Mathematical Theory Exploration

Bruno Buchberger
RISC (Research Institute for Symbolic Computation)
Johannes Kepler University, Linz, Austria

Invited Talk at IJCAR 2006
Seattle, USA, August 17 - 20, 2006

Goals, Obstacles, Chances

The Mathematical Theory Exploration Process

Example: Groebner Bases Algorithm Synthesis

Conclusion

Goals, Obstacles, Chances

The Mathematatcial Theory Exploration Cycle

Example: Groebner Bases Algorithm Synthesis

Conclusion

The Simple Message

- To the automated reasoning community:

  Mathematics is the main target! ...

- To my fellow mathematicians:

  Mathematics is (automated) reasoning. ...

Automated Reasoning: Little Impact on Mathematics.

■ "Mathematics": mathematical theory exploration

for example, writing lecture notes on analysis,

for example, writing a research monograph on Groebner bases theory,

for example, doing research on the Poincare conjecture,

for example, writing a paper for a journal,

for example, re-organizing the knowledge in a journal for checking the originality of results,

...

■ I do not mean: automation of mathematical "intuition"

Mathematical exploration is an alternation between "hot" (intuitive) and "cool" (formal) phase.

It is good enough if we can give significant algorithm-support to the cool phases.

However:

- formal theory exploration *is* one source of intuition,
- intuition often comes from consideration of examples; formal reasoning (computation) supports this.
- intuition often comes from failing proof (algorithm ...) attempts; formal reasoning can support this.

Automated Reasoning: Little Impact on Mathematics. Why?

- **no algorithm libraries in automated reasoning systems (exceptions: ...)**
- **no mathematical knowledge bases (exceptions: ...)**
- **syntax, proof presentation, ... not attractive**
- **little attention to special theorem proving (vs. general theorem proving)**
- **little attention to "mathematical theory exploration" (vs. isolated theorem proving)**
- **little attention to "proof generation" (vs. "proof checking")**
- **little attention to "migration through reasoning levels" (vs. one-level reasoning)**
- **a social reasons: "the mathematicians"**

Automated Mathematical Theory Exploration: Time is Ripe

- **software technology**

- **front-ends: ..., 2-dimensional syntax, "screen and keyboard" instead of "paper and pencil", ...**

- **hardware**

- **mathematical logic and automated reasoning**

- **structural build-up of "all of mathematics" (Bourbakism)**

- **tremendous advances in algorithmic mathematics (discrete, algebraic, symbolic, ...)**

- **an international community on "automated mathematical theory exploration": QED, Calculemus, MKM, ...**

The Way to Go: Systems for Mathematical Theory Exploration

■ predicate logic as the working language

- predicate logic is "practical"
- writing in logical syntax (as opposed to "pretty printing")
- algorithms and theorems expressed in one language
- reasoners: provers, solvers, simplifiers
- object level and meta-level expressed in one language: prove (special) reasoners correct

10 of 63

The Way to Go: Systems for Mathematical Theory Exploration

■ put effort also into special reasoners (as opposed to general reasoners)

"computer algebra and automated theorem proving":

- RISC PhD Curriculum 1982
- JSC editorial 1985
- Calculemus Group 1995
- forthcoming book by J. Harrison.

11 of 63

The Way to Go: Systems for Mathematical Theory Exploration

- **put effort also into automated proof generation (as opposed to automated proof checking)**

- **do not (always) look at "first principles": there are four possibilities:**

build-up - from "first principles" - by "first principles"
- from "intermediate principles" - by "intermediate principles"

- **create the logic frame for "proving reasoners correct":**

build-up - from "first principles" - by "first principles"
- from "intermediate principles" - by "intermediate principles"

- **focus on the 99 % of "easy" reasoning in theory exploration**

("easy" is a relative concept anway!)

