \cos(t) * D(t,x)$.

Variablen, Substitutionen

- Terme und Variablen: $\mathrm{Term}(\Sigma, V)$: Menge der Terme mi Symbolen aus Σ und Variablen aus V.
- Vereinbarung: Variablen am Ende des Alphabets $(x, y, \ldots)$, Funktionssymbole am Anfang $(f, g, a, b, \ldots)$
- Menge der in einem Term t vorkommende Variablen: $\mathrm{Var}(t)$
- Substitution σ ist Abbildung $V \to \mathrm{Term}(\Sigma, W)$
- eine Substitution σ angewendet auf einen Term t erzeu Term $t\sigma$

  Beispiel $t = f(x)$, $\mathrm{Var}(t) = \{x\}$, $\sigma : x \mapsto g(2,2)$, $t\sigma = f(g(2,2))$.

Positionen, Teilterme (Wiederholung)

- Term t, Position $p \in \mathrm{Pos}(t)$.
- $t(p)$ Symbol an Position p in t
- $t\mid_p$ Teilterm an Position p in t
- $t[p := s]$ in t an Position p den Term s einsetzen

Beispiele:

- Term $t = h(1, f(f(2)))$,
- Positionen $\mathrm{Pos}(t) = \{\epsilon, 1, 2, 21, 211\}$,
- Symbol $t(2) = f$,
- Teilterm $t|_2 = f(f(2))$,
- Teilterm ersetzen $t[2 := a] = h(1, a)$.

Regeln

- Regel (l, r), Schreibweise $(l \to r)$, mit $l, r \in \mathrm{Term}(\Sigma, V)$
- Regel $(l \to r)$ an Position p in $t \in \mathrm{Term}(\Sigma)$ anwenden, um s zu erhalten, Schreibweise $t \to_{(l\to r),p} s$

  $\exists \sigma : V \to \mathrm{Term}(\Sigma) : t|_p = l\sigma \wedge s = t[p := r\sigma]$

Beispiel: Regel $(l, r) = (f(x), g(x,x))$, Term $t = h(1, f(f(2)))$.
Position $p = [2] \in \mathrm{Pos}(t)$, Teilterm $t|_p = f(f(2))$,
Substitution $\sigma : x \mapsto f(2)$ mit $t|_p = l\sigma$,
auf r anwenden: $r\sigma = g(f(2), f(2))$,
in t einsetzen:
$s = t[p := r\sigma] = h(1, r\sigma) = h(1, g(f(2), f(2)))$.

Regelsysteme

- R eine Menge von Regeln, definiert Relation (einmalige Anwendung irgendeiner Regel irgendwo):

  $\to_R := \{(t, s) \mid \exists (l, r) \in R, p \in \mathrm{Pos}(t) : t \to_{(l,r),p} s\}$
- transitive Hülle $\to_R^+$ (mehrmalige Regelanwendung)

liefert nichtdeterministisches Berechnungsmodell (vgl. Grammatiken/Wortersetzungssysteme)
Fragen:

- jede Rechnung endet? (Termination) nach welcher Zeit?
- jede Rechnung endet mit genau einem Resultat?

dabei: Resultat = Term, auf den keine Regel anwendbar ist = Normalform.

Beispiele

Signatur (mit Stelligkeiten): $\Sigma = \{e^0, s^1, p^2, m^2, h^2\}$

$$R = \{p(e, y) \to y, p(s(x), y) \to s(p(x, y))\}$$

Ableitungen von $p(p(e, s(e)), s(e))$?

Wort- und Term-Ersetzung

Man kann aus jedem Wort (= Folge von Buchstaben) einen Term konstruieren,
Beispiel: $abaab \approx a(b(a(a(b(\epsilon)))))$.

Damit ist sind Wortersetzungssysteme ein Spezialfall von Termersetzungssystemen.
Beispiel:
$aabb \to bbbaaa \approx a(a(b(b(x)))) \to b(b(b(a(a(a(x))))))$

Alle Aussagen über TES gelten auch für WES.

einige Aussagen über WES sind spezieller, da es für Wörter eine assoziative Verknüpfung gibt, aber für Terme nicht.

Termination und Normalisierung

Für eine zweistellige Relation ρ auf M
(zum Beispiel eine Ersetzungsrelation $\to_R$ auf $\mathrm{Term}(\Sigma)$)

- ρ ist *terminierend (stark normalisierend)*, falls es keine unendliche lange ρ-Folge gibt

  d. h. $\rho(x_0, x_1), \rho(x_1, x_2), \rho(x_2, x_3), \dots$

- ρ is *schwach* normalisierend, falls zu jedem $x_0 \in M$ eine ρ-Folge zu einer ρ-Normalform führt

Beachte: es gibt ρ, die schwach, aber nicht stark normalisieren.
Beispiel (Wortersetzung): $fgfg \to gfgffg$ (Alfons Geser, 2000)

Aufgaben

Signatur (mit Stelligkeiten): $\Sigma = \{e^0, s^1, p^2, m^2, h^2\}$

- $R_p = \{p(e, y) \to y, p(s(x), y) \to s(p(x, y))\}$
- $R_m = R_p \cup \{m(e, y) \to e, m(s(x), y) \to p(m(x, y), y)\}$
- $R_h = R_m \cup \{h(x, e) \to s(e), h(x, s(y)) \to m(h(x, y), x)\}$

Fragen (leicht):

- Menge der R_h-Normalformen?
- Normalformen (Existenz? Eindeutigkeit?) von:
  $p(s(p(s(e), s(e))), s(e))$, $m(s(s(e)), s(s(e)))$,
  $h(s(s(e)), s(s(s(e))))$.

