

# Algorithmische Beweisverfahren

## Das Ende der Mathematik?

Bruno Buchberger

RISC und RICAM

JK Symposium, 20. April 2005

---

Dem Andenken an Professor Hans Knapp gewidmet

### Übersicht:

**Algorithmische Beweisverfahren**

**Einige Beispiele (in Theorema)**

**Diskussion**

Übersicht:

## Algorithmische Beweisverfahren

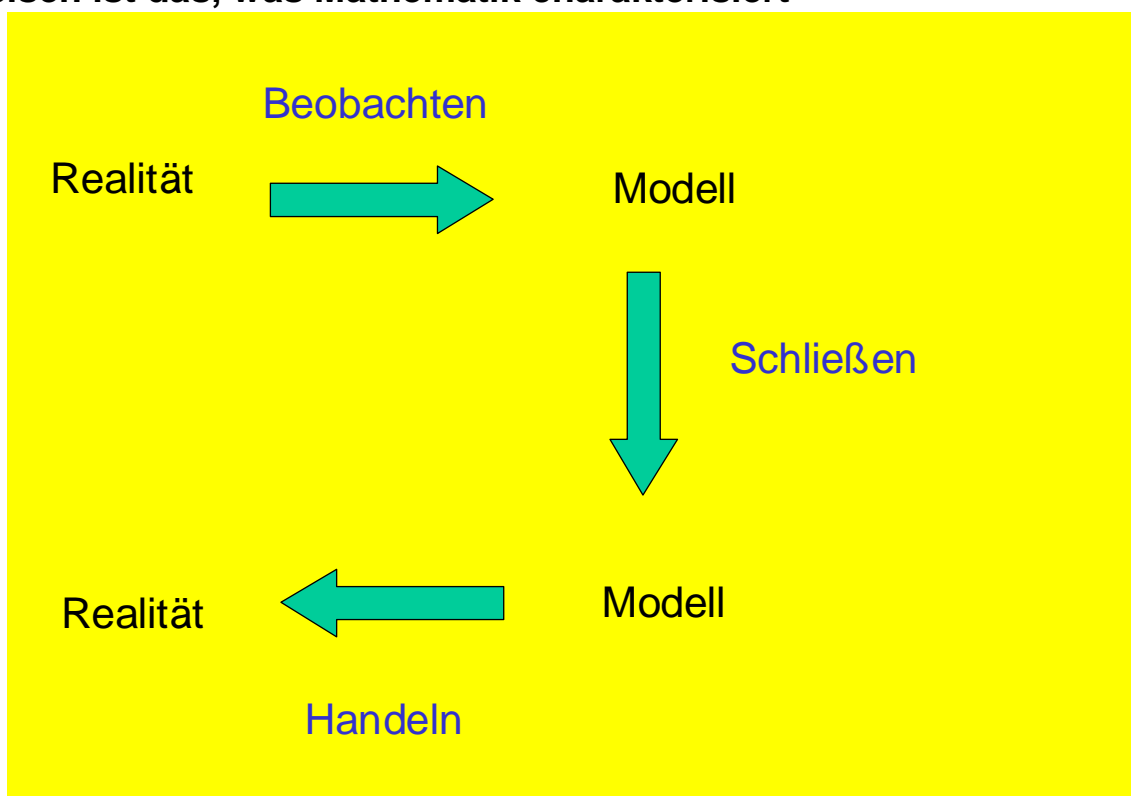
Einige Beispiele (in Theorema)

Diskussion

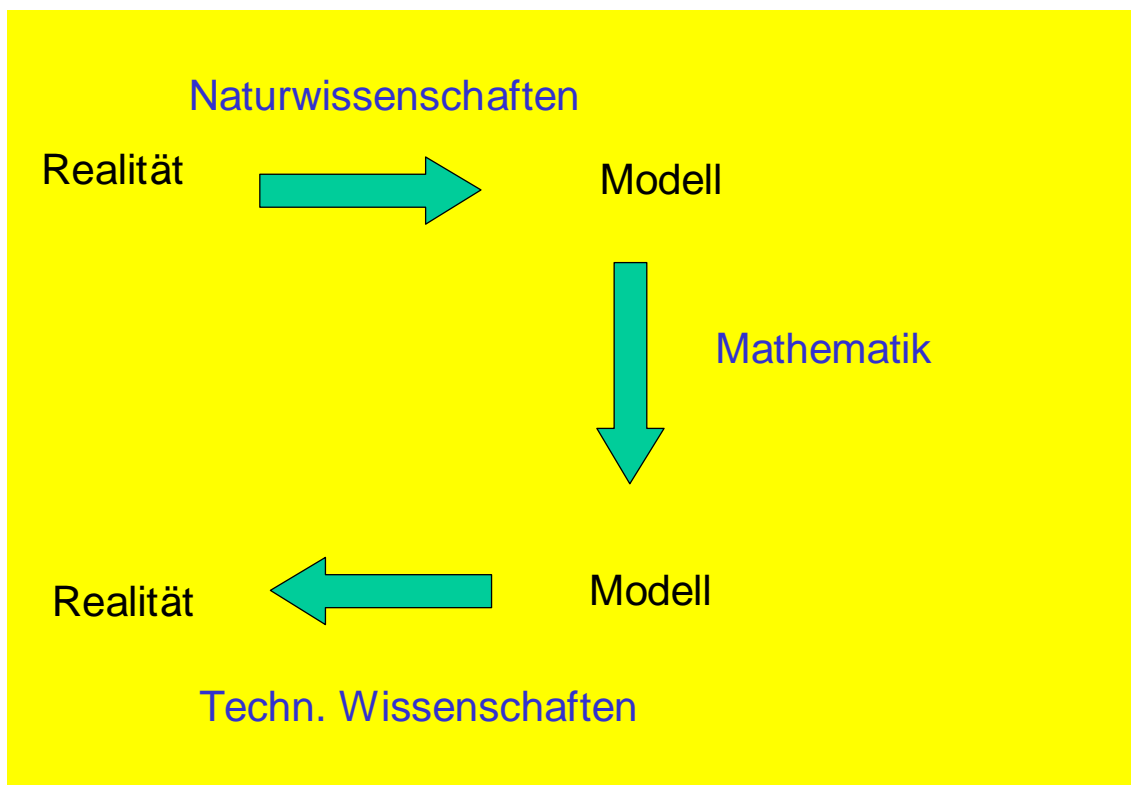


4 of 29

**Beweisen ist das, was Mathematik charakterisiert**



5 of 29



6 of 29

### Das Problem des Beweisen:

Gegeben: Eine Aussage A und Aussagen B.

Frage: Folgt A aus B ? ("Wahrheits-Entscheider")

Gesucht: Eine korrekte Schlusskette, die A aus B herleitet. ("Beweis-Finder")

□

Eine sehr viel schwächere Form des Beweisproblems:

Gegeben: Eine Aussage A, Aussagen B und eine Schlusskette P.

Frage: Ist P eine korrekte Schlusskette, die A aus B herleitet. ("Beweis-Prüfer")

7 of 29

### Beispiel:

Aussage A:

$$(\text{limit}[f, a] \wedge \text{limit}[g, b]) \Rightarrow \text{limit}[f + g, a + b]$$

Aussagen B:

$$\text{limit}[f, a] \Leftrightarrow \forall_{\epsilon > 0} \exists_{N \in \mathbb{N}} \forall_{n \geq N} |f[n] - a| < \epsilon$$

$$(f + g)[x] = f[x] + g[x]$$

etc.



8 of 29

## Automatische Beweisprüfer: Z.B. Mizar

2000 Definitionen, 30.000 automatisch geprüfte Sätze.

Im Rahmen der Tarski-Grothendieck Mengenlehre (in Prädikatenlogik erster Stufe).

Von Mengenlehre bis Banach-Räume ...

Ca. 750 Artikel in Journal of Formal Mathematics.

A. Trybulec et al., University Bialystok, Polen, seit ca. 1976 (1989).



9 of 29

## Automatische Wahrheitsentscheider, Beweisfinder:

Ist das möglich?

Bis zu welchem Grad?

Ist das wünschenswert?

Braucht es dann noch Mathematiker?

Braucht es dann noch Mathematik?