The Way to Go: Systems for Mathematical Theory Exploration

■ **support also invention of mathematical knowledge: e.g. by schemes and learning form failure**

■ **support structured build-up of mathematical knowledge:**

- categories / functors
- "completion of knowledge"
- "mathematical personalities"
- algorithmic knowledge
- "(anti)-Bourbakism of the 21st century"

13 of 63

The Way to Go: Systems for Mathematical Theory Exploration

■ **the next generation of mathematicians: formally and algorithmically trained**

14 of 63

The Way to Go: Systems for Mathematical Theory Exploration

■ **the future of the organization of mathematics: formal-reasoning based**

- build-up of globally accessible mathematical
  (non-algorithmic and algorithmic) knowledge bases
- refereeing
- knowledge retrieval
- knowledge self-expansion
- re-structuring (expression of "mathematical personalities")

An indicator: the NIST project on "Handbook of Special Functions" (Abramovitz, Stegun; chapter on "computer algebra" by F. Chizak and P. Paule).

15 of 63

Goals, Obstacles, Chances

The Mathematcial Theory Exploration Process

Example: Groebner Bases Algorithm Synthesis

Conclusion

16 of 63

Examples are Taken from the Theorema Project

The Theorema project aims at prototyping features of a system for mathematical theory exploration.

The *Theorema* Group: B. B. (leader), T. Jebelean, W. Windsteiger, T. Kutsia, F. Piroi, M. Rosenkranz, M. Giese, and PhD students.

Some Automated Reasoners in Theorema:

- Predicate logic: natural deduction, S-decomposition
- Elementary analysis: PCS (alternating quantifiers)
- Set theory
- Induction on natural numbers, on tuples
- Equational logic with sequence variables
- Combinatorial identities
- Geometry (based on algebraic methods like Gröbner bases)
- Algorithms for symbolic functional analysis (boundary value problems)
- "Lazy Thinking" method for lemma and algorithm invention

17 of 63

Tools for structuring knowledge bases: functors, schemes, and others.

Computation within logic.

Two-dimensional user-definable syntax including "logicographic symbols".

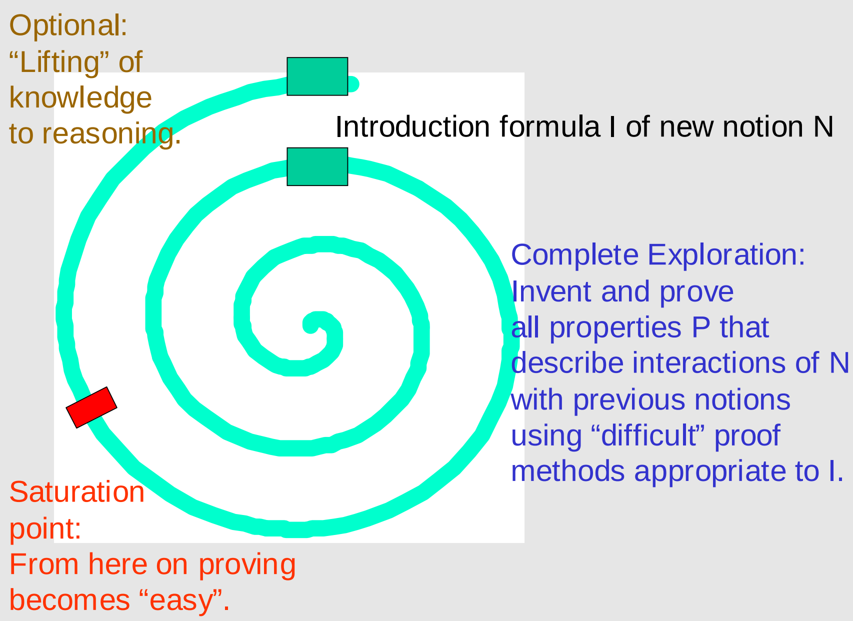
Readable Proofs.

Meta-language: *Mathematica*.

Object language: higher-order predicate logic with sequence variables.

18 of 63

The Exploration Process: A Spiral



19 of 63

Examples of Rounds through the Spiral

Difficult Proving	Introduction	Properties up to saturation	Easy Proving	Lifted Proving
induction	x+0=x x+y'=(x+y)'	arithm.laws	rewriting	count ' sort vars

20 of 63

Examples of Rounds through the Spiral

Difficult Proving	Introduction	Properties up to saturation	Easy Proving	Lifted Proving
Equational logic	group axioms	7 more equalities by mgu interaction	normal f. rewriting	simplification to group nf.

