

# Die Zukunft der algorithmischen Mathematik: Kann mathematische Forschung automatisiert werden?

Bruno Buchberger  
RISC

(Research Institute for Symbolic Computation)  
Johannes Kepler University, Linz, Austria  
bruno.buchberger@jku.at

Vortrag am 21. November 2006 in Graz  
Abendveranstaltung von OCG und OVE

Copyright Note: Copyright Bruno Buchberger 2006.

This file may be copied and stored under the following conditions:

- The file is kept unchanged including this copyright note.
- A mail is sent to [bruno.buchberger@jku.at](mailto:bruno.buchberger@jku.at)
- If material from this file is used, the talk should be appropriately cited.

## Beispiel: Rechnen 1. Klasse

Algorithmik der letzten 40 Jahre

Algorithmik einen Stock höher: Automatisches Beweisen

Noch einen Stock höher: Automatisches Erfinden

Durchgehendes Beispiel: Die Theorie der Gröbner-Basen

## Beispiel: Arithmetik

---

### Ein Problem aus der "realen" Welt:

Was ist mehr wert?

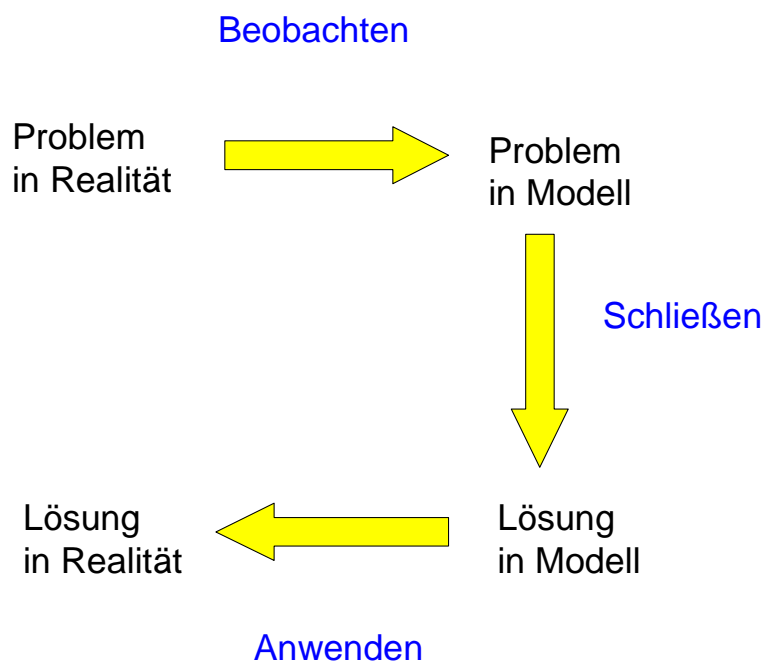
(Wenn 1 Kamel so viel wert ist wie 7 Schafe.)



## Lösung ohne Mathematik ?

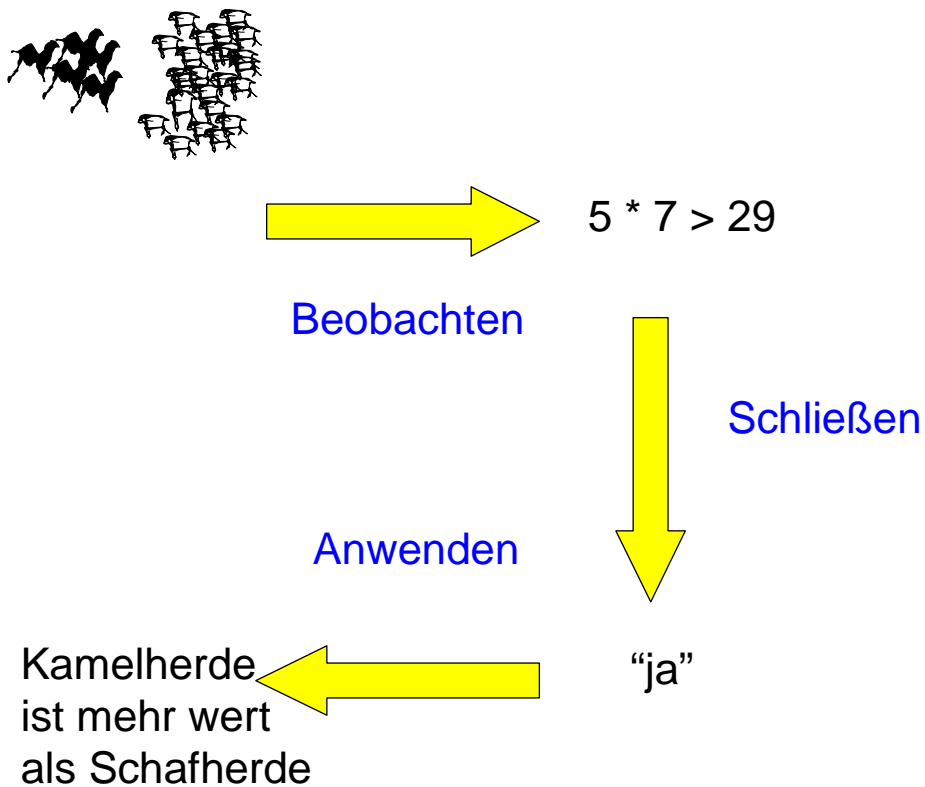


## Lösung mit Mathematik



- **Mathematik ist charakterisiert durch die Methode, das Schließen.**

In unserem Beispiel:



Das Addieren, Multiplizieren, Vergleichen von Zahlen (in Dezimaldarstellung) wird so genau verstanden, dass jeder "Computer" das machen kann:

```

257 * 348 =
  771
 1028
 2056
-----
 89436

```

```

257 * 348

```

```

89436

```

**Ein Teil des Lebens wird damit "automatisiert":**

Problemlösen auf "Knopfdruck", keine "Einsicht" nötig, ...

- **Mathematik ist im Zentrum der Automatisierung.**
- **Automatisierung des "Schließens" ("Rechnens", Arbeitens in Modellen, ...).**
- **Prinzip der Mathematik: Einmal gründlich (an den Grund gehend) denken, und (unendlich) oft nicht mehr denken müssen.**
- **Beispiel: Einmal beweisen, dass Multiplizieren und Addieren "distributiv" sind und dann nur mehr mit dem "kleinen 1 x 1" rechnen müssen.**

für alle  $x, y, z$ :  $(x + y) * z = x * z + y * z$

$$257 * 348 = (2 H + 5 Z + 7 E) * (3 H + 4 Z + 8 E) = (6 H * H + \dots + 56 E)$$

- **"Das Wunder" der Mathematik: Beweise über unendlich viele Situationen sind in endlich vielen Denkschritten möglich!**

## Beispiel: Rechnen

Algorithmik der letzten 40 Jahre

Algorithmik einen Stock höher: Automatisches Beweisen

Noch einen Stock höher: Automatisches Erfinden

Durchgehendes Beispiel: Die Theorie der Gröbner-Basen

---

## Mathematisches Wissen und mathematische Methoden

In der Mathematik erfindet und beweist man **Wissen** und **Methoden**.

**Wissen** und **Methoden** sind nur zwei Seiten derselben Medaille.

Nur eine Seite auszuleben, ist eine gewaltige **Verarmung** der Mathematik.

Die letzten 40 Jahre Mathematik haben eine Explosion der mathematischen "**Algorithmik**" (Methoden) gebracht.

Die Motivation dafür kam aus "dem **Computer**" (eine mathematische Erfindung!)

Die neue mathematische **Algorithmik** ist auf neuem mathematischen **Wissen** gegründet.