(schwerer) $\Sigma = \{a^2, d^0\}, R = \{a(a(d, x), y) \to a(x, a(x, y))\}$:
Normalformen? Ableitungslängen?

Normalformen

Für eine zweistellige Relation ρ auf M
(zum Beispiel eine Ersetzungsrelation $\to_R$ auf $\mathrm{Term}(\Sigma)$)
heißt $x \in M$ eine ρ-Normalform, falls $\neg\exists y \in M : \rho(x, y)$.

Beachte: für passende ρ und x kann vorkommen:

- p hat mehrere Normalformen
- p hat keine Normalform

Termersetzung/Anwendungen

Termersetzung ist

- Turing-vollständiges Berechnungsmodell
- Grundlage für funktionale Programmierung
- Grundlage für XML-Transformationen mit XSLT

Für Anwendungen wichtig sind

- Termination (keine unendlich langen Rechnungen)
- Konfluenz (eindeutige Ergebnisse von Rechnungen)

Term-Ersetzung und Computeralgebra

Regeln (Term-Ersetzungs-System) für das Differenzieren

$$D_x(x) = 1, \text{ wenn } x \notin A : D_x(A) = 0$$
$$D_x(A + B) = D_x(A) + D_x(B)$$
$$D_x(A \cdot B) = \dots, D_x(A/B) = \dots$$
$$D_x(\log x) = 1/x, D_x(\sin x) = \cos x, \dots$$

Dazu braucht man aber noch Vereinfachungsregeln.
Wie drückt man die Kettenregel $D_x(f(g(x)) = \dots$ aus?
Hier sind f und g Variablen, aber zweiter Ordnung (bezeichnen Funktionen).
Regeln für das Integrieren?
Existenz und Eindeutigkeit von Normalformen?

Konfluenz

Eine zweistellige Relation ρ heißt *konfluent*, wenn

$$\forall x, y_1, y_2 : \rho^*(x, y_1) \wedge \rho^*(x, y_2) \Rightarrow \exists z : \rho^*(y_1, z) \wedge \rho^*(y_2, z)$$

(Bild ist einfacher zu merken)

Satz: wenn ρ auf M konfluent ist, dann besitzt jedes $x \in M$ höchstens eine ρ-Normalform.

Beachte: es wird nicht behauptet, daß x *überhaupt eine* Normalform besitzt.

Falls ρ jedoch terminiert, dann läßt sich Konfluenz charakterisieren und entscheiden durch einen Hilfsbegriff (lokale Konfluenz, später)

Lokale Konfluenz

Eine zweistellige Relation ρ heißt *lokal konfluent*, wenn

$$\forall x, y_1, y_2 : \rho(x, y_1) \wedge \rho(x, y_2) \Rightarrow \exists z : \rho^*(y_1, z) \wedge \rho^*(y_2, z)$$

Beachte: es gibt Relationen ρ, die lokal konfluent sind, ab nicht konfluent. Satz: wenn ρ terminiert und lokal konfluen ist, dann ist ρ konfluent.

Kritische Paare

Für ein Termersetzungssystem R über Σ: falls

- $(l_1 \to r_1)$ und $(l_2 \to r_2)$ sind Regeln in R ohne gemeinsame Variable (ggf. vorher umbenennen!)
- es gibt $p \in \mathrm{Pos}(l_1)$ so daß $l_1[p]$ und l_2 *unifizierbar* sind

  d. h., es existiert *mgu* σ mit $l_1[p]\sigma = l_2\sigma$
- falls $(l_1 \to r_1) = (l_2 \to r_2)$ (vor Umbenennung), dann $p \neq \epsilon$
- dann heißt $(r_1\sigma, (l_1\sigma)[p := r_2\sigma])$ *kritisches Paar* von R.

Ein kritisches Paar (s, t) heißt *zusammenführbar*, falls $\exists u : s \to_R^* u \wedge t \to_R^* u$.
Satz: ein Termersetzungssystem ist genau dann lokal konfluent, wenn alle kritische Paare zusammenführbar sind.

Unifikation

- ein *Unifikator* von zwei Termen $s, t \in \mathrm{Term}(\Sigma, V)$

  ist eine Substitution $\sigma : V \to \mathrm{Term}(\Sigma, V)$ mit $t\sigma = s\sigma$
- zwei Terme s, t können keinen, einen oder mehrere Unifikatoren haben
- Substitutionen kann man ordnen durch $\sigma_1 \leq \sigma_2$ (σ_1 ist allgemeiner als σ_2) falls $\exists \tau : \sigma_2 = \sigma_1 \circ \tau$
- Satz: Wenn s, t unifizierbar sind, dann gibt es einen allgemeinsten Unifikator (most general unifier, mgu) σ von s und t:

  für jeden Unifikator σ_2 von s und t gilt $\sigma \leq \sigma_2$.