Examples of Rounds through the Spiral

Difficult Proving	Introduction	Properties up to saturation	Easy Proving	Lifted Proving
pred. log. set th.	def. of Groebner bases	S-poly theorem ...	rewriting w.r.t. ideal props.	ideal membersh. by Groebner bases

Examples of Rounds through the Spiral

Difficult Proving	Introduction	Properties up to saturation	Easy Proving	Lifted Proving
simple pred. log. plus Groebner b. algorithm	geo notions after coordinat.	propos. about configurations	???	???

Examples of Rounds through the Spiral

Difficult Proving	Introduction	Properties up to saturation	Easy Proving	Lifted Proving
pred. logic	definitions of $\bigcup$, $\bigcap$, ...	boolean algebra propert.	rewriting	boolean algebra simplifier

Examples of Rounds through the Spiral

Difficult Proving	Introduction	Properties up to saturation	Easy Proving	Lifted Proving
pred. logic	axioms of real closed fields	delineability theorem	reduce quants. to finite connectives	Collins' cylindrical alg. decomp.

25 of 63

Examples of Rounds through the Spiral

Difficult Proving	Introduction	Properties up to	Easy Proving	Lifted Proving

PCS (pred. log. plus Collins' algor.)	def. of limit	rewrite laws for limit	rewriting	inference rules for limit quantifier

26 of 63

More Details about Limit Exploration

Let's start from the situation:

- we already have explored the theory of the reals with +, < ... "completely"
- we already have explored the operations $\oplus$, ... on sequences of reals

27 of 63

Introduction of Limit

$$\textbf{Definition}\left[\text{"limit:"},\ \textbf{any}[f, a],\ \textbf{limit}[f, a] \Leftrightarrow \underset{\substack{\epsilon \\ \epsilon>0}}{\forall}\ \underset{N}{\exists}\ \underset{\substack{n \\ n\geq N}}{\forall}\ |f[n] - a| < \epsilon\right]$$

(Please, don't torture me with questions on types!)

28 of 63

A Typical Interaction of New Notion with Previous Notions

$$\textbf{Proposition}[\text{"limit of sum"},\ \textbf{any}[f, a, g, b],\ (\textbf{limit}[f, a] \wedge \textbf{limit}[g, b]) \Rightarrow \textbf{limit}[f \oplus g, a + b]]$$

29 of 63

Such Properties can be Invented Automatically by Schemes

A typical scheme:

$$\underset{P,F,G}{\forall} \left(\text{Monotony}[P, F, G] \Longleftrightarrow \underset{a,b,f,g}{\forall} \left(P[f, a] \wedge P[g, b] \right) \Rightarrow P[F[f, g], G[a, b]] \right)$$

Given a knowledge base in which 'Limit', '+', and '+' occurs, we can apply the above scheme for "inventing" (proposing, conjecturing) a proposition:

$$\text{Monotony}[\text{Limit}, \oplus, +]$$

i.e.

$$\underset{a,b,f,g}{\forall} \left(\text{Limit}[f, a] \wedge \text{Limit}[g, b] \right) \Rightarrow \text{Limit}[f \oplus g, a + b]$$

('Monotony' is a "relator" or (the description of) a "category".)

(I do not discuss the question here how the right substitutions for F and G can be automatically guessed. See, however, section 3 on algorithm synthesis: the creative power of analyzing failing proofs!)

The Role of Formula Schemes for Invention

By setting up a library of formula schemes, most formulae in the completion process of exploring a new notion can be generated automatically:

- propositions

- problems

- algorithms.

(Even, many interesting notions can be generated automatically by applying formulae schemes.)

The PCS Method for Analysis Proving (BB 2001)

This method reduces proving in elementary analysis (formulae with "alternating quantifiers" on functions) systematically to the solution of inequalities over the real numbers.

Produces "natural" proofs that also contain algorithmic information.