**Beispiele:** Computer-Analysis, Computer-Algebra, Kryptographie, ..., automatisches Beweisen.

---

## Beispiel: Groebner-Basen (BB 1965, ...)

- **Warum** sind Gröbner-Basen wichtig?
- **Was** sind Gröbner-Basen?
- **Wie** kann man Gröbner-Basen berechnen?

---

## Warum Groebner-Basen?

- Dutzende **fundamentaler Probleme** der Mathematik können auf das Problem der Konstruktion von Gröbner-Basen zurückgeführt werden. Es werden ständig mehr.
- Dementsprechend: 10 Lehrbücher, **1000 Publikationen**, 3000 Citations, millionenfach installiert, eigenes Stichwort in der AMS-Klassifikation der Mathematik

Siehe Special Semester on Gröbner Bases: [www.risc.uni-linz.ac.at](http://www.risc.uni-linz.ac.at)

Übersicht: B. B., F. Winkler. *Gröbner Bases: Theory and Applications*. Cambridge University Press, 1998. 560 pages.



## Was sind Groebner-Basen?

Eine Art "kanonische Form" von nicht-linearen ("polynomialen") (Gleichungs-) systemen.

## Linearkombinationen von Polynomen

$$\begin{aligned} f_1 &= -2y + xy \\ f_2 &= -x^2 + y^2 \end{aligned}$$

Führende Potenzprodukte:

Linearkombinationen von  $f_1$  von  $f_2$ , z.B.

$$g = (y) f_1 + (-x + 2) f_2$$

$$-2x^2 + x^3$$

Beobachtung: führendes Potenzprodukt  $x^3$  von  $g$  ist

weder ein Vielfaches von  $xy$  of  $f_1$

noch ein Vielfaches von  $y^2$  of  $f_2$ .

## Definition

Polynommenge  $F$  heißt **Groebner-Basis**, wenn das obige Phänomen nicht passieren kann, d.h. wenn

für alle  $f_1, \dots, f_m \in F$  und alle (**unendlich vielen**) Polynome  $h_1, \dots, h_m$ ,

führendes Potenzprodukt von  $h_1 f_1 + \dots + h_m f_m$

ist Vielfaches des führenden Potenzprodukts von mindestens einem der Polynome in  $F$ .

## Gegenbeispiel und Beispiel einer Gröbner-Basis

Gegenbeispiel: Die Menge  $\{-2y + xy, -x^2 + y^2\}$  von oben ist keine Gröbner-Basis. Warum?

Beispiel: Die Menge  $\{-2x^2 + x^3, -2y + xy, -x^2 + y^2\}$  ist eine Gröbner-Basis. Warum?

## Der Hauptsatz der Theorie der Gröbner-Basen (BB 1965):

$$F \text{ ist eine Gröbner-Basis} \iff \forall_{f_1, f_2 \in F} \text{rest}[F, \text{S-Polynom}[f_1, f_2]] = 0.$$

$$\text{s-polynomial}[-2y + xy, -x^2 + y^2] = y(-2y + xy) - x(-x^2 + y^2)$$

$$x^3 - 2y^2$$

**Beweis:** nichttrivial, kombinatorisch.

Das Theorem **reduziert** einen **unendlichen** Test auf einen **endlichen** Test, nämlich ...

Die Kraft der Gröbner-Basen Theorie beruht auf diesem Satz (in der Erfindung des Begriffs des S-Polynoms) und seinem Beweis.

## Das Problem der *Konstruktion* von Gröbner-Basen

Gegeben  $F$ , finde  $G$ , sodass  $G$  eine (endliche) Gröbner-Basis ist

und  $F$  und  $G$  dieselbe Menge von Linearkombinationen erzeugen.

## Ein Algorithmus zur *Konstruktion* von Gröbner-Basen (BB 1965)

Nochmals der Hauptsatz:

$$F \text{ ist eine Gröbner-Basis} \iff \forall_{f_1, f_2 \in F} \text{rest}[F, \text{S-Polynom}[f_1, f_2]] = 0.$$

Darauf basiert der folgende Algorithmus zur Konstruktion von Gröbner-Basen:

$G := F$ .

für alle Paare  $f_1, f_2 \in G$ :

$h := \text{rest}[G, \text{S-Polynom}[f_1, f_2]]$

Wenn  $h = 0$ , betrachte das nächste Paar.

Wenn  $h \neq 0$ , für  $h$  zu  $G$  hinzu und iteriere.

## Termination des Algorithmus

Mit Dickson's Lemma (Dickson 1913, BB 1970).

## Beispiel einer Anwendung: Beliebige nichtlineare Gleichungssysteme

```
f1 = x y - 2 y z - z;
f2 = y2 - x2 z + x z;
f3 = z2 - y2 x + x;
F = {f1, f2, f3};
```

```
{time, G} = GroebnerBasis[F] // Timing
```

```
{0.047 Second,
{-z - 4 z3 + 17 z4 - 3 z5 + 45 z6 - 60 z7 + 29 z8 - 124 z9 + 48 z10 - 64 z11 + 64 z12,
-22001 z + 14361 y z + 16681 z2 + 26380 z3 + 226657 z4 + 11085 z5 -
90346 z6 - 472018 z7 - 520424 z8 - 139296 z9 - 150784 z10 + 490368 z11,
43083 y2 - 11821 z + 267025 z2 - 583085 z3 + 663460 z4 - 2288350 z5 +
2466820 z6 - 3008257 z7 + 4611948 z8 - 2592304 z9 + 2672704 z10 - 1686848 z11,
43083 x - 118717 z + 69484 z2 + 402334 z3 + 409939 z4 + 1202033 z5 -
2475608 z6 + 354746 z7 - 6049080 z8 + 2269472 z9 - 3106688 z10 + 3442816 z11}}
```

```
zsol = NSolve[G[[1]] == 0, z]
```

```
{{z → -0.331304 - 0.586934 i}, {z → -0.331304 + 0.586934 i},
{z → -0.296413 - 0.705329 i}, {z → -0.296413 + 0.705329 i},
{z → -0.163124 - 0.37694 i}, {z → -0.163124 + 0.37694 i},
{z → 0.}, {z → 0.0248919 - 0.89178 i}, {z → 0.0248919 + 0.89178 i},
{z → 0.468852}, {z → 0.670231}, {z → 1.39282}}
```

```
Gsubnum = G /. zsol[[1]]
```

```
{1.11022 × 10-15 + 5.55112 × 10-16 i,
(-523.519 - 4967.65 i) - (4757.86 + 8428.97 i) y,
(-7846.9 - 8372.06 i) + 43083 y2, (-16311.7 + 16611. i) + 43083 x}
```

```
ysol = NSolve[Gsubnum[[2]] == 0, y]
```

```
{{y → -0.473535 - 0.205184 i}}
```

**Theorem** (Roider, Kalkbrener et al. 1990): It suffices to consider the poly in y with lowest degree.

```
xsol = NSolve[Gsubnum[[4]] == 0, x]
```

```
{{x → 0.378611 - 0.385558 i}}
```

```
F /. zsol[[1]] /. ysol[[1]] /. xsol[[1]]
```

```
{-3.21965 × 10-15 - 3.45557 × 10-15 i,
 4.02456 × 10-15 - 8.04912 × 10-16 i, 5.07927 × 10-15 + 1.83187 × 10-15 i}
```

## Beispiel einer Anwendung: Invariantentheorie

Frage: Kann

```
h = x17 x2 - x1 x27
```

```
x17 x2 - x1 x27
```

polynomial durch die Polynome in folgender Menge ausgedrückt werden:

```
F = {x12 + x22, x12 x22, x13 x2 - x1 x23}
```

```
{x12 + x22, x12 x22, x13 x2 - x1 x23}
```

?