Bestimmung des mgu

(vergleiche Logische Programmierung)

- falls $s = t$, dann return identische Abbildung
- falls s Variable
  - falls s in t vorkommt, dann fail
  - sonst return $(s \mapsto t)$
- falls t Variable entsprechend
- falls $s = f(x_1, \ldots)$ und $t = g(y_1, \ldots)$, dann
  - falls $f \neq g$, dann fail, sonst …
  - bestimme $\sigma = \mathrm{mgu}(x_1, y_1)$
  - return $\sigma \cup \mathrm{mgu}((x_2\sigma, \ldots), (y_2\sigma, \ldots))$

Dieser Algorithmus ist korrekt, aber nicht effizient.

Orthogonale Systeme

- ein TES R über Σ heißt *nichtüberlappend*, wenn R kein kritischen Paare besitzt.
- ein TES R heißt linkslinear, wenn in keiner linken Regelseite eine Variable mehrfach vorkommt.
- ein TES R heißt *orthogonal*, falls es linkslinear und nichtüberlappend ist.

Satz: jedes orthogonale System ist konfluent.

Funktionale Programmierung $\approx$ System R ist orthogonal und … (nächste Folie)

Konstruktor-Systeme

- Für TES R über Σ: ein Symbol $f \in \Sigma$ heißt *definiert*, wenn es als Wurzel einer linken Regelseite in R vorkommt.
- Alle anderen Symbole heißten *Konstruktoren*.
- R heißt Konstruktor-System, falls in jeder linken Seite nur ein definiertes Symbol vorkommt (und zwar in der Wurzel).

Beispiele: $\{P(E, y) \to \ldots, P(S(x), y) \to \ldots, M(E, y) \to \ldots, M(S(x), y) \to \ldots\}$ ist Konstruktor-System, definierte Symbole sind $\{P, M\}$, Konstruktoren sind $\{S, E\}$
aber $\{A(A(D, x), y) \to A(x, A(x, y))\}$ nicht.

XSLT

- (extensible style sheet language transformations)
- ist eine (seltsame) Art, ein Term-Ersetzungssystem für XML-Dokumente hinzuschreiben.
- besteht aus Ersetzungsregeln (Templates) und Ersetzungsstrategie
- Regeln haben Form $f(x_1, \ldots, x_n) \to \ldots$ für Elemente f evtl. mit zusätzlichen Bedingungen
- ungefähr outermost rewriting (bei der Wurzel beginnend behandelt aber jeden Knoten (normalerweise) höchster einmal.

Ersetzungssysteme und Programmanalyse

- R ein Ersetzungssystem, L eine Sprache (Menge) von Bäumen,
  $$R^*(L) = \{t \mid \exists s \in L : s \to_R^* t\}$$
  alle Mehr-Schritt-Nachfolger von L.
- modelliert: R ein Programm, L eine Menge von (möglichen) Eingaben, dann $R^*(L)$ Menge von erreichbaren Zuständen (Zwischenergebnissen)
- E eine Menge von verbotenen Zuständen. Kein verbotener Zustand erreichbar: $R^*(L) \cap E = \emptyset$
- Beispiel: durch XSLT-Transformation soll immer Schema-konformes Dokument entstehen

Ersetzung und Automaten (Regularität)

- Normalformen
  - Normalformen für $\{fgfg \to \ldots\}$?
  - Normalformen für $\{A(A(D, x), y) \to \ldots\}$?
  - Ist Menge der Normalformen immer regulär? (Nein.)
- Nachfolgermengen
  - L und E durch endliche (Baum-)Automaten beschrieben. Wie kann man $R^*(L) \cap E = \emptyset$ entscheiden?
  - im Allgemeinen gar nicht, wg. Halteproblem
  - in Einzelfällen doch
    $\Sigma = \{a, b\}, R = \{aa \to aba\}, L = a^*, E = \Sigma^* bb \Sigma^*$

Lambda-Kalkül

Motivation

Term-Ersetzung drückt nicht alle Aspekte der üblichen mathematischen/logischen Sprache aus:
es fehlen Variablen (die sind zwar in den Regeln, aber nicht in den Termen, auf die diese angewendet werden)

gebundene Variablen in der …

- Analysis: $\int x^2 \,\mathrm{d}\, x, \sum_{k=0}^{n} k^2$
- Logik: $\forall x \in A : \forall y \in B : P(x, y)$
- Programmierung:
  `static int foo (int x) { ... }`