Instead of a detailed explanation of the proof method, let's look to the proof generated for the above example of a proposition:

■ Initialize Theorema

32 of 63

The Corresponding Part of the Theorema Session

```
Definition["limit:", any[f, a],
    limit[f, a] ⟺ ∀_{ϵ, ϵ>0} ∃_N ∀_{n, n≥N} |f[n] - a| < ϵ]
```

```
Proposition["limit of sum", any[f, a, g, b],
    (limit[f, a] ∧ limit[g, b])  ⇒  limit[f + g, a + b]]
```

Formulae from the knowledge base generated in the previous exploration rounds:

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

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

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

Pack everything into one "theory":

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

Now call the prover:

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

```
- ProofObject -
```

The following proof is generated fully automatically by the PCS prover:

Prove:

(Proposition (limit of sum)) $\underset{f,a,g,b}{\forall}(\text{limit}[f, a] \wedge \text{limit}[g, b] \Rightarrow \text{limit}[f+g, a+b])$,

under the assumptions:

(Definition (limit:)) $\underset{f,a}{\forall}\left(\text{limit}[f, a] \Leftrightarrow \underset{\substack{\epsilon \\ \epsilon>0}}{\forall}\ \underset{N}{\exists}\ \underset{\substack{n \\ n\geq N}}{\forall}(|f[n]-a|<\epsilon)\right)$,

(Definition (+:)) $\underset{f,g,x}{\forall}((f+g)[x] = f[x]+g[x])$,

(Lemma (|+|)) $\underset{x,y,a,b,\delta,\epsilon}{\forall}(|(x+y)-(a+b)| < \delta+\epsilon \Leftarrow (|x-a| < \delta \wedge |y-b| < \epsilon))$,

(Lemma (max)) $\underset{m,M1,M2}{\forall}(m \geq \max[M1, M2] \Rightarrow m \geq M1 \wedge m \geq M2)$.

We assume

(1) $\text{limit}[f_0, a_0] \wedge \text{limit}[g_0, b_0]$,

and show

(2) $\text{limit}[f_0+g_0, a_0+b_0]$.

Formula (1.1), by (Definition (limit:)), implies:

(3) $\underset{\substack{\epsilon \\ \epsilon>0}}{\forall}\ \underset{N}{\exists}\ \underset{\substack{n \\ n\geq N}}{\forall}(|f_0[n]-a_0|<\epsilon)$.

By (3), we can take an appropriate Skolem function such that

(4) $\underset{\substack{\epsilon \\ \epsilon>0}}{\forall}\ \underset{\substack{n \\ n\geq N_0[\epsilon]}}{\forall}(|f_0[n]-a_0|<\epsilon)$,

Formula (1.2), by (Definition (limit:)), implies:

(5) $\underset{\substack{\epsilon \\ \epsilon>0}}{\forall}\ \underset{N}{\exists}\ \underset{\substack{n \\ n\geq N}}{\forall}(|g_0[n]-b_0|<\epsilon)$.

By (5), we can take an appropriate Skolem function such that

(6) $\underset{\substack{\epsilon \\ \epsilon>0}}{\forall}\ \underset{\substack{n \\ n\geq N_1[\epsilon]}}{\forall}(|g_0[n]-b_0|<\epsilon)$,

Formula (2), using (Definition (limit:)), is implied by:

(7) $\underset{\substack{\epsilon \\ \epsilon>0}}{\forall}\ \underset{N}{\exists}\ \underset{\substack{n \\ n\geq N}}{\forall}\ (|(f_0 + g_0)[n] - (a_0 + b_0)| < \epsilon)$.

We assume

(8) $\epsilon_0 > 0$,

and show

(9) $\underset{N}{\exists}\ \underset{\substack{n \\ n\geq N}}{\forall}\ (|(f_0 + g_0)[n] - (a_0 + b_0)| < \epsilon_0)$.

We have to find N_2^* such that

(10) $\underset{n}{\forall}\ (n \geq N_2^* \Rightarrow |(f_0 + g_0)[n] - (a_0 + b_0)| < \epsilon_0)$.