(Diese Polynomie sind die "Fundamentalinvarianten der Gruppe"  $\mathbb{Z}_4$ ).

## Reduktion auf die Berechnung von Gröbner-Basen

```
{time, GB} = GroebnerBasis[
  {-i1 + x12 + x22, -i2 + x12 x22, -i3 + x13 x2 - x1 x23}, {x2, x1, i3, i2, i1}] // Timing
```

```
{0. Second,
{-i12 i2 + 4 i22 + i32, i2 - i1 x12 + x14, -i12 i3 x1 + 2 i2 i3 x1 + i1 i3 x13 - i12 i2 x2 + 4 i22 x2,
i12 x1 - 2 i2 x1 - i1 x13 + i3 x2, -i1 i3 + 2 i3 x12 - i12 x1 x2 + 4 i2 x1 x2,
-i3 x1 - 2 i2 x2 + i1 x12 x2, -i3 - i1 x1 x2 + 2 x13 x2, -i1 + x12 + x22}}
```

```
PolynomialReduce[x17 x2 - x1 x27, GB,
{x2, x1, i3, i2, i1}, MonomialOrder → Lexicographic]
```

```
{{0, i3 +  $\frac{1}{2}$  i1 x1 x2 + x13 x2, 0,  $\frac{3 i_1 x_2}{4} - \frac{1}{2} x_1^2 x_2 + \frac{x_2^3}{2}$ , i1 -  $\frac{x_1^2}{2} + \frac{3 x_2^2}{4}$ ,
 $\frac{3 i_1 x_1}{2} + x_1 x_2^2$ ,  $\frac{x_2^4}{2}$ , - $\frac{1}{4} i_1^2 x_1 x_2 - \frac{1}{2} i_1 x_1 x_2^3 - x_1 x_2^5$ }, i12 i3 - i2 i3}}
```

**Theorem** (Sweedler, Sturmfels et al. 1988):  $h$  kann durch ausgedrückt werden  $F$  iff der Rest bzgl. der "Groebner-Basis von  $F$  mit Hilfsvariablen" ein Polynom in den Hilfsvariablen ist (das die Darstellung liefert).

```
i12 i3 - i2 i3 /. {i1 -> x12 + x22, i2 -> x12 x22, i3 -> x13 x2 - x1 x23} // Expand
```

```
x17 x2 - x1 x27
```

```
R = PolynomialReduce[x16 x2 - x1 x26, GB,
  {x2, x1, i3, i2, i1}, MonomialOrder -> Lexicographic]
```

```
{ {0, -1/2 i1 x1 + i1 x2 + x12 x2, 0, 3/4 i1 - x12/2 + x22/2,
  -x1/4 + 3x2/4, 3/4 i1 + x1 x2, x23/2, -1/4 i12 x1 - 1/2 i1 x1 x22 - x1 x24},
  -i13 x1 + 2 i1 i2 x1 + 1/2 i1 i3 x1 + i12 x13 - i2 x13 + 1/2 i3 x13 + 1/2 i1 i2 x2 }
```

$x_1^6 x_2 - x_1 x_2^6$  can not be expressed by the fundamental invariants in I.

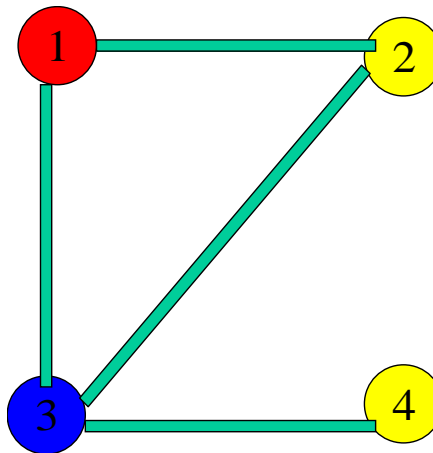
## Beispiel einer Anwendung: Färben von Graphen

**Problem:**

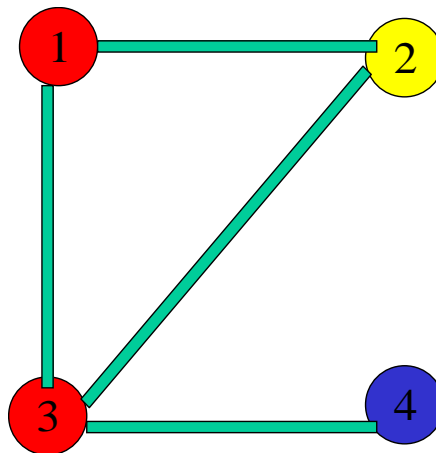
Find alle zulässigen Färbungen in k Farben eines Graphen mit n Knoten und Kanten E.

Beispiel: 3 Farben, 4 Knoten und Kanten {1,2}, {1,3}, {2,3}, {3,4}:

Zulässig:



Nicht zulässig:



## Reduktion auf die Konstruktion von Gröbner-Basen

**Theorem (D. Bayer, 1980):** Die zulässigen Färbungen entsprechen 1-1 den Lösungen des folgenden Gleichungssystems:

```

{ -1 + x13, ... at vertex 1 color is a 3 - ary root of 1
  -1 + x23, ... at vertex 2 color is a 3 - ary root of 1
  -1 + x33,
  -1 + x43,
  x12 + x1 x2 + x22, ... the colors at 1 and 2 must be different,
  x12 + x1 x3 + x32,
  x22 + x2 x3 + x32,
  x32 + x3 x4 + x42 }
  
```

## Lösung durch Gröbner-Basen

```

GB = GroebnerBasis[{-1 + x13, -1 + x23, -1 + x33, -1 + x43,
  x12 + x1 x2 + x22, x12 + x1 x3 + x32, x22 + x2 x3 + x32, x32 + x3 x4 + x42},
  {x4, x3, x2, x1}]
  
```

```

{-1 + x13, x12 + x1 x2 + x22, x1 + x2 + x3, x1 x2 - x1 x4 - x2 x4 + x42}
  
```

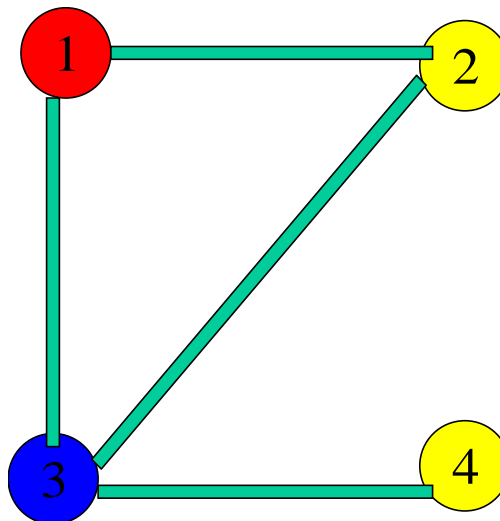
```
Solve[{-1 + x1^3 == 0, -1 + x2^3 == 0, -1 + x3^3 == 0, -1 + x4^3 == 0,
  x1^2 + x1 x2 + x2^2 == 0, x1^2 + x1 x3 + x3^2 == 0, x2^2 + x2 x3 + x3^2 == 0, x3^2 + x3 x4 + x4^2 == 0},
{x4, x3, x2, x1}]
```

```
{ {x4 -> 1, x1 -> 1, x2 -> -1 + (-1)^(1/3), x3 -> -(-1)^(1/3)},
  {x4 -> 1, x1 -> 1, x2 -> -1 - (-1)^(2/3), x3 -> (-1)^(2/3)},
  {x4 -> 1, x2 -> 1, x1 -> -1 + (-1)^(1/3), x3 -> -(-1)^(1/3)},
  {x4 -> 1, x2 -> 1, x1 -> -1 - (-1)^(2/3), x3 -> (-1)^(2/3)},
  {x4 -> -(-1)^(1/3), x2 -> -1 + (-1)^(1/3), x1 -> -(-1)^(1/3), x3 -> 1},
  {x4 -> -(-1)^(1/3), x2 -> -1 - (-1)^(2/3), x1 -> (-1)^(2/3), x3 -> 1},
  {x4 -> (-1)^(2/3), x2 -> -1 + (-1)^(1/3), x1 -> -(-1)^(1/3), x3 -> 1},
  {x4 -> (-1)^(2/3), x2 -> -1 - (-1)^(2/3), x1 -> (-1)^(2/3), x3 -> 1},
  {x4 -> -1 + (-1)^(1/3), x1 -> 1, x2 -> -1 + (-1)^(1/3), x3 -> -(-1)^(1/3)},
  {x4 -> -1 + (-1)^(1/3), x2 -> 1, x1 -> -1 + (-1)^(1/3), x3 -> -(-1)^(1/3)},
  {x4 -> -1 - (-1)^(2/3), x1 -> 1, x2 -> -1 - (-1)^(2/3), x3 -> (-1)^(2/3)},
  {x4 -> -1 - (-1)^(2/3), x2 -> 1, x1 -> -1 - (-1)^(2/3), x3 -> (-1)^(2/3)}
```