Der Lambda-Kalkül

(Alonzo Church, 1936 … Henk Barendregt, 1984 …)
ist der Kalkül für Funktionen mit benannten Variablen
(und sonst nichts—keine Zahlen usw., die einzigen Daten sind Funktionen!)
die wesentliche Operation ist das Anwenden einer Funktion:

$$(\lambda x.B)A \to B[x := A]$$

Beispiel: $(\lambda x.x * x)(3 + 2) \to (3 + 2) * (3 + 2)$

Lambda-Terme

Menge Λ der Lambda-Terme (mit Variablen aus einer Menge V):

- (Variable) wenn $x \in V$, dann $x \in \Lambda$
- (Applikation) wenn $F \in \Lambda, A \in \Lambda$, dann $(FA) \in \Lambda$
- (Abstraktion) wenn $x \in V, B \in \Lambda$, dann $(\lambda x.B) \in \Lambda$

das sind also Lambda-Terme:
$x, (\lambda x.x), ((xz)(yz)), (\lambda x.(\lambda y.(\lambda z.((xz)(yz)))))$

verkürzte Notation

- Applikation als links-assoziativ auffassen, Klammern weglassen:

  $$(\ldots((FA_1)A_2)\ldots A_n) \sim FA_1A_2\ldots A_n$$

  Beispiel: $((xz)(yz)) \sim xz(yz)$
- geschachtelte Abstraktionen unter ein Lambda schreibe

  $$\lambda x_1.(\lambda x_2.\ldots(\lambda x_n.B)\ldots) \sim \lambda x_1 x_2 \ldots x_n.B$$

  Beispiel: $\lambda x.\lambda y.\lambda z.B \sim \lambda xyz.B$

Mehrstellige Funktionen

die vorigen Abkürzungen sind sinnvoll, denn
$(\lambda x_1 \ldots x_n.B)A_1 \ldots A_n$
verhält sich wie eine Anwendung einer mehrstelligen Funktion.
um die zu beschreiben, genügt also ein Kalkül für einstellige Funktionen.
(Beispiel)

Ableitungen (Ansatz)

Absicht: Relation $\to$ auf Λ (Ein-Schritt-Ersetzung):

- $(\lambda x.B)A \to B[x := A]$ (Vorsicht)
- $F \to F' \Rightarrow (FA) \to (F'A)$
- $A \to A' \Rightarrow (FA) \to (FA')$
- $B \to B' \Rightarrow \lambda x.B \to \lambda x.B'$

was soll $(\lambda x.B)[x := 3 + 4]$ bedeuten?
ist das sinnvoll:
$(\lambda x.(\lambda y.xyx))(yy) \to (\lambda y.yx)[x := (yy)] = \lambda y.y(yy)$
das freie y wird fälschlich gebunden

Das falsche Binden von Variablen

(voriges Beispiel in C++):
Diese Programme sind *nicht* äquivalent:

```
int f (int x) {
  int y = x + 3; int sum = 0;
  for (int x = 0; x<4; x++) { sum = sum + y
  return sum;
}
int g (int x) {
                 int sum = 0;
  for (int x = 0; x<4; x++) { sum = sum + (x
  return sum;
}
```

Gebundene Umbenennungen

wir dürfen $(\lambda x.B)A \to B[x := A]$ nur ausführen, wenn x nicht in A frei vorkommt.
falls doch, müssen wir $\lambda x.B$ in $\lambda y.B[x := y]$ umbenennen
wobei y weder in A frei noch in B überhaupt vorkommt.

(Beispiel) (Def. $\mathrm{FV}(t)$)
eine solche gebundene Umbenennung in einem Teilterm heißt α-Konversion.
α-konvertierbare Terme sind äquivalent (verhalten sich gleich bzgl. Ableitungen)
(Beispiel)
mit o.g. Bedingung ergibt sich eine vernünftige Relation –
(β-Reduktion).

(Beispiel-Ableitungen)

Eigenschaften der Reduktion

$\to$ auf Λ ist

- konfluent,
- aber nicht terminierend.

  $W = \lambda x.xx, \Omega = WW.$

- es gibt Terme mit Normalform und unendlichen Ableitungen, $KI\Omega$ mit $K = \lambda xy.x, I = \lambda x.x$

Einige Eigenschaften klingen erstaunlich: z. B. jeder Term F besitzt einen Fixpunkt A, d. h. $FA \to^* A$.
Den kann man sogar ausrechnen: es gibt R mit $F(RF) \to^* RF$.

Rechnen mit simulierten Zahlen

- Wahrheitswerte $T = \lambda xy.x, F = \lambda xy.y$

  denn $BMN \to M$ oder $BMN \to N$ für $B \in \{T, F\}$

- Zahlen:

  $[0] = I, [n+1] = \lambda x.xF[n]$

- Dafür sind Nachfolger, Vorgänger, Null-Test definierbar.

mit Fixpunktsatz gibt es auch Rekursion (beliebige Schleifen), also ist jede Turing-berechenbare Funktion auch Lambda-berechenbar (und umgekehrt).
Übung: Addition, Multiplikation, Potenzieren
(tatsächlich ist das Modell älter als die Turing-Maschine)

Erweiterungen, Anwendungen

ausgehend vom einfachen Lambda-Kalkül baut man:

- Typsysteme (jeder Ausdruck, jede Variable besitzt Typ)
- eingebaute Datentypen (außer Funktionen) und Operationen, z. B. Zahlen
- effiziente Implementierung von Reduktionen (ohne die umständlichen Umbenennungen)

das bildet die Grundlage für

- exakte Analyse von Programmier/mathematischen/logischen Sprachen
- Implementierung von Sprachen und Refactoring-Werkzeugen

Lambda-Kalkül und Computeralgebra

- Kalkül beschreibt Rechnen mit Ausdrücken mit gebundenen Variablen, diese kommen in CAS vor.
- die Erkenntisse aus dem Kalkül werden in verschiedenen CAS mit verschiedener Konsquenz angewendet (leider).
- Probleme beginnen damit, daß Variablenbindungen schon gar nicht korrekt notiert werden
- ... das ist nur ein getreues Abbild entsprechender Probleme in der Mathematik (solche Fehler heißen dort aber Konventionen)

Vereinfachung (Simplifikation)

Einleitung, Motivation

Literatur: H.-G. Gräbe, Skript Computeralgebra, Kapitel 3

$$\sqrt{2+\sqrt{3}} - \sqrt{2-\sqrt{3}}$$

$$\frac{b^2}{(a-b)\ (b-c)} - \frac{a^2}{(a-b)\ (a-c)} - \frac{c^2}{(a-c)\ (b-c)}$$

Vereinfacher (Simplifikator) $S : \mathrm{Term} \to \mathrm{Term}$
überall definiert, terminierend, idempotent, äquivalent, null-erkennend? kanonisch?

Simplifikatoren für ...

- Polynome
  - distributive Darstellung
  - rekursive Darstellung
- rationale Funktionen
- trigonometrische Funktionen
- Logarithmus- und Exponentialfunktion

Polynome in einer Variablen

$p(x) = \sum a_i x^i$

- dicht: alle Koeffizienten speichern
- dünn: nur die $a_i \neq 0$
- ... nach Exponenten geordnet

Polynome in mehreren Variablen

$p(x, y) = 3xyz + x^3 - y^2z^4$

- rekursive Darstellung

  (nach Wahl einer Ordnung auf Variablen)

- distributive Darstellung

  (nach Wahl einer kanonischen Form und Ordnung auf Monomen = Produkten von Variablenpotenzen)

  wünschenswert ist Monotonie der Ordnung bzgl. Multiplikation

Rationale Funktionen

= (Summe von) Brüchen aus Polynomen in mehreren Variablen
Simplifikation: Hauptnenner bilden, Zähler und Nenner einzeln vereinfachen
Eigenschaften?
Gemeinsame Teiler zu entfernen ist aufwendig, aber möglich (gcd von Polynomen)

Verallgemeinerte Kerne

Rationale Ausdrücke nicht in Variablen, sondern in Ausdrücken mit Wurzel-, Winkel- oder Logarithmusfunktionen in der Wurzel.

Trigonometrische Ausdrücke

- Summe → Produkt

  $\sin(x + y) = \dots$

- Produkt → Summe

  $\sin(x)\sin(y) = \dots$

- entspr. für Potenzen/Vielfache

  $\sin(7x), \sin^9(x)$

- einfache Herleitung über $\exp(inx)$

Trigonometrische Ausdrücke (II)

- jedes Polynom in $(\sin(x), \cos(x))$ läßt sich durch „Produkt → Summe“ eindeutig umformen zu

$$\sum a_k \sin(kx) + \sum b_k \cos(kx) + c$$

- jedes Polynom in $\sin(kx), \cos(kx)$ läßt sich durch „Summe → Produkt“ eindeutig umformen zu

$$P(\cos(x)) + \sin(x)Q(\cos(x))$$

- $\sin(x)$ und $\cos(x)$ lassen sich als rationale Funktionen von $\tan(x/2)$ ausdrücken

Anwendung: Integration rationaler Funktionen von $\sin(kx), \cos(kx) \Leftarrow$ Integration rationaler Funktionen (von x)

Algebraische Zahlen

Definition, Beispiele

Algebraische Zahl = Wurzel eines Polynoms mit ganzzahligen Koeffizienten.
Beispiel: $x^2 - x - 1 = 0$, $x = \dots$
Beispiel: $x = \sqrt{2} + \sqrt{3}$, welches Polynom?
ganzzahlige lineare Abhängigkeiten zwischen Potenzen von x ausnutzen
kann man numerisch raten (L^3-Algorithmus).