Wird sich die Art, Mathematik zu betreiben, ändern?



10 of 29

## Übersicht:

### Algorithmische Beweisverfahren

### Einige Beispiele (in Theorema)

### Diskussion

⏪ ⏩ ⏴ ⏵

11 of 29

---

## Das Theorema-Projekt

Ziel: Computer-Unterstützung des Explorations-Prozesses mathematischer Theorien.

Die Theorema-Sprache ist eine Version der Prädikatenlogik.

Sie enthält eine Programmiersprache.

Die Theorema-Gruppe: B. B., T. Jebelean, W. Windsteiger, T. Kutsia, M. Rosenkranz, F. Piroi, G. Regensburger, M. Giese und PhD- Studenten.

Viele andere Gruppen: ACL (Boyer, J.S. Moore et al.), Coq, Elf (F. Pfenning), IMPS (W. Farmer et al.), Isabelle (L. Paulson), Nuprl (R. Constable et al.), Omega (J. Siekmann et al.), Otter (L. Wos) u.v.a.

⏪ ⏩ ⏴ ⏵

12 of 29

---

## Beispiel: Beweise in elementarer Analysis mit PCS-Beweiser

PCS-Methode: B.B. 2000, D. Vasaru 2000;

### □ Zu beweisen

```
Proposition["limit of sum", any[f, a, g, b],
  (limit[f, a] ^ limit[g, b]) => limit[f + g, a + b]]
```

▫ Wissen

```
Definition["limit:", any[f, a],
  limit[f, a] ⇔  $\forall_{\epsilon > 0} \exists_{N \in \mathbb{N}} \forall_{n \geq N} |f[n] - a| < \epsilon$ 
```

```
Definition["+":, any[f, g, x],
  (f + g)[x] = f[x] + g[x]
```

```
Lemma["|+|", any[x, y, a, b, δ, ε],
  (|(x + y) - (a + b)| < (δ + ε)) ⇐ (|x - a| < δ ∧ |y - b| < ε)
```

```
Lemma["max", any[m, M1, M2],
  m ≥ max[M1, M2] ⇒ (m ≥ M1 ∧ m ≥ M2)
```

▫ Wissen zusammenfassen

```
Theory["limit",
  Definition["limit:"]
  Definition["+":]
  Lemma["|+|"]
  Lemma["max"]
]
```

■ Aufruf des Beweisers

```
Prove[Proposition["limit of sum"], using → Theory["limit"], by → PCS]
```

```
- ProofObject -
```

## Beispiel: Beweise in Koordinaten-Geometrie mit Gröbner-Basen

Das Prinzip der Methode (B.B. 1965; B.B., S. Stifter, B. Kutzler 1986; D. Kapur 1986; F. Winkler 1990, J. Robu 2003):

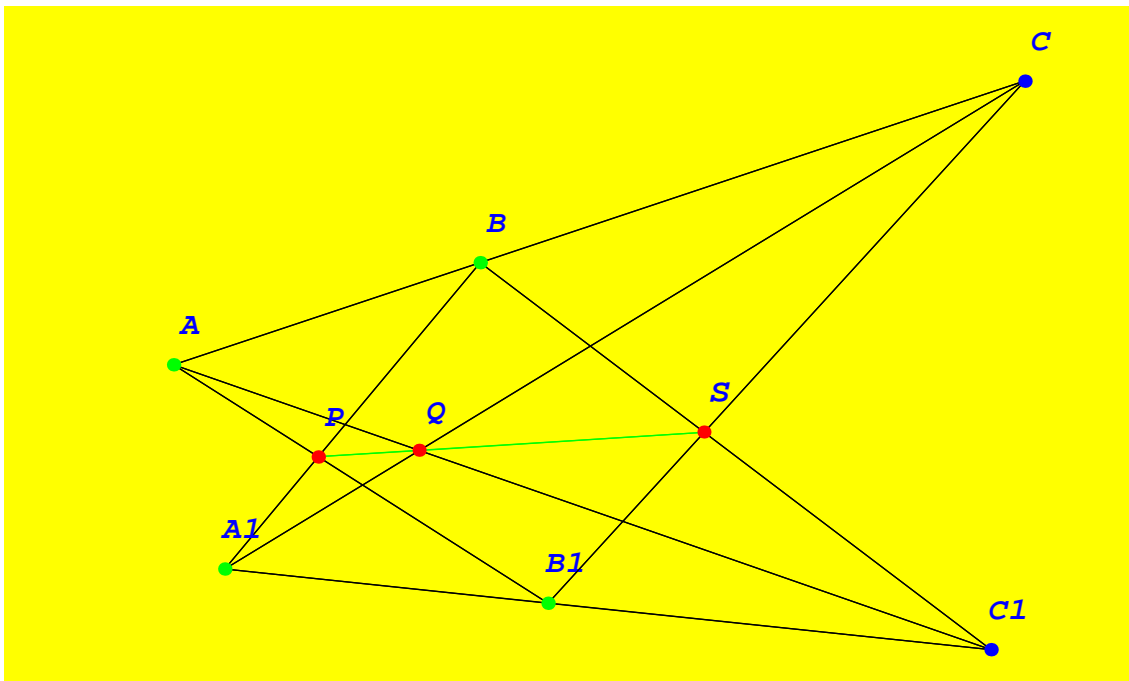
```
Formula["Test", any[x, y],
  ( (x2 y - 3 x) ≠ 0) ∨ ((x y + x + y) ≠ 0) ∨
    ((x2 y + 3 x = 0) ∨ ((-2 x2 + -7 x y + x2 y + x3 y + -2 y2 + -2 x y2 + 2 x2
    ∧ ((x2 + -x y + x2 y + -2 y2 + -2 x y2) = 0))
]
```

```
Prove[Formula["Test"], using → {}, by → GroebnerBasesProver]
```

```
- ProofObject -
```

### ■ Beispiel einer Anwendung: Pappus Theorem

#### □ Graphisch



Seien A, B, C und A<sub>1</sub>, B<sub>1</sub>, C<sub>1</sub> auf zwei Geraden und seien  $P = AB_1 \cap A_1B$ ,  $Q = AC_1 \cap A_1C$ ,  $S = BC_1 \cap B_1C$ . Dann sind P, Q, und S kollinear.

▫ In der Sprache der Geometrie (Dissertation Dr. Judit Robu)

```
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]]
```

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

▫ Der automatisch erzeugte Beweis

Prove:

(Proposition (Pappus))

$$\forall_{A, B, A1, B1, C, C1, P, Q, S} (\text{point}[A, B, A1, B1] \wedge \text{pon}[C, \text{line}[A, B]] \wedge \text{pon}[C1, \text{line}[A1, B1]] \wedge \text{inter}[P, \text{line}[A, B1], \text{line}[A1, B]] \wedge \text{inter}[Q, \text{line}[A, C1], \text{line}[A1, C]] \wedge \text{inter}[S, \text{line}[B, C1], \text{line}[B1, C]] \Rightarrow \text{collinear}[P, Q, S])$$

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  $\{ \}$  and  $\{B, C\}$  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 given construction is:

$$\begin{aligned}
(1) \quad & \forall_{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7} \left( \mathbf{u}_3 \mathbf{u}_4 + -\mathbf{u}_2 \mathbf{u}_5 + -\mathbf{u}_3 \mathbf{u}_7 + \mathbf{u}_5 \mathbf{u}_7 + \mathbf{u}_2 \mathbf{x}_1 + -\mathbf{u}_4 \mathbf{x}_1 = 0 \wedge \right. \\
& \mathbf{u}_5 \mathbf{x}_2 + -\mathbf{u}_4 \mathbf{x}_3 = 0 \wedge -\mathbf{u}_1 \mathbf{u}_2 + \mathbf{u}_1 \mathbf{x}_2 + -\mathbf{u}_3 \mathbf{x}_2 + \mathbf{u}_2 \mathbf{x}_3 = 0 \wedge \\
& \mathbf{x}_1 \mathbf{x}_4 + -\mathbf{u}_7 \mathbf{x}_5 = 0 \wedge -\mathbf{u}_2 \mathbf{u}_6 + -\mathbf{u}_3 \mathbf{x}_4 + \mathbf{u}_6 \mathbf{x}_4 + \mathbf{u}_2 \mathbf{x}_5 = 0 \wedge \\
& \mathbf{u}_1 \mathbf{u}_7 + -\mathbf{u}_1 \mathbf{x}_6 + \mathbf{x}_1 \mathbf{x}_6 + -\mathbf{u}_7 \mathbf{x}_7 = 0 \wedge \\
& -\mathbf{u}_4 \mathbf{u}_6 + -\mathbf{u}_5 \mathbf{x}_6 + \mathbf{u}_6 \mathbf{x}_6 + \mathbf{u}_4 \mathbf{x}_7 = 0 \Rightarrow \\
& \left. \mathbf{x}_3 \mathbf{x}_4 + -\mathbf{x}_2 \mathbf{x}_5 + -\mathbf{x}_3 \mathbf{x}_6 + \mathbf{x}_5 \mathbf{x}_6 + \mathbf{x}_2 \mathbf{x}_7 + -\mathbf{x}_4 \mathbf{x}_7 = 0 \right)
\end{aligned}$$