Leicht umgeordnet:

```
{ {x1 -> 1, x2 -> -(-1)^(1/3), x3 -> -1 + (-1)^(1/3), x4 -> 1},
  {x1 -> 1, x2 -> -(-1)^(1/3), x3 -> -1 + (-1)^(1/3), x4 -> -(-1)^(1/3)},
  {x1 -> 1, x2 -> (-1)^(2/3), x3 -> -1 - (-1)^(2/3), x4 -> 1},
  {x1 -> 1, x2 -> (-1)^(2/3), x3 -> -1 - (-1)^(2/3), x4 -> (-1)^(2/3)},
  {x1 -> -(-1)^(1/3), x2 -> 1, x3 -> -1 + (-1)^(1/3), x4 -> 1},
  {x1 -> -(-1)^(1/3), x2 -> 1, x3 -> -1 + (-1)^(1/3), x4 -> -(-1)^(1/3)},
  {x1 -> -(-1)^(1/3), x2 -> -1 + (-1)^(1/3), x3 -> 1, x4 -> -(-1)^(1/3)},
  {x1 -> -(-1)^(1/3), x2 -> -1 + (-1)^(1/3), x3 -> 1, x4 -> -1 + (-1)^(1/3)},
  {x1 -> (-1)^(2/3), x2 -> 1, x3 -> -1 - (-1)^(2/3), x4 -> 1},
  {x1 -> (-1)^(2/3), x2 -> 1, x3 -> -1 - (-1)^(2/3), x4 -> (-1)^(2/3)},
  {x1 -> (-1)^(2/3), x2 -> -1 - (-1)^(2/3), x3 -> 1, x4 -> (-1)^(2/3)},
  {x1 -> (-1)^(2/3), x2 -> -1 - (-1)^(2/3), x3 -> 1, x4 -> -1 - (-1)^(2/3)}
```

Zum Beispiel,  $\{x_1 \rightarrow 1, x_2 \rightarrow -(-1)^{1/3}, x_3 \rightarrow -1 + (-1)^{1/3}, x_4 \rightarrow -(-1)^{1/3}\}$  entspricht



## Beispiel einer Anwendung: Ganzzahlige Optimierung

Beispiel (B. Sturmfels):

Was ist die minimale Anzahl von Münzen (z.B.  $p$  Pennies,  $n$  Nickels,  $d$  Dimes,  $q$  Quarters), um einen bestimmten Wert, z.B. 117, darzustellen:

Reduktion auf die Konstruktion von Gröbner-Basen (C. Traverso et al. 1986):

Kodiere die Werte  $p, n, d, q$  als Exponenten von Potenzprodukten!

Kodiere die Zielfunktion als den verallgemeinerten Grad der Potenzprodukte!

Kodiere die Wechsel-Regeln zwischen den Münzen als Polynome:

$$F = \{P^5 - N, P^{10} - D, P^{25} - Q\}$$

$$\{-N + P^5, -D + P^{10}, P^{25} - Q\}$$

Berechne nun die Gröbner-Basis von  $F$  (bzgl. einer Grad-Ordnung):

$$G = \text{GroebnerBasis}[F, \text{MonomialOrder} \rightarrow \text{DegreeLexicographic}]$$

$$\{-D + N^2, D^3 - NQ, D^2N - Q, -N + P^5\}$$

Nun kann man sicher sein, dass jede zulässige Lösung (z.B. ( $p=17, n=10, d=5, q=0$ ), durch Restbildung bzgl.  $G$ , zu einer minimalen Lösung führt.

$$\text{PolynomialReduce}[P^{17} N^{10} D^5, G, , \text{MonomialOrder} \rightarrow \text{DegreeLexicographic}]$$

$$\{\{D^9 P^{17} + D^8 N^2 P^{17} + D^7 N^4 P^{17} + D^6 N^6 P^{17} + D^5 N^8 P^{17} + D^4 P^{17} Q^2 + P^7 Q^4, \\ D^7 P^{17} + D^4 N P^{17} Q + D^2 P^{17} Q^2, P^{17} Q^3, D P^2 Q^4 + N P^7 Q^4 + P^{12} Q^4\}, D N P^2 Q^4\}$$

Antwort: nimm 4 quarters, 1 dime, 1 nickel, 2 pennies.

## Beispiel: Arithmetik

Algorithmik der letzten 40 Jahre

Algorithmik einen Stock höher: Automatisches Beweisen

Noch einen Stock höher: Automatisches Erfinden

Durchgehendes Beispiel: Die Theorie der Gröbner-Basen

### Beispiel: Automatisches Beweisen (Widerlegen) in der Geometrie (BB, Stifter, Kutzler, Kapur, ... 1984, ...)

Reduktion auf das Problem der Konstruktion von Gröbner-Basen:

Geo Theorem  $\rightarrow$  ( durch Koordinatisierung )

$\forall_{x,y,\dots} ( \text{poly1}(x,y,\dots)=0 \wedge \dots \Rightarrow \text{poly}(x,y,\dots)=0 ) \rightarrow$

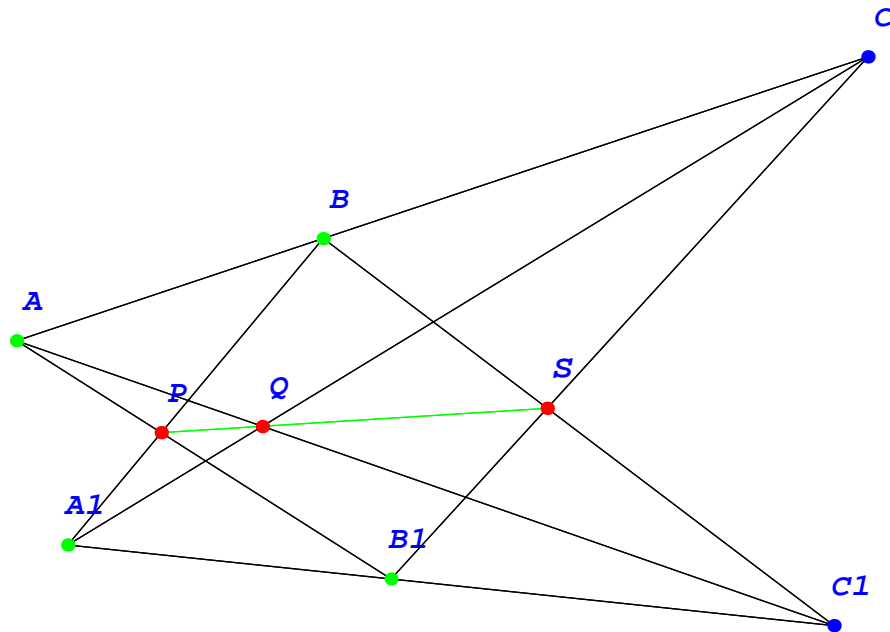
$\neg \exists_{x,y,\dots} ( \text{poly1}(x,y,\dots)=0 \wedge \dots \wedge \text{poly}(x,y,\dots) \neq 0 ) \rightarrow$

$\neg \exists_{x,y,\dots,a} ( \text{poly1}(x,y,\dots)=0 \wedge \dots \wedge a \cdot \text{poly}(x,y,\dots) - 1 = 0 )$

Die letzte Frage kann durch den Gröbner-Basen-Algorithmus entschieden werden!