Algebraische Strukturen

- Halbgruppe $(M, \cdot)$
- Monoid $(M, \cdot, 1)$:

  Matrizen mit Multiplikation, Wörter mit Verkettung, Funktionen mit Komposition

- Gruppe, abelsche Gruppe
- Halbring $(M, +, 0, \cdot, 1)$: Wahrheitswerte, (Max,Plus), formale Sprachen
- Ring: relle Polynome, Matrizen
- Körper:
- Vektorraum über einem Körper

Euklidische Ringe

„Ring, in dem der Euklidische Algorithmus funktioniert“
Ring R mit Bewertung $g : R \setminus \{0\} \to \mathbb{N}$ heißt euklidisch, fal

- $a \neq 0 \wedge b \neq 0 \Rightarrow ab \neq 0$
- $a \neq 0 \wedge b \neq 0 \Rightarrow g(ab) \geq g(a)$
- $\forall a \neq 0 \forall b \exists q, r : b = qa + r \wedge (r = 0 \vee g(r) < g(a))$

welche Bewertungen für: ganze Zahlen, gaußsche Zahlen $\mathbb{Q}(\sqrt{-1})$, Polynome?
$\mathbb{Q}(\sqrt{-3})$ ist nicht euklidisch

Polynome

- Ein Polynom (in einer Variablen) über einem Körper K ist eine Folge $(a_n, \ldots, a_0) \in K^*$,
- ... geschrieben $a_n x^n + \ldots + a_0 x^0$.
- Satz: Wenn x_0 Nullstelle des Polynoms p, dann gibt es ein Polynom q mit $p = (x - x_0)q$.
- Beweis: Wir können q durch Division $p : (x - x_0)$ bestimmen, der Rest muß 0 sein!

Körper-Erweiterungen

- wir adjungieren zu einem Körper K die Nullstelle eines irreduziblen Polynoms p (vom Grad n)
- Es entsteht eine Menge M von Vektoren $(a_k, \ldots, a_0)$ bz Ausdrücken $a_k x^k + \ldots + a_0 x^0$
- ... mit einer Äquivalenzrelation, die erzeugt wird aus $p \equiv 0$ (durch Multiplikation mit Konstanten und x).
- $M/\equiv$ bildet den Erweiterungskörper von K.
- eine Nullstelle von p heißt algebraische Zahl.

Algebraische Zahlen

Ist eine Zahl (etwa durch einen Wurzel-Ausdruck) gegeben, können wir fragen, durch welche Körpererweiterung sie aus $\mathbb{Q}$ entsteht.
Wir bestimmen dazu das entsprechende Minimalpolynom.
Beispiel $\alpha = \sqrt{2 + 3\sqrt[3]{2}}$.

Rechnen mit algebraischen Zahlen

Man kann beweisen: Summe, Differenz, Produkt und Quotient algebraischer Zahlen sind wieder algebraisch.
D. h., wenn die Minimalpolynome der beiden Zahlen gegeben sind, muß man das Minimalpolynom des Resultats konstruieren.
Beispiel: $y = \sqrt{2} + \sqrt[3]{3}$.

Beispiel (Minimal-)Polynom

$y = \sqrt{2} + \sqrt[3]{3}$ bestimme alle Koeffizienten

$$M = \begin{array}{c|cccccc}
 & 1 & \sqrt[3]{3} & \sqrt{2} & \sqrt{2}\sqrt[3]{3} & \sqrt[3]{3^2} & \sqrt{2}\sqrt[3]{3^2} \\
\hline
y^0 & 1 & 0 & 0 & 0 & 0 & 0 \\
y^1 & 0 & 1 & 1 & 0 & 0 & 0 \\
y^2 & 2 & 0 & 0 & 2 & 1 & 0 \\
y^3 & 3 & 6 & 2 & 0 & 0 & 3 \\
y^4 & 4 & 3 & 12 & 8 & 12 & 0 \\
y^5 & 60 & 20 & 4 & 15 & 3 & 20 \\
\hline
y^6 & 17 & 90 & 120 & 24 & 60 & 18
\end{array}$$

nenne die Zeilenvektoren $x_0, \ldots, x_6$, löse lineare Gleichung

$$\left(x_0^T \ldots x_5^T\right) \cdot c^T = x_6^T$$

ergibt Koeffizientenvektor $c = (-1, 36, -12, 6, 6, 0)$, d. h. $-1 + 36y - 12y^2 + 6y^3 + 6y^4 = y^6$. Ein Polynom mit Nullstelle y ist deswegen

$$y^6 - 6y^4 - 6y^3 + 12y^2 - 36y + 1$$

.

LLL-Algorithmus

gegeben eine (hochgenaue) Näherung x einer algebraischen Zahl, gesucht das dazugehörige Polynom.
Bestimme eine kurzen Vektor in dem Gitter, das durch diese Spaltenvektoren aufgespannt wird:

$$\begin{pmatrix} 1 & & 0 \\ & \ddots & \\ & & 1 \\ F \cdot x^d & \ldots & F \cdot x^0 \end{pmatrix}$$

