Mathematical Theory Exploration: Case Study Groebner Bases

Bruno Buchberger
RISC
Johannes Kepler University, Linz, Austria

SYNASC 2006
Timisoara, September 26-29

Mathematical Theory Exploration

Groebner Bases Algorithm Synthesis

Conclusion

Mathematical Theory Exploration

Groebner Bases Algorithm Synthesis

Conclusion

Relation to Previous Talks of Mine at SYNASC

SYNASC 2000: Mathematical Theory Exploration: A Methodology

SYNASC 2001: Axiomatic Groebner Bases Theory

SYNASC 2003: Lazy Thinking Method for Automated Algorithm Synthesis

Today: Synthesis of a Groebner Bases Algorithm by Lazy Thinking.

BB 2005:

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.

(Semi-)Automated Mathematical Theory Exploration

(Semi-)automate the process of exploring mathematical theories:

- invent definitions and axioms for concepts
- invent and prove / disprove propositions about the concepts
- invent problems involving the concepts
- invent and verify algorithms for problems

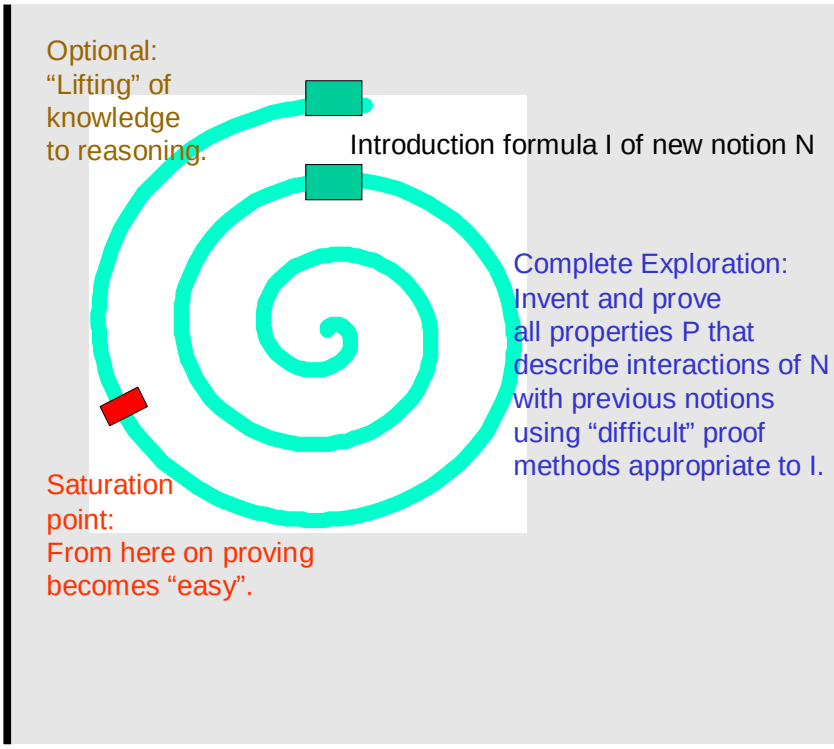
Store and retrieve mathematical knowledge.

BB 2001, BB 2002:

A bottom-up process ("systematic study of all interactions between concepts using schemes") and a top-down process ("generate conjectures from failing proof attempts") go hand in hand.

For an example of the bottom-up process: see talk by M. Hodorog and A. Craciun.

The Exploration Process: A Spiral



Today's Talk

Algorithm invention method: "Lazy Thinking" (BB 2002).

Case study for the invention of a "non-trivial" algorithm: algorithm for the construction of Groebner bases.

"Non-trivial"

Construction of Groebner bases:

- at the time of invention (1965, BB) was conjectured to be algorithmically unsolvable
- dozens of applications in algebraic geometry, invariant theory, optimization, coding theory, cryptography, symbolic summation, geo theorem proving, graph theory, ..., origami proving, sudoku solving, ...
- > 1000 papers, > 10 textbooks, > 3000 citations
- not yet synthesized by other synthesis methods.

The Algorithm Invention ("Synthesis") Problem

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

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

A general synthesis algorithm cannot exist but ...