### Beispiel: Pappus-Theorem

- Geometrischer Inhalt:



- Formulierung im Lehrbuch:

Seien  $A, B, C$  und  $A_1, B_1, C_1$  auf zwei Geraden und  $P = AB_1 \cap A_1B$ ,  $Q = AC_1 \cap A_1C$ ,  $S = BC_1 \cap B_1C$ . Dann sind  $P, Q$ , und  $S$  kollinear.

- Input in den GB-Beweiser:

```
Proposition["Pappus", any[A, B, A1, B1, C, C1, P, Q, S],
  point[A, B, A1, B1] ^ pon[C, line[A, B]] ^ pon[C1, line[A1, B1]] ^
  inter[P, line[A, B1], line[A1, B]] ^ inter[Q, line[A, C1], line[A1, C]] ^
  inter[S, line[B, C1], line[B1, C]] => collinear[P, Q, S]]
```

- Aufruf:

```
Prove[Proposition["Pappus"], by -> GeometryProver,
  ProverOptions -> {Method -> "GroebnerProver", Refutation -> True}]
```

- Folgender Beweis wird nun automatisch erzeugt:

Prove:

(Proposition (Pappus))

```

$$\forall_{A, B, A_1, B_1, C, C_1, P, Q, S} (\text{point}[A, B, A_1, B_1] \wedge \text{pon}[C, \text{line}[A, B]] \wedge$$


$$\text{pon}[C_1, \text{line}[A_1, B_1]] \wedge \text{inter}[P, \text{line}[A, B_1], \text{line}[A_1, B]] \wedge$$


$$\text{inter}[Q, \text{line}[A, C_1], \text{line}[A_1, C]] \wedge$$


$$\text{inter}[S, \text{line}[B, C_1], \text{line}[B_1, C]] \Rightarrow \text{collinear}[P, Q, S])$$

```

with no assumptions.

To prove the above statement we shall use the Gröbner basis method. First we have to transform the problem into algebraic form.

Algebraic Form:

To transform the geometric problem into algebraic form we have to chose first an orthogonal coordinate system.

Let's have the origin in point **A**, and points  $\{B, C\}$  on the two axes.

Using this coordinate system we have the following points:

$$\{\{A, 0, 0\}, \{B, 0, u_1\}, \{A1, u_2, u_3\}, \{B1, u_4, u_5\}, \\ \{C, 0, u_6\}, \{C1, u_7, x_1\}, \{P, x_2, x_3\}, \{Q, x_4, x_5\}, \{S, x_6, x_7\}\}$$

The algebraic form of the assertion is:

$$(1) \quad \forall_{x_1, x_2, x_3, x_4, x_5, x_6, x_7} (u_3 u_4 + -u_2 u_5 + -u_3 u_7 + u_5 u_7 + u_2 x_1 + -u_4 x_1 == 0 \wedge \\ u_5 x_2 + -u_4 x_3 == 0 \wedge -u_1 u_2 + u_1 x_2 + -u_3 x_2 + u_2 x_3 == 0 \wedge \\ x_1 x_4 + -u_7 x_5 == 0 \wedge -u_2 u_6 + -u_3 x_4 + u_6 x_4 + u_2 x_5 == 0 \wedge \\ u_1 u_7 + -u_1 x_6 + x_1 x_6 + -u_7 x_7 == 0 \wedge -u_4 u_6 + -u_5 x_6 + u_6 x_6 + u_4 x_7 == 0 \Rightarrow \\ x_3 x_4 + -x_2 x_5 + -x_3 x_6 + x_5 x_6 + x_2 x_7 + -x_4 x_7 == 0)$$

This problem is equivalent to:

$$(2) \quad \neg \left( \exists_{x_1, x_2, x_3, x_4, x_5, x_6, x_7} (u_3 u_4 + -u_2 u_5 + -u_3 u_7 + u_5 u_7 + u_2 x_1 + -u_4 x_1 == 0 \wedge \\ u_5 x_2 + -u_4 x_3 == 0 \wedge -u_1 u_2 + u_1 x_2 + -u_3 x_2 + u_2 x_3 == 0 \wedge \\ x_1 x_4 + -u_7 x_5 == 0 \wedge -u_2 u_6 + -u_3 x_4 + u_6 x_4 + u_2 x_5 == 0 \wedge \\ u_1 u_7 + -u_1 x_6 + x_1 x_6 + -u_7 x_7 == 0 \wedge -u_4 u_6 + -u_5 x_6 + u_6 x_6 + u_4 x_7 == 0 \wedge \\ x_3 x_4 + -x_2 x_5 + -x_3 x_6 + x_5 x_6 + x_2 x_7 + -x_4 x_7 \neq 0) \right)$$

To remove the last inequality, we use the Rabinowitsch trick: Let  $v_0$  be a new variable. Then the problem becomes:

$$(3) \quad \neg \left( \exists_{x_1, x_2, x_3, x_4, x_5, x_6, x_7, v_0} (u_3 u_4 + -u_2 u_5 + -u_3 u_7 + u_5 u_7 + u_2 x_1 + -u_4 x_1 == 0 \wedge \\ u_5 x_2 + -u_4 x_3 == 0 \wedge -u_1 u_2 + u_1 x_2 + -u_3 x_2 + u_2 x_3 == 0 \wedge \\ x_1 x_4 + -u_7 x_5 == 0 \wedge -u_2 u_6 + -u_3 x_4 + u_6 x_4 + u_2 x_5 == 0 \wedge \\ u_1 u_7 + -u_1 x_6 + x_1 x_6 + -u_7 x_7 == 0 \wedge -u_4 u_6 + -u_5 x_6 + u_6 x_6 + u_4 x_7 == 0 \wedge \\ 1 + -v_0 (x_3 x_4 + -x_2 x_5 + -x_3 x_6 + x_5 x_6 + x_2 x_7 + -x_4 x_7) == 0) \right)$$

This statement is true iff the corresponding Gröbner basis is  $\{1\}$ .

The Gröbner bases is  $\{1\}$ .

Hence, the statement and the original assertion is true.

Statistics:

Time needed to compute the Gröbner bases: 0.42 Seconds.

---

## Was haben wir gemacht?

Die Algorithmik aus [einem Bereich der Mathematik](#) (z.B. Theorie der Gröbner-Basen für Polynome)

kann auf der "Meta-Ebene" [eines anderen Bereichs der Mathematik](#) (z.B. Koordinatengeometrie)

zur [Automatisierung des Beweisens](#) führen.