Dabei F sehr groß wählen.
Matrix berechnen durch diese Mupad-Funktion

```
M := (x, d, f) -> [[(if s = z then
                       1
```

```
                    else
                      if z = d + 1 then
                        round(f*x^((d +
                      else
                        0
                      end_if
                    end_if) $ s = 0..d]
```

```
DIGITS := 50
x:=2^(1/2)+3^(1/3)
```

kurzen Gittervektor finden durch LLL-Algorithmus (Lenstra,Lenstra,Lovasz), findet nicht den kürzesten, hat dafür aber vernünftige Laufzeit.

```
lllint (M(x,6,10^40)  -- verschiedene Dime
```

liefert Kandidaten für Koeffizienten eines Polynoms mit

Nullstelle x.
Andere Anwendung (Übung): finde nahe beieinanderliegende (Primzahl-)Produkte Matrix wie oben, untere Zeile besteht aus
$F\log(2), F\log(3), F\log(5), F\log(7)$. Für $F = 10^6$ findet man die Koeffizienten $[4, -2, 11, -2, -6]$, das entspricht
$2^4 5^1 1 = 781250000, 3^2 * 7^2 * 11^6 = 781258401$. vgl. abc-Vermutung und Tabelle `http://www.mathematik.uni-jena.de/~aros/abc.html`

Gröbnerbasen

Begriff

- Gröbnerbasis: endliche Darstellung für ein Polynom-Ideal,
- mit der man entscheiden kann, ob ein gegebenes Polynom zum Ideal gehört.
- Anwendung: Beweis geometrischer Aussagen, die sich algebraisch formulieren lassen.
- Erfinder: Bruno Buchberger 1965

Literatur: H.-G. Gräbe: Gröbnerbasen und Anwendungen (Skript, Uni Leipzig); V. Messerschmidt: Automatisches

Beweisen in der ebenen Geometrie mittels Gröbnerbasen (Diplomarbeit, Uni Kassel), F. Baader und T. Nipkow: Term Rewriting and all that (Kapitel 8), Cambridge Univ. Press 1998.

Anwendung

Beispiel (Feuerbachkreis): in jedem Dreieck gilt: die Höhenfußpunkte und die Seitenmitten liegen auf einem Kreis (es gilt noch viel mehr).
algebraische Behandlung: für jeden Punkt $P_i = (x_i, y_i)$ zw Variablen, für jede gegebene/konstruierte Bedingung eine Gleichung, ergibt Gleichungssystem (von Gleichungen de Form „Polynom $= 0$“).
Die Frage ist, ob behauptete Aussagen daraus folgen, d. ob entsprechende Polynome dann auch $= 0$ sind.

Ideale

Ring $R = (M, 0, +, 1, \cdot)$
Ideal I ist Menge $\subseteq M$ mit

- abgeschlossen unter Summe: $\forall x \in I, y \in I : (x + y) \in I$.
- abgeschlossen bzgl. Multiplikation mit M: $\forall x \in I, y \in M : (xy) \in I$.

Das von $M = \{x_1, \dots, x_n\}$ erzeugte Ideal ist

$$\mathrm{Ideal}(M) = \{\sum c_i x_i \mid c_i \in M\}$$

- wie kann man Polynomideale vernünftig darstellen?
- wie kann man entscheiden, ob $P \in \mathrm{Ideal}(\{P_1, \dots, P_n\})$?

Ordnung auf Monomen

wählen eine totale und wohlfundierte Ordnung auf Monomen:

- wenn $m_1 \mid m_2$, dann $m_1 \le m_2$
- wenn $m_1 < m_2$, dann $mm_1 < mm_2$

(z. B. lexikografisch, oder: erst nach Exponentensumme, dann lexikografisch)
$H(f)$ = führendes (größtes) Monom, $R(f)$ = alle anderen
also $f = H(f) + R(f)$

Reduktion von Polynomen

$p \to_f q$, falls p ein Monom $a \cdot m$ enthält und ein m' exist. mit $m = H(f)m'$ und $q = p - am'f$.
F eine endliche Menge von Polynomen, dann definiere $\to_F = \bigcup\{\to_f \mid f \in F\}$.
Satz: $\to_F$ terminiert immer.
Beweis-Idee: wenn $g \to_f g'$, dann wird in g das Monom $H(f)$ gelöscht und durch eventuell mehrere, aber jedenfalls kleinere ersetzt. Weil $>$ auf Monomen nach Voraussetzung terminiert, terminiert auch die dadurch erzeugte Ordnung auf Polynomen, die man so definiert: $p \gg q$ falls: Monome jeweils bzgl. $>$ absteigend sortieren und dann beide Folgen lexikografisch vergleichen. Beachte: da werden immer nur die Monome (ohne Koeffizienten) verglichen.

Gröbnerbasen

die eben definierte Relation $\to_F$ ist nicht notwendig konfluent.
Beispiel (8.2.10 aus Baader/Nipkow) $F = \{f_1, f_2\}$ mit $f_1 = x^2y - x^2, f_2 = xy^2 - y^2$. Ordnung $>$ auf Monomen: er nach Exponentensumme, dann lexikografisch.
$x^2y^2 \to_{f_1} x^2y^2 - y \cdot f_1 = x^2y \to_{f_1} x^2y - 1 \cdot f_1 = x^2$ und $x^2y^2 \to_{f_2} ? \to_{f_2} y^2$. Beides sind Normalformen, also $\to_F$ nicht konfluent.
Definition: eine endliche Menge G von Polynomen heißt *Gröbnerbasis* für ein Polynomideal I, falls $\mathrm{Ideal}(G) = I$ u $\to_G$ konfluent.
Nicht-Beispiel (Fortsetzung) F ist keine Gröbnerbasis.