Literature

There is a rich literature on algorithm synthesis methods, see survey

[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.

Our method is in the class of "scheme-based" methods. Closest (but essentially different):

> [Lau et al. 1999] K. K. Lau, M. Ornaghi, S. Tärnlund. Steadfast logic programs. Journal of Logic Programming, 38/3, 1999, pp. 259-294.
>
> And the work of A. Bundy and his group (U of Edinburgh) on the automated invention of induction schemes.

Algorithm Synthesis by "Lazy Thinking" (BB 2002, see also SYNASC 2003)

Given: A problem specification P. Find: An algorithm A for P.

- We assume we have "complete" knowledge on the auxiliary notion appearing in P.
- Consider known fundamental ideas ("algorithm schemes A") of how to structure algorithms A in terms of subalgorithms B, ...

  Try one scheme A after the other.
- For the chosen scheme A, try to prove $\forall_x$ P[x, A[x]]: From the failing proof construct specifications for the subalgorithms B, ... occurring in A.

Automated Invention of Sufficient Specifications for the Subalgorithms

A simple (but amazingly powerful) rule (B ... an unknown subalgorithm):

Collect temporary assumptions T[x0, ... A [ ], ...]

and temporary goals G[x0, ...B [... A [ ] ...]]

and produce specification

$$\forall_{x,\ldots,Y,\ldots} \left(T[x,\ldots Y,\ldots] \Rightarrow G[x,\ldots B[\ldots Y\ldots]]\right).$$

Details: see papers [BB 2003] and example.

Mathematical Theory Exploration

Groebner Bases Algorithm Synthesis

Conclusion

The Problem of Constructing Gröbner Bases

Find algorithm Gb such that

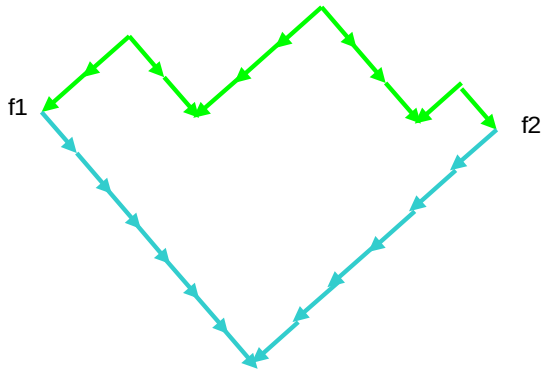
$$\forall_{\text{is-finite}[F]} \begin{pmatrix} \text{is-finite}[\,\text{Gb}[F]\,] \\ \text{is-Gröbner-basis}[\,\text{Gb}[F]] \\ \text{ideal}[F] = \text{ideal}[\,\text{Gb}[F]]. \end{pmatrix}$$

$$\text{is-Gröbner-basis}[G] \Leftrightarrow \text{is-Church-Rosser}[\,\rightarrow_G\,].$$

$\rightarrow_G$ ... a division step.

Confluence of Division $\rightarrow_G$

```
is-Church-Rosser[ → ] : ⇔ ∀_{f1,f2} (f1 ↔* f2 ⇒ f1↓* f2)
```



16 of 32

The Essential Algorithmic Idea in Groebner Bases Theory (BB 1965)

It suffices to consider the reduction of

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

for all polynomials g1 and g2 in the basis F.

17 of 32

Hence, the Essential Methodologic Question for the Power of Algorithm Synthesis

Can we *automatically* produce the idea (and can we automatically prove the idea correct) that

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

are the essential objects we have to consider.

So let's start with the synthesis using the "lazy thinking" method.