## Beispiel: Arithmetik

Algorithmik der letzten 40 Jahre

Algorithmik einen Stock höher: Automatisches  
Beweisen

Noch einen Stock höher: Automatisches Erfinden

## Durchgehendes Beispiel: Die Theorie der Gröbner-Basen

### Beispiel: Automatisches Erfinden und Beweisen der Theorie der Gröbner-Basen (BB 2005, ...)

[Begriff](#) des S-Polynoms

[Hauptsatz](#)

[Algorithmus](#) zur Konstruktion der Gröbner-Basen

---

## Was ich mit diesem Beispiel zeigen möchte:

Eine Kombination von

- (automatischem) [Beweisen](#)
- (automatischer) Anwendung von Formel-[Schemata](#) zum Erfinden
- und (automatischer) [Analyse von misslungenen](#) Beweisen

ergibt eine erstaunliche leistungsfähige Methode zur (automatischen) Erfindung von Algorithmen ("lazy thinking method", BB 2002).

Leistungsfähig: kann auch für nicht-triviale Probleme Algorithmen erfinden.

(Insbesondere leistungsfähig genug, um meine eigene Problemlösekraft zu ersetzen.)

---

## "Non-trivial"

Konstruktion von Gröbner-Basen:

- zur Zeit der Erfindung (1965, BB) wurde das Problem als "algorithmisch unlösbar" eingestuft
- zahlreiche nicht-triviale Anwendungen
- noch nicht durch andere Synthesemethoden synthetisiert.

---

## Das Problem der Algorithmen-Synthese (-Erfindung)

Gegeben eine Problemspezifikation P (eine prädikatenlogische Formel), finde einen Algorithmus A, sodass

$$\forall_x P[x, A[x]] .$$

Ein allgemeiner Algorithmen-Synthese-Algorithmus kann nicht existieren, aber ...

---

## Literatur

Reichhaltige Literatur, siehe Übersichtsartikel:

[Basin et al. 2004] D. Basin, Y. Deville, P. Flener, A. Hamfelt, J. F. Nilsson. Synthesis of Programs in Computational Logic. In: M. Bruynooghe, K. K. Lau (eds.), Program Development in Computational Logic, Lecture Notes in Computer Science, Vol. 3049, Springer, 2004, pp. 30-65.

"Lazy thinking" gehört zu den "scheme-based" Methoden.

## Algorithmen-Synthese durch "Lazy Thinking" (BB 2002)

Gegen: Problemspezifikation P. Find: Algorithmus A für P.

- ♣ Wir nehmen an, dass wir "vollständiges" Wissen über alle Begriffe, die in P vorkommen, haben.
- ♣ Betrachte fundamentale Ideen ("Algorithmen-Schemata") wie man Algorithmen A aus Unter-Algorithmen B, C, ... komponiert.

Probiere ein Schema nach dem anderen:

- ♣ Für das gewählte Schema A, versuche zu beweisen  $\forall_x P[x, A[x]]$ : Aus dem misslungenen Beweis konstruiere Specifications für die Subalgorithmen B, ...

## Automatische Erfindung von hinreichenden Specifications für die Unteralgorithmen

Eine einfache (aber erstaunlich starke) Regel (B ... ein unbekannter Unteralgorithmus):

Sammele die temporären Annahmen  $T[x_0, \dots, A[\ ], \dots]$

und temporären Ziele  $G[x_0, \dots, B[\dots, A[\ ] \dots]]$

und produziere die Spezifikation

$$\forall_{x, \dots, y, \dots} (T[x, \dots, Y, \dots] \Rightarrow G[y, \dots, B[Y]]).$$

Details: siehe [BB 2003] und die Beispiele unten.

## The method works well on simple problems like sorting

See [Buchberger 2003].

For example, using the divide-and-conquer scheme

$$\forall_{N,S,M,L,R} \text{Divide-and-Conquer}[A, S, M, L, R] \Leftrightarrow \forall_x \left( A[x] = \begin{cases} S[x] & \Leftarrow \text{is-trivial-tuple}[x] \\ M[\text{sorted}[L[x]], \text{sorted}[R[x]]] & \Leftarrow \text{otherwise} \end{cases} \right)$$

the method finds (in approx. 2 minutes on a laptop) that all subalgorithms **S**, **M**, **L**, **R** satisfying the following specifications make **A** a correct sorting algorithm:

$$\forall_x (\text{is-trivial-tuple}[x] \Rightarrow \mathbf{S}[x] = x)$$

$$\forall_{y,z} \left( \begin{array}{l} \text{is-sorted}[y] \\ \text{is-sorted}[z] \end{array} \Rightarrow \begin{array}{l} \text{is-sorted}[\mathbf{M}[y, z]] \\ \mathbf{M}[y, z] \approx (y \prec z) \end{array} \right)$$

$$\forall_x (\mathbf{L}[x] \approx \mathbf{R}[x] \approx x)$$

Note: the specifications generated are not only sufficient but natural !

Now we can continue, recursively, with synthesizing - or retrieving - algorithms **S**, **M**, **L**, **R** satisfying the above specifications.

## Wie weit trägt die Methode?

Erfolgreiche Synthese eines Algorithmus zur Konstruktion von Gröbner-Basen:

B. Buchberger. Towards the Automated Synthesis of a Gröbner Bases Algorithm. RACSAM (Review of the Royal Spanish Academy of Science), Vol. 98/1, 2005, pp. 65-75.

## Das Problem der Konstruktion von Gröbner-Basen

Finde Algorithmus **Gb** sodass

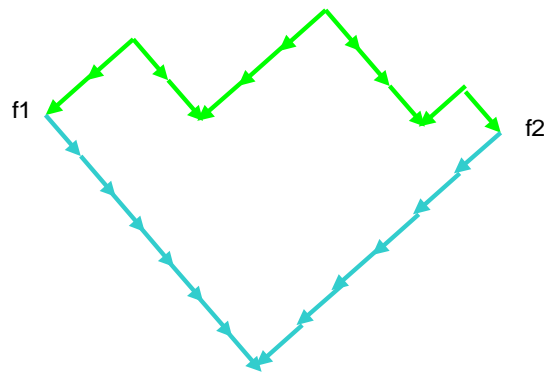
$$\forall_{\text{is-finite}[F]} \left( \begin{array}{l} \text{is-finite}[\mathbf{Gb}[F]] \\ \text{is-Gröbner-basis}[\mathbf{Gb}[F]] \\ \text{ideal}[F] = \text{ideal}[\mathbf{Gb}[F]]. \end{array} \right)$$

$$\text{is-Gröbner-basis}[G] \Leftrightarrow \text{is-confluent}[\rightarrow_G].$$

$\rightarrow_G$  ... ein Divisionsschritt.

## Konfluenz der Division $\rightarrow_G$

```
is-confluent[  $\rightarrow$  ] :  $\Leftrightarrow \forall_{f_1, f_2} (f_1 \leftrightarrow^* f_2 \Rightarrow f_1 \downarrow^* f_2)$ 
```



## Die entscheidende Idee in der Gröbner-Basen-Theorie (BB 1965)

Es genügt, die folgenden (endlich vielen) Polynome anzuschauen,

```
least-common-multiple[lp[g1], lp[g2]]
```

( $g_1$  und  $g_2$  in der gegebenen endlichen Menge von Polynomen  $F$ .)

## Die entscheidende Frage also

Kann "Lazy Thinking" *automatisch diese Idee produzieren* (und als korrekt erweisen)?

Starten wir also mit "Lazy Thinking" für die Eingabespezifikation.

## Vorhandenes Wissen

$$h1 \rightarrow_G h2 \Rightarrow p . h1 \rightarrow_G p . h2$$

etc.