This problem is equivalent to:

$$\begin{aligned}
(2) \quad & \neg \left( \exists_{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7} \left( \mathbf{u}_3 \mathbf{u}_4 + -\mathbf{u}_2 \mathbf{u}_5 + -\mathbf{u}_3 \mathbf{u}_7 + \mathbf{u}_5 \mathbf{u}_7 + \mathbf{u}_2 \mathbf{x}_1 + -\mathbf{u}_4 \mathbf{x}_1 = \right. \right. \\
& 0 \wedge \mathbf{u}_5 \mathbf{x}_2 + -\mathbf{u}_4 \mathbf{x}_3 = 0 \wedge -\mathbf{u}_1 \mathbf{u}_2 + \mathbf{u}_1 \mathbf{x}_2 + -\mathbf{u}_3 \mathbf{x}_2 + \mathbf{u}_2 \mathbf{x}_3 = 0 \wedge \\
& \mathbf{x}_1 \mathbf{x}_4 + -\mathbf{u}_7 \mathbf{x}_5 = 0 \wedge -\mathbf{u}_2 \mathbf{u}_6 + -\mathbf{u}_3 \mathbf{x}_4 + \mathbf{u}_6 \mathbf{x}_4 + \mathbf{u}_2 \mathbf{x}_5 = 0 \wedge \\
& \mathbf{u}_1 \mathbf{u}_7 + -\mathbf{u}_1 \mathbf{x}_6 + \mathbf{x}_1 \mathbf{x}_6 + -\mathbf{u}_7 \mathbf{x}_7 = 0 \wedge \\
& -\mathbf{u}_4 \mathbf{u}_6 + -\mathbf{u}_5 \mathbf{x}_6 + \mathbf{u}_6 \mathbf{x}_6 + \mathbf{u}_4 \mathbf{x}_7 = 0 \wedge \\
& \left. \left. \mathbf{x}_3 \mathbf{x}_4 + -\mathbf{x}_2 \mathbf{x}_5 + -\mathbf{x}_3 \mathbf{x}_6 + \mathbf{x}_5 \mathbf{x}_6 + \mathbf{x}_2 \mathbf{x}_7 + -\mathbf{x}_4 \mathbf{x}_7 \neq 0 \right) \right)
\end{aligned}$$

To remove the last inequality, we use the well-known Rabinowitsch trick. Let  $\mathbf{v}_0$  be a new variable. Then the problem becomes:

$$\begin{aligned}
(3) \quad & \neg \left( \exists_{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7, \mathbf{v}_0} \left( \mathbf{u}_3 \mathbf{u}_4 + \right. \right. \\
& -\mathbf{u}_2 \mathbf{u}_5 + -\mathbf{u}_3 \mathbf{u}_7 + \mathbf{u}_5 \mathbf{u}_7 + \mathbf{u}_2 \mathbf{x}_1 + -\mathbf{u}_4 \mathbf{x}_1 = 0 \wedge \\
& \mathbf{u}_5 \mathbf{x}_2 + -\mathbf{u}_4 \mathbf{x}_3 = 0 \wedge -\mathbf{u}_1 \mathbf{u}_2 + \mathbf{u}_1 \mathbf{x}_2 + -\mathbf{u}_3 \mathbf{x}_2 + \mathbf{u}_2 \mathbf{x}_3 = 0 \wedge \\
& \mathbf{x}_1 \mathbf{x}_4 + -\mathbf{u}_7 \mathbf{x}_5 = 0 \wedge -\mathbf{u}_2 \mathbf{u}_6 + -\mathbf{u}_3 \mathbf{x}_4 + \mathbf{u}_6 \mathbf{x}_4 + \mathbf{u}_2 \mathbf{x}_5 = 0 \wedge \\
& \mathbf{u}_1 \mathbf{u}_7 + -\mathbf{u}_1 \mathbf{x}_6 + \mathbf{x}_1 \mathbf{x}_6 + -\mathbf{u}_7 \mathbf{x}_7 = 0 \wedge \\
& -\mathbf{u}_4 \mathbf{u}_6 + -\mathbf{u}_5 \mathbf{x}_6 + \mathbf{u}_6 \mathbf{x}_6 + \mathbf{u}_4 \mathbf{x}_7 = 0 \wedge \\
& \left. \left. 1 + -\mathbf{v}_0 (\mathbf{x}_3 \mathbf{x}_4 + -\mathbf{x}_2 \mathbf{x}_5 + -\mathbf{x}_3 \mathbf{x}_6 + \mathbf{x}_5 \mathbf{x}_6 + \mathbf{x}_2 \mathbf{x}_7 + -\mathbf{x}_4 \mathbf{x}_7) = 0 \right) \right)
\end{aligned}$$

To prove this statement we have to compute the Gröbner bases of the above polynomials.

The polynomials of the Gröbner bases are:  $\{1\}$

As the obtained Gröbner bases is  $\{1\}$  the statement is generically true.

Statistics:

The length of the Gröbner bases is 1 .

Time needed to compute the Gröbner bases: 0 . 42 Seconds.

## ■ Beispiel einer Anwendung: Butterfly Theorem

### □ Textbuch-Formulierung

$A, B, C$  and  $D$  are four points on circle  $(O)$ .  $E$  is the intersection of  $AC$  and  $BD$ . Through  $E$  draw a line perpendicular to  $OE$ , meeting  $AD$  at  $F$  and  $BC$  at  $G$ . Show that  $\overline{FE} = \overline{GE}$ .

### □ Sitzung mit dem Geo-Beweiser

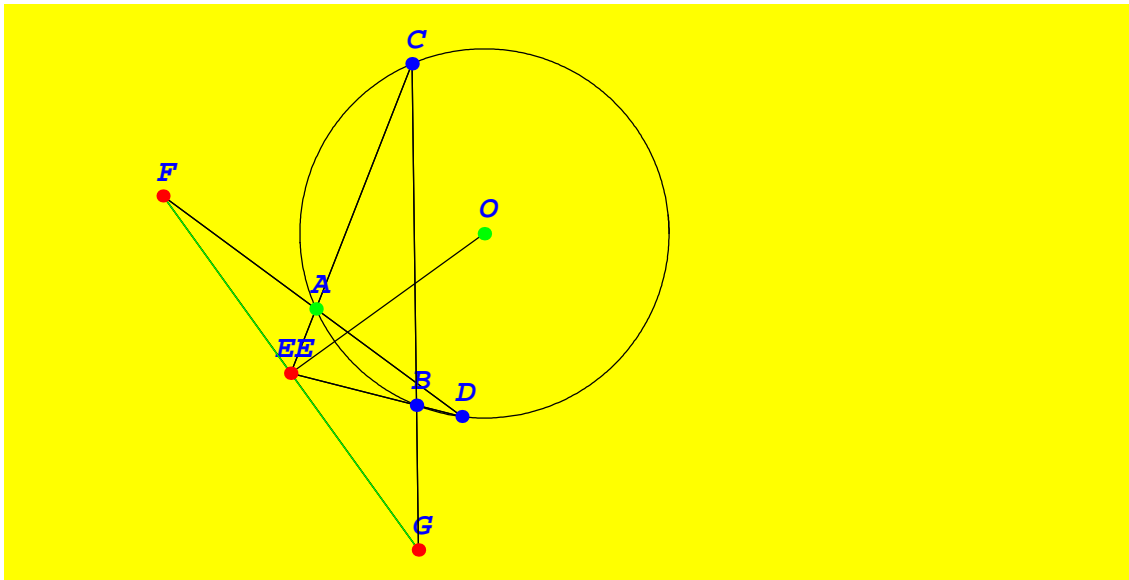
- Input to the system:

```
Proposition["Butterfly", any[O, A, B, C, D, EE, F, G], point[O, A] ^
  pon[C, circle[O, A]] ^
  pon[B, circle[O, A]] ^
  pon[D, circle[O, A]] ^
  inter[EE, line[A, C], line[B, D]] ^ inter[F, line[D, A],
  tline[EE, EE, O]] ^ inter[G, line[B, C], line[EE, F]]
  => distequal[EE, F, G, EE]
```