We Assume that we Have "Complete" Knowledge on the Auxiliary Concepts Involved

In fact, in a "natural" setting (where we build up mathematical knowledge in carefully designed layers), a few formulae are in the knowledge are sufficient for synthesizing a Groebner bases algorithm:

$$\forall_{F} \left(\text{is-Groebner-basis}[F] \Leftrightarrow \text{is-Church-Rosser}[\longrightarrow_F]\right),$$

$$\forall_{G} \left(\text{is-Church-Rosser}[\longrightarrow_G] \Leftrightarrow \forall_{p} \forall_{f1,f2} \left(\text{is-pp}[p] \wedge p \longrightarrow_G f1 \wedge p \longrightarrow_G f2 \Rightarrow f1 \downarrow^*_G f2\right)\right),$$

$$\forall_{\substack{\text{is-pp}[p] \\ G \\ f}} \left(p \longrightarrow_G f \Rightarrow \exists_{g} \left(g \in G \wedge (\text{lp}[g] \mid p) \wedge (f = \text{rd}[p, g])\right)\right),$$

$$\forall_{f1,f2,G} \left(\text{trd}[f1, G] \downarrow^*_G \text{trd}[f2, G] \Rightarrow f1 \downarrow^*_G f2\right),$$

$$\forall_{f1,f2,G} \left(f1 \downarrow^*_G f2 \Leftrightarrow \exists_{g} \left(f1 \rightarrow^*_G g \wedge f2 \rightarrow^*_G g\right)\right),$$

$$\forall_{a,q,p,g,G} \left(\text{trd}[\text{rd}[a * q * p, g], G] = a * q * \text{trd}[\text{rd}[p, g], G]\right),$$

$$\forall_{g1,g2,G} \left(g1 \in G \wedge g2 \in G \Rightarrow \forall_{p,a,q} \left((\text{trd}[\text{rd}[p, g1], G] = \text{trd}[\text{rd}[p, g2], G]) \Rightarrow (\text{trd}[\text{rd}[a * q * p, g1], G] = \text{trd}[\text{rd}[a * q * p, g2], G])\right)\right),$$

$$\forall_{f,g,G} \left((f \rightarrow_G g) \Leftarrow (g = \text{trd}[f, G])\right).$$

Use Algorithm Schemes

For example: a scheme for any domain, in which we have a reduction operation

rd[f, g] (result of "reducing f by g") satisfying rd[f,g] ≺ f

w.r.t. some Noetherian ordering ≺.

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

$$\forall_{F} \; A[F] = A[F, \text{pairs}[F]]$$

$$\forall_{F} \; A[F, \langle\rangle] = F$$

$$\forall_{F,g1,g2,\bar{p}} \; A[F, \langle\langle g1, g2\rangle, \bar{p}\rangle] =$$

$$\text{where}\left[f = lc[g1, g2],\; h1 = trd[rd[f, g1], F],\; h2 = trd[rd[f, g2], F],\; \begin{cases} A[F, \langle\bar{p}\rangle] & \Leftarrow\; h1 = h2 \\ A\left[F \frown df[h1, h2], \; \langle\bar{p}\rangle \asymp \left\langle \langle F_k, df[h1, h2]\rangle \;\middle|_{k=1,\ldots,|F|}\right\rangle\right] & \Leftarrow\; \text{otherwise} \end{cases}\right]$$

(What would happen, if we started with another algorithm scheme, e.g. divide-and-conquer?)

Now Start the (Automated) Correctness Proof

With current theorem proving technology, in the *Theorema* system (and other provers?), the proof attempt can be done automatically. (Ongoing PhD thesis by A. Craciun.)

Details

First, it can be proved (independently of what lc and df are), that if the algorithm terminates, the final result is a finite set (of polynomials) G that has the property

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

Using the available knowledge, we now try to prove that,if G has this property, then

$$\begin{aligned} &\text{is-finite}[G], \\ &\equiv_F \ = \ \equiv_G, \\ &\text{is-Gröbner-basis}[G], \\ &\quad \text{i.e. is-Church-Rosser}[\ \to_G\ ]. \end{aligned}$$

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

The (Automated) Proof Attempt

We use (a version of) Newman's Lemma (1942):

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

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

$$\left\{ \begin{matrix} p \to_G f1 \\ p \to_G f2. \end{matrix} \right.$$

We have to find a polyonomial g such that

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

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

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

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

$$\bigvee \left\{ \begin{matrix} h1 = h2 \\ \text{df}[h1, h2] \in G, \end{matrix} \right.$$

where

```
h1 := trd[f1', G], f1' := rd[lc[g1, g2], g1],
h2 := trd[f2', G], f2' := rd[lc[g1, g2], g2].
```

23 of 32

Case h1=h2

```
lc[g1, g2] →g1 rd[lc[g1, g2], g1] →G* trd[rd[lc[g1, g2], g1], G] =
 trd[rd[lc[g1, g2], g2], G] ←G* rd[lc[g1, g2], g2] ←g2 lc[g1, g2].
```

(Note that we assumed that lc[g1,g2] is reducible w.r.t. g1 and g2. The other case is easy.)