## Wähle ein Algorithmen-Schema

Zum Beispiel: ein Schema, das für alle mathematische Strukturen anwendbar ist, in welcher es eine "Reduktionsoperation"

$rd[f, g]$  (Result der "Reduktion von  $f$  mit  $g$ ") gibt, die satisfying  $rd[f, g] \leq f$  erfüllt.

$$\forall_{A, lc, df} \text{pair-completion}[A, lc, df] \Leftrightarrow$$

$$\begin{aligned} & \forall_F A[F] = A[F, \text{pairs}[F]] \\ & \forall_F A[F, \langle \rangle] = F \\ & \forall_{F, g1, g2, \bar{p}} A[F, \langle \langle g1, g2 \rangle, \bar{p} \rangle] = \\ & \quad \text{where } [f = lc[g1, g2], \\ & \quad \quad h1 = \text{trd}[rd[f, g1], F], h2 = \text{trd}[rd[f, g2], F], \\ & \quad \left\{ \begin{array}{ll} A[F, \langle \bar{p} \rangle] & \Leftarrow h1 = h2 \\ A[F - df[h1, h2], \\ \langle \bar{p} \rangle \times \langle \langle F_k, df[h1, h2] \rangle_{k=1, \dots, |F|} \rangle \end{array} \right. & \Leftarrow \text{otherwise} \end{array} \end{aligned}$$

## Beginne nun den (automatischen) Korrektheitsbeweis

Mit der heutigen automatischen Beweis-Technologie im Theorema-System [BB et al. 1995, ...] können solche Beweise automatisch gemacht werden. (PhD Thesis meines Studenten A. Craciun.)

## Details

Zunächst kann bewiesen werden, dass für die am Ende des Algorithmus vorliegende Basis  $G$  Folgendes gilt:

$$\forall_{g1, g2 \in G} \left( \text{where } [f = \text{lc}[g1, g2], h1 = \text{trd}[\text{rd}[f, g1], G], \right. \\ \left. h2 = \text{trd}[\text{rd}[f, g2], G], \bigvee \left\{ \begin{array}{l} h1 = h2 \\ \text{df}[h1, h2] \in G \end{array} \right\} \right).$$

Wir versuchen nun zu beweisen:

```
is-finite[G],
ideal[F] = ideal[G],
is-Gröbner-basis[G],
i.e. is-Church-Rosser[→G].
```

(Hier beschränken wir uns auf die dritte, die wichtigste Eigenschaft).

## Using Available Knowledge

Using Newman's lemma and some elementary properties it can be shown that it is sufficient to prove

$$\text{is-Church-Rosser}[\rightarrow_G] \Leftrightarrow \forall_p \forall_{f1, f2} \left( \left( \left\{ \begin{array}{l} p \rightarrow f1 \\ p \rightarrow f2 \end{array} \right\} \Rightarrow f1 \downarrow^* f2 \right) \right).$$

Newman's lemma (1942):

$$\text{is-Church-Rosser}[\rightarrow] \Leftrightarrow \forall_{f, f1, f2} \left( \left( \left\{ \begin{array}{l} f \rightarrow f1 \\ f \rightarrow f2 \end{array} \right\} \Rightarrow f1 \downarrow^* f2 \right) \right).$$

## The (Automated) Proof Attempt

Let now the power product  $p$  and the polynomials  $f1, f2$  be arbitrary but fixed and assume

$$\left\{ \begin{array}{l} p \rightarrow_G f1 \\ p \rightarrow_G f2. \end{array} \right.$$

We have to find a polynomial  $g$  such that

$$\begin{aligned} f_1 &\rightarrow_G^* g, \\ f_2 &\rightarrow_G^* g. \end{aligned}$$

From the assumption we know that there exist polynomials  $g_1$  and  $g_2$  in  $G$  such that

$$\begin{aligned} \text{lp}[g_1] &| p, \\ f_1 &= \text{rd}[p, g_1], \\ \text{lp}[g_2] &| p, \\ f_2 &= \text{rd}[p, g_2]. \end{aligned}$$

From the final situation in the algorithm scheme we know that for these  $g_1$  and  $g_2$

$$\bigvee \begin{cases} h_1 = h_2 \\ \text{df}[h_1, h_2] \in G, \end{cases}$$

where

$$\begin{aligned} h_1 &:= \text{trd}[f_1', G], f_1' := \text{rd}[\text{lc}[g_1, g_2], g_1], \\ h_2 &:= \text{trd}[f_2', G], f_2' := \text{rd}[\text{lc}[g_1, g_2], g_2]. \end{aligned}$$

## Case $h_1=h_2$

$$\begin{aligned} \text{lc}[g_1, g_2] &\rightarrow_{g_1} \text{rd}[\text{lc}[g_1, g_2], g_1] \rightarrow_G^* \text{trd}[\text{rd}[\text{lc}[g_1, g_2], g_1], G] = \\ &\text{trd}[\text{rd}[\text{lc}[g_1, g_2], g_2], G] \leftarrow_G^* \text{rd}[\text{lc}[g_1, g_2], g_2] \leftarrow_{g_2} \text{lc}[g_1, g_2]. \end{aligned}$$

(Note that here we used the requirements that  $\text{lc}[g_1, g_2]$  is reducible w.r.t.  $g_1$  and  $g_2$ . The other cases are easy.)

Hence, by elementary properties of polynomial reduction,

$$\begin{aligned} \forall_{a, q} ( a q \text{lc}[g_1, g_2] &\rightarrow_{g_1} a q \text{rd}[\text{lc}[g_1, g_2], g_1] \rightarrow_G^* a q \text{trd}[\text{rd}[\text{lc}[g_1, g_2], g_1], G] = \\ &a q \text{trd}[\text{rd}[\text{lc}[g_1, g_2], g_2], G] \leftarrow_G^* a q \text{rd}[\text{lc}[g_1, g_2], g_2] \leftarrow_{g_2} a q \text{lc}[g_1, g_2] ). \end{aligned}$$

Now we are stuck in the proof.

## Wir sind nun auf einem toten Punkt im Beweis angelangt

Durch Anwenden der obigen Regel zur Extraktion von Spezifikationen für Unteralgorithmen aus misslungenen Beweisen, sieht man, dass man mit dem Beweis fortfahren könnte, wenn  $lc[g1, g2]$  die folgende Bedingung erfüllte:

$$\forall_{p, g1, g2} \left( \left( \left\{ \begin{array}{l} lp[g1] \mid p \\ lp[g2] \mid p \end{array} \right\} \Rightarrow \left( \exists_{a, q} (p = a \ q \ lc[g1, g2]) \right) \right) \right), \quad (lc \text{ requirement})$$

With such an  $lc$ , we then would have

$$p \rightarrow_{g1} rd[p, g1] = a \ q \ rd[lc[g1, g2], g1] \rightarrow_G^* a \ q \ trd[rd[lc[g1, g2], g1], G] = a \ q \ trd[rd[lc[g1, g2], g2], G] \leftarrow_G^* a \ q \ rd[lc[g1, g2], g2] = rd[p, g2] \leftarrow_{g2} p$$

and, hence,

$$f1 \rightarrow_G^* a \ q \ trd[rd[lc[g1, g2], g1], G],$$

$$f2 \rightarrow_G^* a \ q \ trd[rd[lc[g1, g2], g1], G],$$

i.e. we would have found a suitable  $g$ .

## Zusammenfassung der (automatisch erzeugten) hinreichenden Spezifikationen des Unteralgorithmus $lc$

$$\forall_{p, g1, g2} \left( \left( \left\{ \begin{array}{l} lp[g1] \mid p \\ lp[g2] \mid p \end{array} \right\} \Rightarrow (lc[g1, g2] \mid p) \right) \right),$$

$$lp[g1] \mid lc[g1, g2], \\ lp[g2] \mid lc[g1, g2].$$

So ein Unteralgorithmus  $lc$  can aber nun mit elementarsten Mitteln gefunden werden (Mittelschulmathematik!)