- Input to the system:

```
Simplify[Proposition["Butterfly"], by -> GraphicSimplifier]
```

- Output generated automatically by the system



$|EEF| \cong |GEE|$  for this configuration of the points

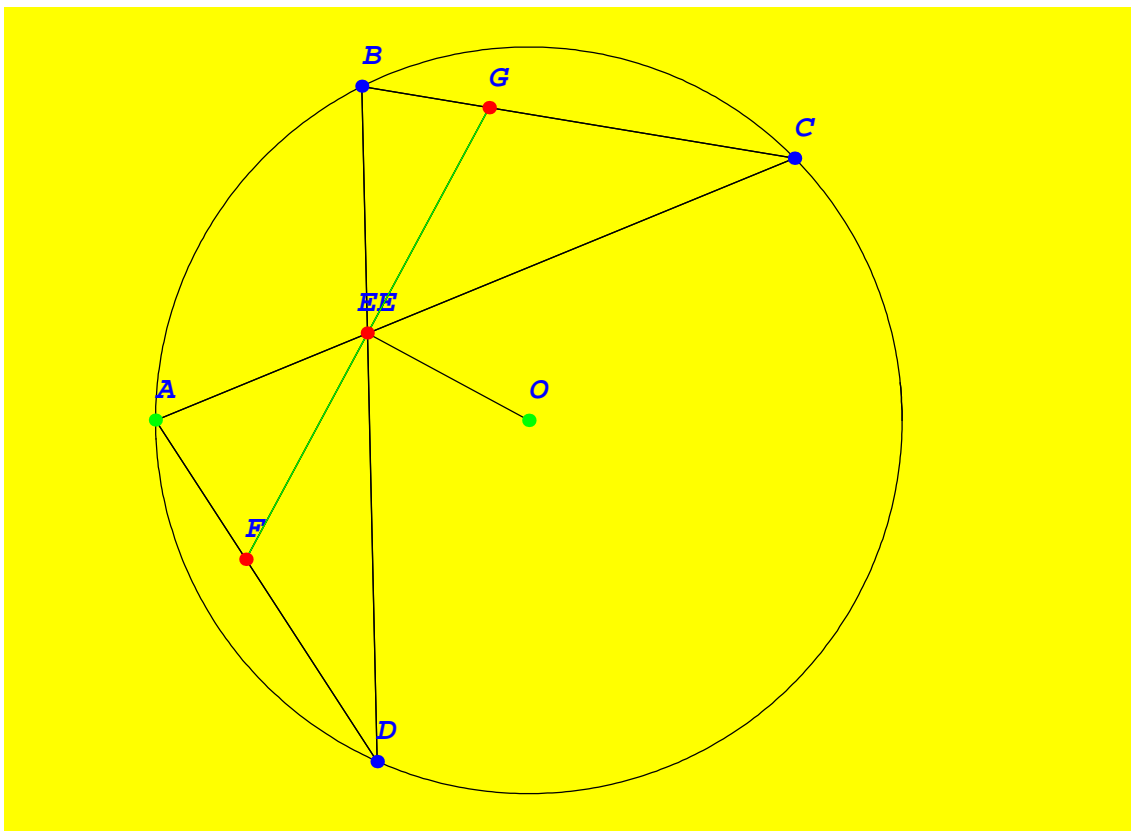
- End of output
- Input to the system:

```
KnowledgeBase["Butterfly_g", any[O, A, B, C, D], {{O, {100, 100}},
  {A, {0, 100}}, {B, {55, 190}}, {C, {171, 170}}, {D, {60, 10}}}]
```

- Input to the system:

```
Simplify[Proposition["Butterfly"],
  by -> GraphicSimplifier, using -> KnowledgeBase["Butterfly_g"]]
```

- Output generated automatically by the system



$|EEF| \cong |GEE|$  for this configuration of the points

- End of output
- Input to the system:

```
Prove[Proposition["Butterfly"], by → GeometryProver,
  ProverOptions → {Method → "GroebnerProver", Refutation → False}]
```

```
- ProofObject -
```

- Inserted notebook generated automatically by the system

Prove:

(Proposition (Butterfly))

$$\forall_{O,A,B,C,D,EE,F,G} (\text{point}[O, A] \wedge$$

$$\text{pon}[C, \text{circle}[O, A]] \wedge \text{pon}[B, \text{circle}[O, A]] \wedge$$

$$\text{pon}[D, \text{circle}[O, A]] \wedge \text{inter}[EE, \text{line}[A, C], \text{line}[B, D]] \wedge$$

$$\text{inter}[F, \text{line}[D, A], \text{tline}[EE, EE, O]] \wedge$$

$$\text{inter}[G, \text{line}[B, C], \text{line}[EE, F]] \Rightarrow \text{distequal}[EE, F, G, EE])$$

with no assumptions.

To prove the above statement we shall use the Gröbner bases 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  $EE$ , and points  $\{O\}$  and  $\{F, G\}$  on the two axes.

Using this coordinate system we have the following points:

$$\{\{EE, 0, 0\}, \{O, u_1, 0\}, \{A, u_2, u_3\}, \{B, u_4, x_1\},$$

$$\{F, 0, x_2\}, \{G, 0, x_3\}, \{C, x_4, x_5\}, \{D, x_6, x_7\}\}$$

The algebraic form of the given construction is:

(1)

$$\forall_{v_1, v_2, x_1, x_2, x_3, x_4, x_5, x_6, x_7} ((-1) + u_4 v_2 + -v_2 x_6 == 0 \wedge$$

$$(-1) + u_2 v_1 + -v_1 x_4 == 0 \wedge 2 u_1 u_2 + -u_2^2 + -u_3^2 + -2 u_1 x_4 + x_4^2 + x_5^2 == 0 \wedge$$

$$2 u_1 u_2 + -u_2^2 + -u_3^2 + -2 u_1 u_4 + u_4^2 + x_1^2 == 0 \wedge$$

$$2 u_1 u_2 + -u_2^2 + -u_3^2 + -2 u_1 x_6 + x_6^2 + x_7^2 == 0 \wedge u_3 x_4 + -u_2 x_5 == 0 \wedge$$

$$x_1 x_6 + -u_4 x_7 == 0 \wedge -u_2 x_2 + -u_3 x_6 + x_2 x_6 + u_2 x_7 == 0 \wedge$$

$$u_4 x_3 + x_1 x_4 + -x_3 x_4 + -u_4 x_5 == 0 \Rightarrow x_2^2 + -x_3^2 == 0)$$

We have to compute the Gröbner bases of the hypothesis polynomials (GB).