Hence, by elementary properties of polynomial reduction,

$$\forall_{a,q} \left(a\ q\ \mathrm{lc}[g1, g2] \to_{g1} a\ q\ \mathrm{rd}[\mathrm{lc}[g1, g2], g1] \to_{G}^{*} a\ q\ \mathrm{trd}[\mathrm{rd}[\mathrm{lc}[g1, g2], g1], G] = a\ q\ \mathrm{trd}[\mathrm{rd}[\mathrm{lc}[g1, g2], g2], G] \leftarrow_{G}^{*} a\ q\ \mathrm{rd}[\mathrm{lc}[g1, g2], g2] \leftarrow_{g2} a\ q\ \mathrm{lc}[g1, g2] \right).$$

Now we are stuck in the proof.

24 of 32

Now Use the Specification Generation Algorithm

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

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

With such an lc, we then would have

```
p →g1 rd[p, g1] = a q rd[lc[g1, g2], g1] →G* a q trd[rd[lc[g1, g2], g1], G] =
   a q trd[rd[lc[g1, g2], g2], G] ←G* a q rd[lc[g1, g2], g2] = rd[p, g2] ←g2 p
```

and, hence,

```
f1 →G* a q trd[rd[lc[g1, g2], g1], G],
```

```
f2 →G* a q trd[rd[lc[g1, g2], g1], G],
```

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

Summarize the (Automatically Generated) Specifications of the Subalgorithm lc

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

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

and the requirements:

```
lp[g1] | lc[g1, g2],
lp[g2] | lc[g1, g2].
```

Now this problem can be attacked independently of any Gröbner bases theory, ideal theory etc. In fact, it can be solved by high-school mathematics!

A Suitable lc

```
lcp[g1, g2] = lcm[lp[g1], lp[g2]]
```

is a suitable function that satisfies the above requirements.

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

Case h1≠h2

In this case, df[h1,h2]∈G:

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

If we apply the rule for the generation of subalgorithm specifications, we obtain immediately:

$$\forall_{\substack{G,h1,h2 \\ h1 \neq h2}} (df[h1, h2] \in G \Rightarrow h1 \downarrow_G{}^* h2) \quad (df\ \text{requirement}).$$

28 of 32

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 satifies (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!)

29 of 32

The Necessary Power of Automated Reasoning

The above proof is in the range of current automated theorem proving methodology. It is possible in Theorema, see the ongoing PhD thesis of Adrian Craciun.

Suitable automated provers for the lazy thinking synthesis method must have a couple of properties:

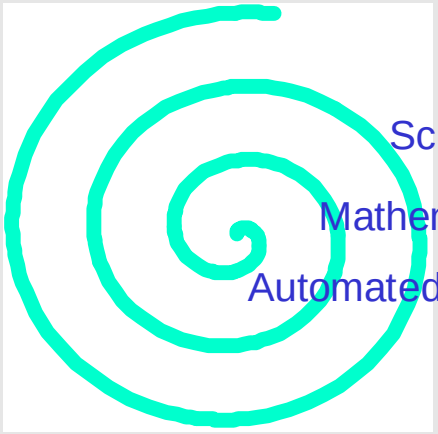
- must be proof generators (not only checkers)
- must generate a proof object also in the case of failure
- must adhere to the natural deduction paradigm
- must have a certain power.

30 of 32

Mathematical Theory Exploration

Groebner Bases Algorithm Synthesis

Conclusion



The automation of the mathematical theory exploration process is a fundamental goal for 21th century mathematics.

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 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.