## Ein geeigneter Algorithmus lc

$$lc_p[g1, g2] = lcm[lp[g1], lp[g2]]$$

ist ein geeigneter Unteralgorithmus.

Heureka! Die wesentliche Funktion lc wurde automatisch synthetisiert.

## Case $h1 \neq h2$

In this case,  $df[h1, h2] \in G$ :

In this part of the proof we are basically stuck right at the beginning.

We can try to reduce this case to the first case, which would generate the following requirement

$$\forall_{h1, h2} (h1 \downarrow_{\{df[h1, h2]\}} * h2) \quad (\text{df requirement}).$$

## Looking to the Knowledge Base for a Suitable df

(Looking to the knowledge base of elementary properties of polynomial reduction, it is now easy to find a function df that satisfies (df requirement), namely

$$df[h1, h2] = h1 - h2,$$

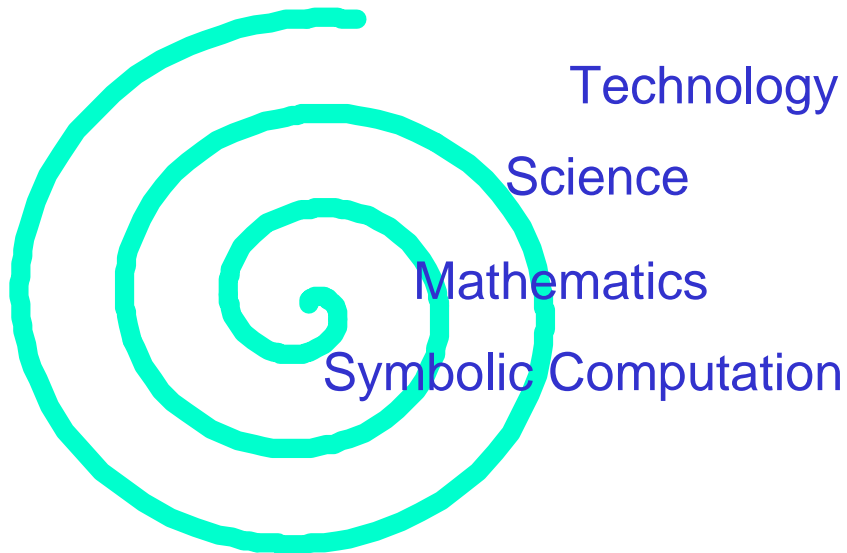
because, in fact,

$$\forall_{f, g} (f \downarrow_{\{f-g\}} * g).$$

Eureka! The function df (the "completion" function) in the critical pair / completion algorithm scheme has been "automatically" synthesized!

---

## Konklusion



Symbolic Computation ist Selbstanwendung der Mathematik.

Die gesamte Mathematik ist in gewisser Weise Symbolic Computation.

Die Fortentwicklung der Mathematik passiert durch permanente Entfaltung höherer Stufen durch Anwendung niederer Stufen auf der jeweiligen Meta-Ebene.

- **Mathematik ist Automatisierung**
  
- **Es gibt keine Grenzen für die Automatisierung in der Mathematik**  
("Mathematik ist Gödelsch")
  
- **Konsequenzen** für mathematische Forschung, Anwendung, Lehre
  
- **Mathematik und Meditation** spannen die diametralen Bewegungen des Bewusstseins auf

## References

### ■ On Gröbner Bases

[Buchberger 1970]

B. Buchberger. Ein algorithmisches Kriterium für die Lösbarkeit eines algebraischen Gleichungssystems (An Algorithmical Criterion for the Solvability of Algebraic Systems of Equations). *Aequationes mathematicae* 4/3, 1970, pp. 374-383. (English translation in: [Buchberger, Winkler 1998], pp. 535 -545.)  
Published version of the PhD Thesis of B. Buchberger, University of Innsbruck, Austria, 1965.

[Buchberger 1998]

B. Buchberger. Introduction to Gröbner Bases. In: [Buchberger, Winkler 1998], pp.3-31.

[Buchberger, Winkler, 1998]

B. Buchberger, F. Winkler (eds.). Gröbner Bases and Applications, Proceedings of the International Conference "33 Years of Gröbner Bases", 1998, RISC, Austria, London Mathematical Society Lecture Note Series, Vol. 251, Cambridge University Press, 1998.

[Becker, Weispfenning 1993]

T. Becker, V. Weispfenning. Gröbner Bases: A Computational Approach to Commutative Algebra, Springer, New York, 1993.

## ■ On Mathematical Knowledge Management

B. Buchberger, G. Gonnet, M. Hazewinkel (eds.)

Mathematical Knowledge Management.

Special Issue of Annals of Mathematics and Artificial Intelligence, Vol. 38, No. 1-3, May 2003, Kluwer Academic Publisher, 232 pages.

A. Asperti, B. Buchberger, J.H. Davenport (eds.)

Mathematical Knowledge Management.

Proceedings of the Second International Conference on Mathematical Knowledge Management (MKM 2003), Bertinoro, Italy, Feb. 16-18, 2003, Lecture Notes in Computer Science, Vol. 2594, Springer, Berlin-Heidelberg-New York, 2003, 223 pages.

A. Asperti, G. Bancerek, A. Trybulec (eds.).

Proceedings of the Third International Conference on Mathematical Knowledge Management, MKM 2004, Bialowieza, Poland, September 19-21, 2004, Lecture Notes in Computer Science, Vol. 3119, Springer, Berlin-Heidelberg-New York, 2004

## ■ On Theorema

[Buchberger et al. 2000]

B. Buchberger, C. Dupre, T. Jebelean, F. Kriftner, K. Nakagawa, D. Vasaru, W. Windsteiger. The Theorema Project: A Progress Report. In: M. Kerber and M. Kohlhase (eds.), Symbolic Computation and Automated Reasoning (Proceedings of CALCULEMUS 2000, Symposium on the Integration of Symbolic Computation and Mechanized Reasoning, August 6-7, 2000, St. Andrews, Scotland), A.K. Peters, Natick, Massachusetts, ISBN 1-56881-145-4, pp. 98-113.

## ■ On Theory Exploration and Algorithm Synthesis

[Buchberger 2000]

B. Buchberger. Theory Exploration with *Theorema*.

Analele Universitatii Din Timisoara, Ser. Matematica-Informatica, Vol. XXXVIII, Fasc.2, 2000, (Proceedings of SYNASC 2000, 2nd International Workshop on Symbolic and Numeric Algorithms in Scientific Computing, Oct. 4-6, 2000, Timisoara, Rumania, T. Jebelean, V. Negru, A. Popovici eds.), ISSN 1124-970X, pp. 9-32.

[Buchberger 2003]

B. Buchberger. Algorithm Invention and Verification by Lazy Thinking.

In: D. Petcu, V. Negru, D. Zaharie, T. Jebelean (eds), Proceedings of SYNASC 2003 (Symbolic and Numeric Algorithms for Scientific Computing, Timisoara, Romania, October 1-4, 2003), Mirton Publishing, ISBN 973-661-104-3, pp. 2-26.

[Buchberger, Craciun 2003]

B. Buchberger, A. Craciun. Algorithm Synthesis by Lazy Thinking: Examples and Implementation in Theorema. in: Fairouz Kamareddine (ed.), Proc. of the Mathematical Knowledge Management Workshop, Edinburgh, Nov. 25, 2003, Electronic Notes on Theoretical Computer Science, volume dedicated to the MKM 03 Symposium, Elsevier, ISBN 044451290X, to appear.

[Buchberger 2005]

B. Buchberger.

Towards the Automated Synthesis of a Gröbner Bases Algorithm.

RACSAM (Review of the Royal Spanish Academy of Science), Vol. 98/1, 2005, pp. 65-75.