The polynomials of the Gröbner bases are:

$$\begin{aligned} & \{-u_2^2 + -u_3^2 + (-2u_1u_2^2 + 2u_2^3 + 2u_2u_3^2) v_1, \\ & -2u_1u_2 + u_2^2 + u_3^2 + 2u_1u_4 + (4u_1u_2u_4 - 2u_2^2u_4 - 2u_3^2u_4 - 2u_1u_4^2) v_2, \\ & -2u_1u_2u_3u_4 + u_2^2u_3u_4 + u_3^3u_4 + (2u_1u_2^2 + -u_2^3 + -u_2u_3^2) x_1 + \\ & (2u_1u_2^2 + -u_2^3 + -u_2u_3^2 + -u_2^2u_4 + -u_3^2u_4) x_2, \\ & 2u_1u_2u_3u_4 + -u_2^2u_3u_4 + -u_3^3u_4 + (-2u_1u_2^2 + u_2^3 + u_2u_3^2) x_1 + \\ & (2u_1u_2^2 + -u_2^3 + -u_2u_3^2 + -u_2^2u_4 + -u_3^2u_4) x_3, \\ & -2u_1u_2^2 + u_2^3 + u_2u_3^2 + (u_2^2 + u_3^2) x_4, \\ & 2u_1u_2u_3 + -u_2^2u_3 + -u_3^3 + (-u_2^2 + -u_3^2) x_5, \\ & 2u_1u_2u_4 + -u_2^2u_4 + -u_3^2u_4 + (2u_1u_2 + -u_2^2 + -u_3^2 + -2u_1u_4) x_6, \\ & (-2u_1u_2 + u_2^2 + u_3^2) x_1 + (-2u_1u_2 + u_2^2 + u_3^2 + 2u_1u_4) x_7, \\ & 2u_1u_2 + -u_2^2 + -u_3^2 + -2u_1u_4 + u_4^2 + x_1^2\} \end{aligned}$$

We compute the normal form of the conclusion polynomial modulo GB.

The normal form is: 0

As the obtained normal form equals to 0 the statement is generically true.

Statistics:

The length of the Gröbner bases is 9

Time needed to compute the Gröbner bases: 0.27 Seconds.

Time needed to compute the normal form of the conclusion polynomial modulo GB:  
0.32 Seconds.

□

- End of inserted notebook

## Übersicht:

Algorithmische Beweisverfahren

Einige Beispiele (in Theorema)

**Diskussion**

## Einige Fragen:

Gibt es einen "universellen" Beweis-Finder für die gesamte Mathematik?

Was ist die praktische Alternative?

Kann man auch das Erfinden von Sätzen automatisieren?

⏪

◀

▶

⏩

19 of 29

### ■ Gibt es einen "universellen" Beweis-Finder?

Ja, z.B. A. Robinson's "Resolutionsverfahren", 1965.

War / ist (Haupt-) Forschungsgegenstand im Gebiet des automatischen Beweisens seit 1965.

Etliche Gruppen, Hunderte von Papers, z.B. Otter (L.Wos)

Kann man sich davon den Durchbruch für die Praxis des mathematischen Beweisens erwarten?

Warum nicht?

⏪

◀

▶

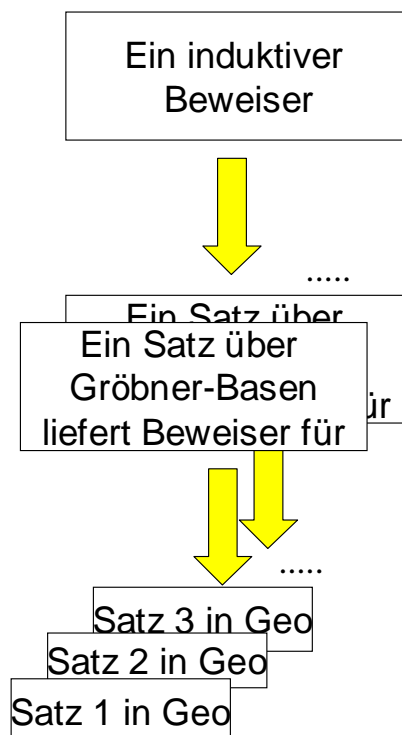
⏩

20 of 29

### ■ Alternative: Beweisen durch als korrekt bewiesene spezielle Beweis-Finder

Z.B.: Gröbner-Basen-Theorie (automatisch / computer-unterstützt) als korrekt bewiesen. (M. Bulo et al., in B.B., J. Campbell eds., Proc. of AISC 2004, RISC, Springer LNCS 3249, S. 171-184.)

Und dann Gröbner-Basen-Algorithmus als (wesentlichen Teil eines) speziellen Beweisers einsetzen.



## ■ Kann man auch das Erfinden von Sätzen automatisieren?

Ja, z.B. mit Methode des "Lazy Thinking" (Analyse von nicht erfolgreichen Beweisen), B.B. 1998.

Methode kann auch zum Erfinden von Algorithmen angewandt werden, B.B. 2003, A. Craciun 2003.

## Automatische Generierung von Lemmata in Induktionsbeweisen:

### ■ Induktive Definition der Addition

```
Definition["addition", any[m, n],
  m + 0 = m      " +0:"
  m + n+ = (m + n)+ " + .:" ]
```

### ■ Beweis, dass 0 auch Linksinverses ist

```
Proposition["left zero", any[m, n],
  0 + n = n  "0+"]
```

```
Prove[Proposition["left zero"],
  using → ⟨Definition["addition"]⟩,
  by → NNEqIndProver,

  ProverOptions → {TermOrder → LeftToRight},
  transformBy → ProofSimplifier, TransformerOptions → {branches → {Proved}}];
```

### ■ Beweisversuch für Kommutativität

```
Proposition["commutativity of addition", any[m, n],
  m + n = n + m  " + = " ]
```

```

Prove[Proposition["commutativity of addition"],
  using → ⟨Definition["addition"]⟩,
  by → NNEqIndProver,

  ProverOptions → {TermOrder → LeftToRight}, transformBy → ProofSimplifier,
  TransformerOptions → {branches → {Proved, Failed}}];

```

### ■ Automatische Generierung von Lemmata aus dem erfolglosen Beweis

*Theorema* Algorithmus "Cascade" ("Lazy Thinking"): Automatische Analyse von Beweisobjekten und Erzeugen von Vermutungen.

```

Prove[Proposition["commutativity of addition"],
  using → Definition["addition"],
  by → Cascade[NNEqIndProver, ConjectureGenerator],
  ProverOptions → {TermOrder → LeftToRight}];

```

## Automatische "Erfindung" des Gröbner-Basenalgorithmus

Ein endliches multivariates Polynomsystem  $G$  ist eine Gröbner-Basis, wenn jedes Polynom bei allen möglichen "Divisionen" bzgl.  $G$  immer denselben "Rest" hat. (Es gibt **unendlich** viele Polynome!)

**Der wesentliche Satz:** Um zu checken, dass ein Polynomsystem  $G$  die Gröbner-Basen-Eigenschaft hat, genügt es die Reste der **endlich** vielen "S-Polynome" aller Paare von  $G$  zu überprüfen.

S-Polynom[  $g_1, g_2$  ] := ein Polynom, das aus dem KGV der "führenden Terme" in  $g_1$  und  $g_2$  gebildet wird.

Mit der obigen Methode der (automatischen) Analyse von (automatischen) erfolglosen Beweisen kann man Folgendes machen:

□

Man setzt einen Algorithmus an, der auf jedes Paar  $(g_1, g_2)$  von Polynomen aus  $G$  einen **unbekannten Algorithmus**  $L$  anwendet und vom Ergebnis die Reste überprüft.

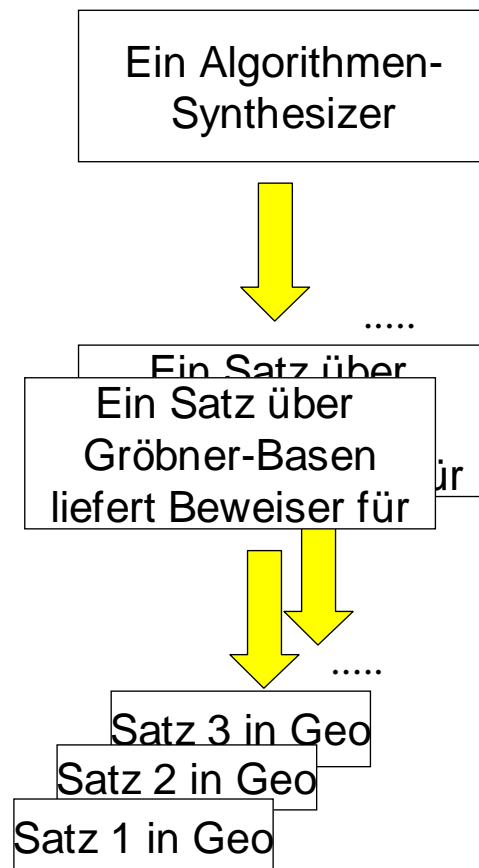
Der (automatische) Beweis der Korrektheit dieses "Algorithmus" schlägt natürlich fehl, weil  $L$  unbekannt ist.