Formula (10), using (Definition (+:)), is implied by:

(11) $\underset{n}{\forall}\ (n \geq N_2^* \Rightarrow |(f_0[n] + g_0[n]) - (a_0 + b_0)| < \epsilon_0)$.

Formula (11), using (Lemma (|+|)), is implied by:

(12) $\underset{\substack{\delta,\epsilon \\ \delta+\epsilon=\epsilon_0}}{\exists}\ \underset{n}{\forall}\ (n \geq N_2^* \Rightarrow |f_0[n] - a_0| < \delta \wedge |g_0[n] - b_0| < \epsilon)$.

We have to find δ_0^*, ϵ_1^*, and N_2^* such that

(13) $(\delta_0^* + \epsilon_1^* = \epsilon_0) \bigwedge \underset{n}{\forall}\ (n \geq N_2^* \Rightarrow |f_0[n] - a_0| < \delta_0^* \wedge |g_0[n] - b_0| < \epsilon_1^*)$.

Formula (13), using (6), is implied by:

$(\delta_0^* + \epsilon_1^* = \epsilon_0) \bigwedge \underset{n}{\forall}\ (n \geq N_2^* \Rightarrow \epsilon_1^* > 0 \wedge n \geq N_1[\epsilon_1^*] \wedge |f_0[n] - a_0| < \delta_0^*)$,

which, using (4), is implied by:

$(\delta_0^* + \epsilon_1^* = \epsilon_0) \bigwedge \underset{n}{\forall}\ (n \geq N_2^* \Rightarrow \delta_0^* > 0 \wedge \epsilon_1^* > 0 \wedge n \geq N_0[\delta_0^*] \wedge n \geq N_1[\epsilon_1^*])$,

which, using (Lemma (max)), is implied by:

(14) $(\delta_0^* + \epsilon_1^* = \epsilon_0) \bigwedge \underset{n}{\forall}\ (n \geq N_2^* \Rightarrow \delta_0^* > 0 \wedge \epsilon_1^* > 0 \wedge n \geq \max[N_0[\delta_0^*], N_1[\epsilon_1^*]])$.

Formula (14) is implied by

(15) $(\delta_0^* + \epsilon_1^* = \epsilon_0) \bigwedge \delta_0^* > 0 \bigwedge \epsilon_1^* > 0 \bigwedge \underset{n}{\forall}\ (n \geq N_2^* \Rightarrow n \geq \max[N_0[\delta_0^*], N_1[\epsilon_1^*]])$.

Partially solving it, formula (15) is implied by

(16) $(\delta_0^* + \epsilon_1^* = \epsilon_0) \wedge \delta_0^* > 0 \wedge \epsilon_1^* > 0 \wedge (N_2^* = \max[N_0[\delta_0^*], N_1[\epsilon_1^*]])$.

Now,

$(\delta_0^* + \epsilon_1^* = \epsilon_0) \wedge \delta_0^* > 0 \wedge \epsilon_1^* > 0$

can be solved for δ_0^* and ϵ_1^* by a call to Collins cad–method yielding a sample solution

$\delta_0^* \leftarrow \frac{\epsilon_0}{2}$,

$\epsilon_1^* \leftarrow \frac{\epsilon_0}{2}$.

Furthermore, we can immediately solve

$N_2^* = \max[N_0[\delta_0^*], N_1[\epsilon_1^*]]$

for N_2^* by taking

$N_2^* \leftarrow \max[N_0[\frac{\epsilon_0}{2}], N_1[\frac{\epsilon_0}{2}]]$.

Hence formula (16) is solved, and we are done.

□

33 of 63

At Saturation Point

We have a situation where inventing more propositions is uninteresting because more complicated propositions can be reduced to the already proved ones by "easy" proving, namely rewriting ("symbolic computation", "high school proving", "physicists proving", ...).