S-Polynome

Man erhält eine Gröberbasis für eine gegebene Menge F durch Vervollständigung (Hinzufügen von Polynomen $s \in \mathrm{Ideal}(F)$)

Für alle Paare $f_1, f_2 \in F$: bestimme $m = \mathrm{lcm}(H(f_1), H(f_2))$ (kleinstes gemeinsames Vielfaches der Köpfe), dann $S(f_1, f_2) := f_1 \cdot m/H(f_1) - f_2 \cdot m/H(f_2)$. Es gilt: $S(f_1, f_2) \in \mathrm{Ideal}(F)$.

Beispiel ($F = \{f_1, f_2\}$ wie oben.)
$S(f_1, f_2) = f_1 y - f_2 x = x^2y - xy^2$

Satz: Wenn für alle $f_1, f_2 \in F : S(f_1, f_2) \to_F 0$, dann ist F eine Gröbnerbasis.

Beispiel: $x^2y - xy^2 \to_F x^2 - y^2 \not\to_F 0$.

Der Buchberger-Algorithmus

Eingabe: endliche Menge F von Polynomen und Ordnung
$>$ auf Monomen.

Ausgabe: eine Gröbnerbasis G für $\mathrm{Ideal}(F)$.

1. Beginne mit $G := F$.
2. wähle $f_1, f_2 \in G$, bestimme eine $\to_G$-Normalform s von $S(f_1, f_2)$.
3. falls $s \neq 0$, füge s zu G hinzu und gehe zu 2.
4. falls man in (2.) nicht so wählen konnte, daß in (3.) $s \neq$
dann Ausgabe G.

Satz: Dieser Algorithmus hält nach endliche vielen Schritten und gibt eine Gröbnerbasis G für F aus.

Den Test in (4.) muß man geeignet implementieren (man wird nicht immer alle Paare vergleichen)

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,
$S(f_2, f_3) = xf_2 - y^2 f_3 = -xy^2 + y^4 \to_{f_2} y^4 - y^2 \to_{f_4}$
$y^3 - y^2 \to_{f_4} 0$.

mit Mupad:

```
groebner::gbasis([x^2*y-x^2,x*y^2-y^2],Degre
```


Der Buchberger-Algorithmus (II)

Beachte: Wielange das dauert und welche Basis man erhält, hängt von der gewählten Ordnung auf Monomen a
Die Schrittzahl kann doppelt exponentiell sein. Mit manchen Ordnungen geht es „meistens“ schneller, da gib
es aber nur Erfahrungswerte.

Anwendung: Mit einer Gröbnerbasis kann man das Ideal-Membership-Problem entscheiden:
$f \in \mathrm{Ideal}(G) \iff f \to_G 0$.

Anwendung: Idealmitgliedschaft

automatisches Beweisen von Behauptungen der Form:
Wenn $p_1(x_1, \ldots) = 0 \wedge \wedge p_n(x_1, \ldots) = 0$, dann $q(x_1, \ldots) = 0$.
äquivalent: $q \in \mathrm{Ideal}(p_1, \ldots, p_n)$.
äquivalent: es gibt kein $(x_1, \ldots)$ mit
$p_1(x_1, \ldots) = 0 \wedge \wedge p_n(x_1, \ldots) = 0 \wedge q(x_1, \ldots) \neq 0$.
äquivalent (Rabinowitsch-Trick): es gibt kein $(x_1, \ldots)$ mit
$p_1(x_1, \ldots) = 0 \wedge \wedge p_n(x_1, \ldots) = 0 \wedge 1 - f \cdot q(x_1, \ldots) = 0$.
äquivalent: Gröbnerbasis dieser Polynome ist $\{1\}$, d. h. das Ideal besteht aus allen Polynomen.

Vgl. Resolution in der Logik: Falls $(A_1 \wedge \ldots \wedge A_n \wedge \not B)$ widersprüchlich, dann $(A_1 \wedge \ldots \wedge A_n \implies B)$; und: aus widersprüchlicher Formel(menge) folgt jede Formel.

Anwendung: Geometrie

Seitenhalb. schneiden sich in einem Punkt:

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

groebner::gbasis([
 mittelpunkt(a,f,b), mittelpunkt(b,d,c), m
 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(
senkrecht:=(a,b,c,d)-> ((a.1-b.1)*(c.1-d.1
```



```
groebner::gbasis([
senkrecht(a,c,b,c), mittelpunkt(a,m,b),
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,a
gleichseitig:=(a,b,c)->(gleichweit(a,b,b,c),

groebner::gbasis([
a.1,a.2,b.2,
gleichseitig(a,b,p), schwerpunkt(a,b,p,u),
gleichseitig(b,c,q), schwerpunkt(b,c,q,v),
```

```
gleichseitig(c,a,r), schwerpunkt(c,a,r,w),
1 - y * gleichseitig(u,v,w)
])
```

vgl. `http://www.symbolicdata.org/`