Der automatische Beweisanalytiker generiert eine Eigenschaft (Spezifikation) von  $L$ , der Erfülltsein den Korrektheitssatz zum Abschluss zu bringen gestattet.

Aus der automatisch erzeugten Spezifikation für  $L$  erkennt man "sofort", dass  $L[g_1, g_2]$  im wesentlichen aus dem KGV der führenden Terme von  $g_1$  und  $g_2$  gebildet werden muss.

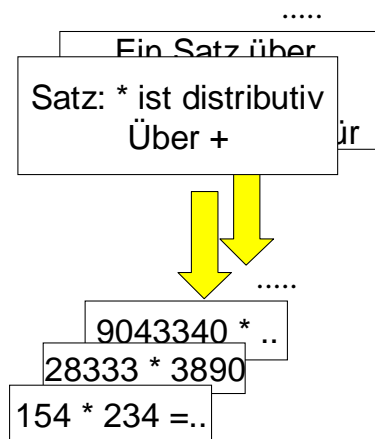
□

Damit ist der Gröbner-Basen-Algorithmus (plus Korrektheitsbeweis) im Wesentlichen automatisch "erfunden".

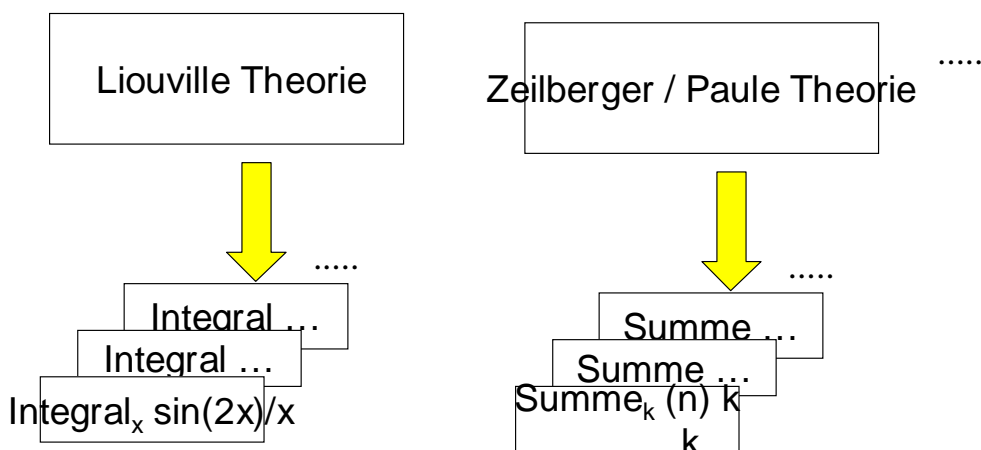


## Automatisches Beweisen ist die natürliche Fortsetzung des Grundgedankens der Mathematik

□



□



□

Einmal nachdenken über das Allgemeine, erspart unendlich oft mal Nachdenken im konkreten Fall.

□

Mathematik führt zur Trivialisierung von Mathematik.

□

Glücklicherweise: Je mehr Mathematik durch mehr Mathematik trivialisiert wird, um so mehr nicht trivialisierte Mathematik tut sich auf.

---

## Mögliche Konsequenzen für die Mathematik / Informatik Forschung, Ausbildung, Anwendung ?

□

Die Art, [Mathematik](#) zu erfinden, ihre Qualität zu kontrollieren, ihre Ergebnisse zu speichern und abzufragen, Mathematik zu unterrichten und anzuwenden [wird sich \(drastisch\) ändern](#).

□

Der Computer ist für die Mathematik Forschungs[gegenstand](#) und Forschungs[werkzeug](#) (die "Kreativitätsspirale").

□

Die Ausbildung im [formalem Denken](#) ist ein Schlüssel zur Innovation.

□

Je mehr Arbeit von anderen gemacht werden kann, umso mehr müssen wir uns [auf den jeweils höchsten Schichten der Arbeit](#) profilieren.

⏪ ⏩

26 of XXX

### ■ Mathematical Knowledge Management: A Recent International Endeavor

The *Theorema* group is (an initiator and member) of the international MKM ([Mathematical Knowledge Management](#)) Network (NuPri, Izabelle, MIZAR, ...)

Numerical Mathematics

Computer Algebra

Automated Theorem Proving

Mathematical Knowledge Management  
("Symbolic Computation" in its widest sense)

⏪ ⏩

27 of XXX

---

## ✓ The Theorema Project

---

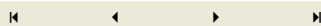
## An "Algorithm" for Algorithm Synthesis

---

## Algorithmic Synthesis of a Gröbner Bases Algorithm

---

## Conclusion



28 of XXX

### ■ The Algorithm Invention ("Synthesis") Problem

Given a problem specification  $P$  (in predicate logic), find an algorithm  $A$  such that

$$\forall_{\mathbf{x}} P[\mathbf{x}, A[\mathbf{x}]] .$$

Examples of specifications  $P$ :

```
P[x, y] ⇔ is-greater[x, y]
P[x, y] ⇔ is-sorted-version[x, y]
P[x, y] ⇔ has-derivative[x, y]
P[x, y] ⇔ are-factors-of[x, y]
P[x, y] ⇔ is-Gröbner-basis[x, y]
....
```

A general algorithm  $S$  for "all"  $P$  cannot exist (cf. B. Caviness 1970, ...) but ...

There is a rich literature on algorithm synthesis methods.



29 of 29

## ■ The "Lazy Thinking" Method (BB 2001): A Very Rough Sketch

**Given:** a problem specification P.

For **finding** an algorithm A that satisfies  $\forall_{\mathbf{x}} P[\mathbf{x}, A[\mathbf{x}]]$ ,

- we generate (automatically) a couple of (hopefully) simpler problems Q, R, ...,
- we synthesize algorithms B, C, ... for Q, R, ...
- from B, C, ... we compose (automatically) an algorithm A.

When can we **terminate**?

When we arrive at problems, for which algorithms are already known.

## ■ The "Lazy Thinking" Method: More Details

Given a problem specification P

- consider various "algorithm schemes" for A
- and try to **prove (automatically)**  $\forall_{\mathbf{x}} P[\mathbf{x}, A[\mathbf{x}]]$ .
- This proof will normally **fail** because nothing is known on the unspecified sub-algorithms in the algorithm scheme.
- From the temporary assumptions and goals in the failing proof situation **(automatically) generate such specifications for the unspecified sub-algorithms** that would make the proof possible.

Now, apply the method recursively to the auxiliary functions.

### □ This is "lazy"

- we use the condensed algorithmic **experience of others** (in the schemes)
- we start (routine) **proving before** we have an algorithm
- we just **wait until we fail** in order to get an idea from the failure.

## ■ Example: Synthesis of Merge-Sort [BB et al. 2003]

### □ Problem

Synthesize "sorted" such that

$$\forall_x \text{is-sorted-version}[x, \text{sorted}[x]].$$

("Correctness Theorem")

□ Knowledge on Problem

$$\forall_{x,y} \left( \text{is-sorted-version}[x, y] \Leftrightarrow \begin{array}{l} \text{is-sorted}[y] \\ \text{is-permuted-version}[x, y] \end{array} \right)$$

$$\forall_{x,y} \left( \text{is-sorted} \Leftrightarrow \begin{array}{l} \text{is-sorted}[y] \\ \text{is-permuted-version}[x, y] \end{array} \right)$$

$$\text{is-sorted}[\langle \rangle]$$

$$\forall_x \text{is-sorted}[\langle x \rangle]$$

$$\forall_{x,y,\bar{z}} \left( \text{is-sorted}[\langle x, y, \bar{z} \rangle] \Leftrightarrow \begin{array}{l} x \geq y \\ \text{is-sorted}[\langle y, \bar{z} \rangle] \end{array} \right)$$

etc.

□ An Algorithm Scheme

$$\forall_x \left( \text{sorted}[x] = \begin{array}{l} \text{special}[x] \qquad \qquad \qquad \leftarrow \text{is-trivial-tuple} \\ \text{merge}[\text{sorted}[\text{left-split}[x]], \text{sorted}[\text{right-split}[x]]] \leftarrow \text{otherwise} \end{array} \right)$$

("divide and conquer")

#### ▣ We Now Start Proving the Correctness Theorem and Analyze the Failing Proof

#### ▣ The Result of Applying Lazy Thinking

Lazy Thinking, [automatically](#) (in approx. 2 minutes on a laptop using the *Theorema* system), finds the following specifications for the sub-algorithms that provenly guarantee the correctness of the above algorithm (scheme):

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

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

$$\forall_{\mathbf{x}} (\text{left-split}[\mathbf{x}] \times \text{right-split}[\mathbf{x}] \approx \mathbf{x})$$

#### ▣ What Do We Have Now?

- **Case A:** We find algorithms [special](#), [merged](#), [left-split](#), [right-split](#) in our knowledge base for which the properties specified above are already contained in the knowledge base or can be derived from the knowledge base.

In this case, we are done, i.e. we have synthesized a sorting algorithm.

- **Case B:** We do not find algorithms [special](#), [merged](#), [left-split](#), [right-split](#) in our knowledge base for which the properties specified can be proved.

In this case, we apply Lazy Thinking again in order to synthesize appropriate [special](#), [merged](#), [left-split](#), [right-split](#)

until we arrive at sub-sub-...-algorithms in our knowledge base (e.g. the basic operations of tuple theory like append, prepend etc.)

Case B can be avoided, if we proceed systematically bottom-up ("complete theory exploration" in layers).

## ■ How Can we Teach (= Automate) the Method

- Compile a library of "algorithm design experience" (= [algorithm schemes](#)).
- Teach [\(automate\) proving](#).
- Teach [\(automate\) generation of useful specifications](#) of sub-algorithms from failing correctness proofs:

A simple (but amazingly powerful) **rule**:

Collect temporary assumptions  $T[x_0, \dots, A[ ], \dots ]$   
and temporary goals  $G[x_0, \dots, m [ A [ ] ] ]$

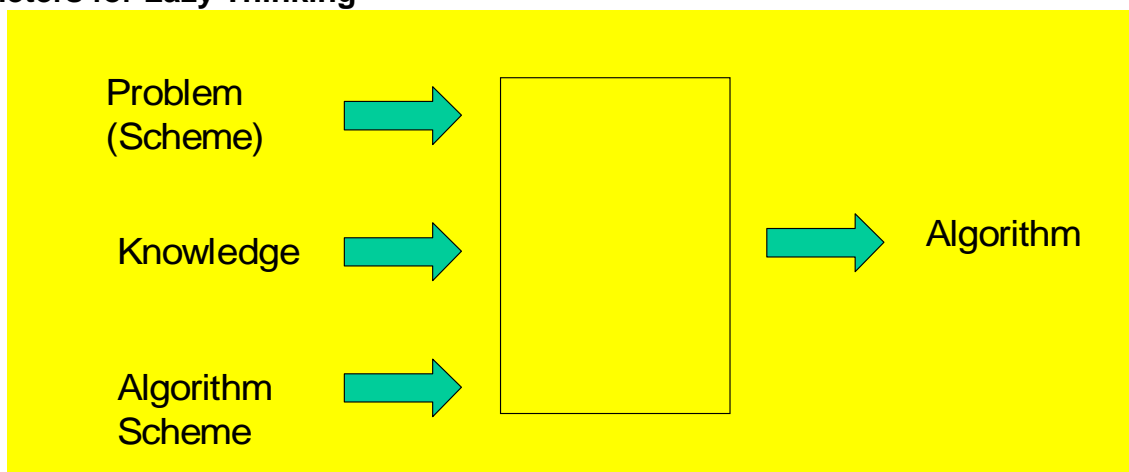
and produces specification

$$\forall x, \dots, y, \dots (T[x, \dots, Y, \dots] \Rightarrow G[y, \dots, m [ Y ] ]).$$

Details: see papers [BB 2003] and example.

Research topic: other rules.

## ■ Parameters for Lazy Thinking



Lazy Thinking

## ■ Example: Synthesis of Insertion-Sort

### □ Problem

Synthesize A such that

$$\forall_x \text{is-sorted-version}[x, A[x]].$$

#### ▫ Algorithm Scheme: "simple recursion"

$$\begin{aligned} A[\langle \rangle] &= c \\ \forall_x A[\langle x \rangle] &= s[\langle x \rangle] \\ \forall_{x, \bar{y}} (A[\langle x, \bar{y} \rangle] &= i[x, A[\langle \bar{y} \rangle]]) \end{aligned}$$

#### ▫ Resulting Specification for Subalgorithms

Lazy Thinking, [automatically](#) (in approx. 2 minutes on a laptop using the *Theorema* system), finds the following specifications for the auxiliary functions

$$\begin{aligned} c &= \langle \rangle \\ \forall_x (s[\langle x \rangle] &= \langle x \rangle) \\ \forall_{x, \bar{y}} \left( \text{is-sorted}[\langle \bar{y} \rangle] \Rightarrow \text{is-sorted}[i[x, \langle \bar{y} \rangle]] \right) \\ & \quad i[\langle x, \bar{y} \rangle] \approx (x - \langle \bar{y} \rangle) \end{aligned}$$

#### ▫ Details of an Automated Synthesis

See the notebooks automatically produced by *Theorema* for the insertion-sort example.

◀ ◁ ▷ ▶ 35 of XXX

### ■ How far can we go with this method?

Can we automatically synthesize algorithms for [non-trivial problems](#)?

Example of a non-trivial problem: construction of Gröbner bases.

What is "non-trivial"?

[Main algorithmic idea](#) of Gröbner bases theory: The "S-polynomials" ("critical pairs") together with the S-polynomial theorem.

Hence, question: Can Lazy Thinking [automatically invent the notion of S-polynomial](#) and automatically deliver the S-polynomial theorem.

✓ What we Have, What we Want

✓ An Algorithm for Algorithm Synthesis

## Algorithmic Synthesis of a Gröbner Bases Algorithm

### Conclusion

### ■ The Problem of Constructing Gröbner Bases

Find algorithm `Gb` such that

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

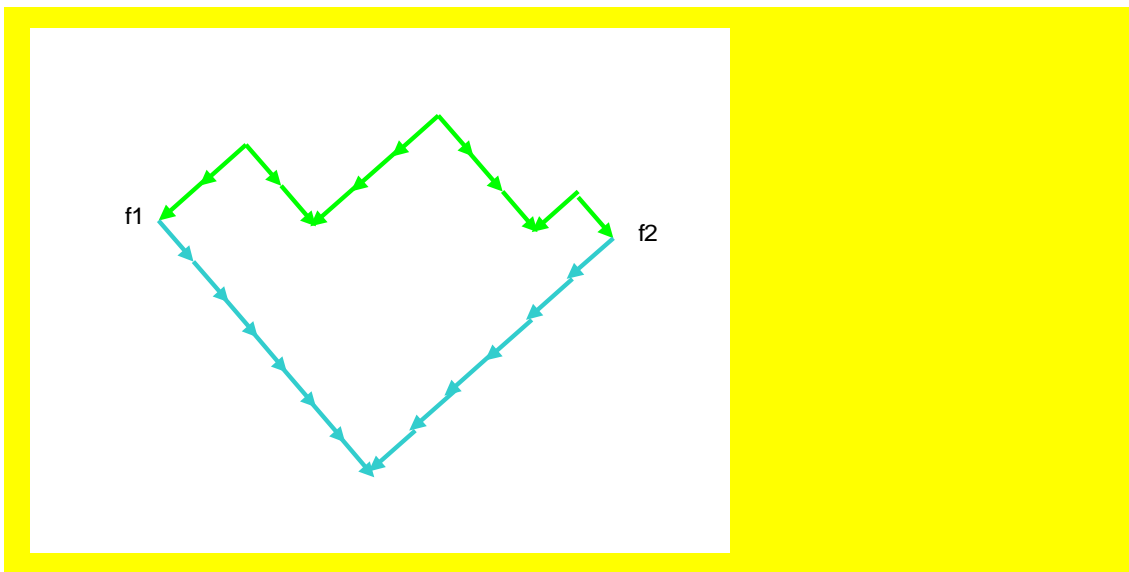
Definitions [BB 1965, 1970]:

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

## ■ Confluence of Division $\rightarrow_G$

$$(h1 \rightarrow_G h2) \Leftrightarrow \exists_{g \in G} \left( \begin{array}{l} lp[g] \mid lp[h1] \\ h2 = h1 - (lm[h1] / lm[g]) g \end{array} \right),$$

$$\text{is-confluent}[\rightarrow] : \Leftrightarrow \forall_{f1, f2} (f1 \leftrightarrow^* f2 \Rightarrow f1 \downarrow^* f2)$$



## ■ Knowledge on the Concepts Involved

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

etc.

## ■ Algorithm Scheme "Critical Pair / Completion"

$$\begin{aligned}
 \mathbf{A}[\mathbf{F}] &= \mathbf{A}[\mathbf{F}, \text{pairs}[\mathbf{F}]] \\
 \mathbf{A}[\mathbf{F}, \langle \rangle] &= \mathbf{F} \\
 \mathbf{A}[\mathbf{F}, \langle \langle g1, g2 \rangle, \bar{p} \rangle] &= \\
 &\text{where } [f = \text{lc}[g1, g2], h1 = \text{trd}[\text{rd}[f, g1], \mathbf{F}], h2 = \text{trd}[\text{rd}[f, g2], \mathbf{F}], \\
 &\left\{ \begin{array}{l} \mathbf{A}[\mathbf{F}, \langle \bar{p} \rangle] \qquad \qquad \qquad \leftarrow h1 = h2 \\ \mathbf{A}[\mathbf{F} - \mathbf{df}[h1, h2], \langle \bar{p} \rangle \times \left\langle \langle \mathbf{F}_k, \mathbf{df}[h1, h2] \rangle_{k=1, \dots, |\mathbf{F}|} \right\rangle \right] \leftarrow \text{otherwise} \end{array} \right. ]
 \end{aligned}$$

This scheme can be tried in any domain, in which we have a reduction operation  $\text{rd}$  that depends on sets  $\mathbf{F}$  of objects and a Noetherian relation  $>$  which interacts with  $\text{rd}$  in the following natural way:

$$\forall_{f, g} (f > \text{rd}[f, g]).$$

⏪

⏩

⏪

⏩

41 of XXX

## ■ The Essential Problem

The problem of synthesizing a Gröbner bases algorithm can now be also stated by asking whether starting with the proof of

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

we can *automatically produce the idea* that

$$\text{lc}[g1, g2] = \text{lcm}[\text{lp}[g1], \text{lp}[g2]]$$

and

$$\mathbf{df}[h1, h2] = h1 - h2$$

and prove that the idea is correct.

42 of XXX



## ■ Now Start the (Automated) Correctness Proof

With current theorem proving technology, in the *Theorema* system, the proof attempt could be done automatically. (Not yet fully implemented.)

43 of XXX

## ■ Details

### □ Upon Termination

It should be clear that, if the algorithm terminates, the final result is a finite set (of polynomials)  $G$  that has the property

$$\forall_{g_1, g_2 \in G} \left( \text{where } [f = \text{lc}[g_1, g_2], h_1 = \text{trd}[\text{rd}[f, g_1], F], \right. \\ \left. h_2 = \text{trd}[\text{rd}[f, g_2], F], \bigvee \left\{ \begin{array}{l} h_1 = h_2 \\ \text{df}[h_1, h_2] \in G \end{array} \right\} \right).$$

### □ We Have to Prove

We now try to prove that, if  $G$  has this property, then

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

Here, we only deal with the third, most important, property.

### □ Using Available Knowledge

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

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

□ **Assumption**

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

$$\begin{cases} p \rightarrow_G f1 \\ p \rightarrow_G f2. \end{cases}$$

We have to find a polynomial  $g$  such that

$$\begin{cases} f1 \rightarrow_G^* g, \\ f2 \rightarrow_G^* g. \end{cases}$$

□ **From the Assumption**

From the assumption we know that there exist polynomials  $g1$  and  $g2$  in  $G$  such that

$$\begin{aligned} &lp[g1] \mid p, \\ &f1 = rd[p, g1], \\ &lp[g2] \mid p, \\ &f2 = rd[p, g2]. \end{aligned}$$

From the final situation in the algorithm scheme we know that for these  $g1$  and  $g2$

$$\forall \begin{cases} h1 = h2 \\ df[h1, h2] \in G, \end{cases}$$

where

$$\begin{aligned} h1 &:= \text{trd}[f1', G], f1' := rd[lc[g1, g2], g1], \\ h2 &:= \text{trd}[f2', G], f2' := rd[lc[g1, g2], g2]. \end{aligned}$$

□ **Case h1=h2: In this case**

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

(Note that here we used the requirements  $rd[lc[g1, g2], g1] \prec lc[g1, g2]$  and  $rd[lc[g1, g2], g2] \prec lc[g1, g2]$ .)

Hence, by elementary properties of polynomial reduction,

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

Now we are stuck in the proof.

□ **Use Specification Generation Algorithm**

However, using the above specification generation rule, we see that we could proceed successfully with the proof if  $lc[g1, g2]$  satisfied the following requirement

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

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,

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

$$f2 \rightarrow_g^* a \ q \ \text{trd}[\text{rd}[\text{lc}[g1, g2], g1], G],$$

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

▫ **Summarizing the Specifications of the Unknown Subalgorithm  $lc$**

( $lc$  requirement), which also could be written in the form:

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

and the requirements:

$$\begin{array}{l} \text{rd}[\text{lc}[g1, g2], g1] < \text{lc}[g1, g2], \\ \text{rd}[\text{lc}[g1, g2], g2] < \text{lc}[g1, g2], \end{array}$$

which, in the case of the domain of polynomials, are equivalent to

$$\begin{array}{l} lp[g1] \mid lc[g1, g2], \\ lp[g2] \mid lc[g1, g2]. \end{array}$$

▫ **A Suitable  $lc$**

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

is a suitable function that satisfies the above requirements.

Heureka! The crucial function  $lc$  (the "critical pair" function) in the critical pair / completion algorithm scheme has been "automatically" synthesized!

▫ **Case  $h1 \neq h2$  and, hence,  $df[h1, h2] \in G$ :**

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 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,

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

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

Namely,



## ■ Summary of the Synthesizing Proof Attempt

### □ Failure Situation

The proof, of course, fails at the point where it would need knowledge about the unknown subalgorithms `lc` and `df`.

### □ Beginning of the proof:

Let  $G$  be  $A[F]$ . We have to prove that

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

### □ Assumption

We only deal with the third, most important, property. For this, we assume

$$\begin{cases} p \rightarrow_G f1 \\ p \rightarrow_G f2. \end{cases}$$

and have to find a polynomial  $g$  such that

$$\begin{aligned} f1 &\rightarrow_G^* g, \\ f2 &\rightarrow_G^* g. \end{aligned}$$

⏪

⏩

⏪

⏩

45 of XXX

### ■ Generation of the Specification of $lc$

In the failing proof situation, by the (automated) analysis algorithm sketched above, we detect that the proof could be completed if the unknown  $lc$  satisfied the following property:

$$\begin{aligned} lp[g1] \mid lc[g1, g2], \\ lp[g2] \mid lc[g1, g2], \end{aligned}$$

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

**Eureka!** It is clear that this specification is (only) met by

$$lc[g1, g2] = lcm[lp[g1], lp[g2]].$$

⏪

⏩

⏪

⏩

46 of XXX

### ■ Generation of the Specification of $df$

Similarly, it can be (automatically) detected that

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

⏪

⏩

⏪

⏩

47 of XXX



## ■ 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-NewYork, 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-NewYork, 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 2004]

B. Buchberger.

Towards the Automated Synthesis of a Gröbner Bases Algorithm.

RACSAM (Review of the Royal Spanish Academy of Science), Vol. 98/1, to appear, 10 pages.