I.e. no point in proving:

```
limit[f + g + h] = limit[f] + limit[g] + limit[h]
```

34 of 63

Now, Lifting Knowledge to Inferencing

After implementation of the proved knowledge on the **predicate / function** Limit on the meta-level as rules for the **quantifier** *lim,* we can now compute (simplify) for example:

$$\text{Compute}\left[\lim_{n\to\infty}[(5+1/n)\,(n/(2n+5))]\right]$$

$$\frac{5}{2}$$

$$\text{Compute}\left[\lim_{a\to\infty}[(5+1/a)\,(a/(3a+5))]\right]$$

$$\frac{5}{3}$$

$$\text{Compute}\left[\lim_{n\to\infty}[(1+1/n)^n]\right]$$

E

(The implementation is, however, not yet proved formally! An important goal for all theory exploration systems designed along these lines!)

What do we have to prove? The application of the lim reasoning rules, without knowledge on Limit, have the same effect as using rewriting with the proved knowledge on Limit.

Goals, Obstacles, Chances

The Mathematcial Theory Exploration Cycle

Example: Groebner Bases Algorithm Synthesis

Conclusion

What I want to illustrate with this example:

Algorithm synthesis: one of the aspects of mathematical theory exploration.

A combination of

- (automated) theorem proving
- (automated) application of formula schemes for invention
- (automated) analysis of failing proofs

yields a powerful algorithm synthesis method ("lazy thinking method", BB 2002).

Powerful: able to synthesize algorithms for non-trivial problems.

In particular, powerful enough to replace myself.

37 of 63

"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.

38 of 63

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 ...

39 of 63

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.

40 of 63

Algorithm Synthesis by "Lazy Thinking" (BB 2002)

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 $\underset{x}{\forall}$ P[x, A[x]]: From the failing proof construct specifications for the subalgorithms B, ... occurring in A.

41 of 63

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

$$\underset{x,\ldots,Y,\ldots}{\forall} (T[x, \ldots Y, \ldots] \Rightarrow G[Y, \ldots m [Y]]).$$

Details: see papers [BB 2003] and example.

42 of 63

The method works well on simple problems like sorting

See [Buchberger 2003].

For example, using the divide-and-conquer scheme

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

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:

43 of 63

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

$$\underset{y,z}{\forall} \left(\begin{matrix} \text{is-sorted}[y] \\ \text{is-sorted}[z] \end{matrix} \Rightarrow \begin{matrix} \text{is-sorted}[M[y, z]] \\ M[y, z] \approx (y \asymp z) \end{matrix} \right)$$

$$\underset{x}{\forall} (L[x] \asymp R[x] \approx x)$$

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.

44 of 63

How Far Can We Go With Lazy Thinking

Successful synthesis of Groebner bases algorithm. See:

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.

45 of 63

The Problem of Constructing Gröbner Bases

Find algorithm Gb such that

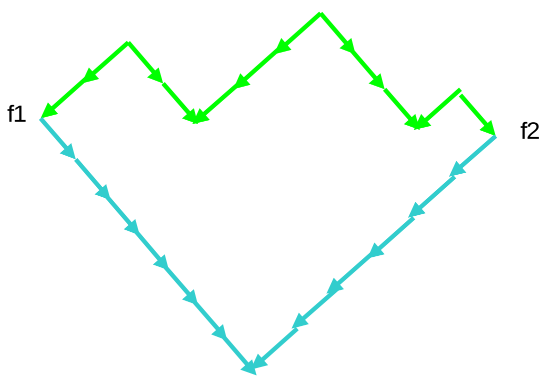
$$\underset{\text{is-finite}[F]}{\forall} \begin{pmatrix} \text{is-finite}[\,\text{Gb}[F]\,] \\ \text{is-Gröbner-basis}[\,\text{Gb}[F]\,] \\ \text{ideal}[F] = \text{ideal}[\,\text{Gb}[F]\,]. \end{pmatrix}$$

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

$\rightarrow_G$ ... a division step.

Confluence of Division $\rightarrow_G$

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



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.

48 of 63

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.

49 of 63

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

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

etc.

50 of 63

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] $\preceq$ f

w.r.t. some Noetherian ordering $\preceq$.

```
   ∀     pair-completion[A, lc, df] ⟺
A,lc,df

    ∀ A[F]  = A[F, pairs[F]]
    F
    ∀ A[F, 〈〉] = F
    F

        ∀        A[F, 〈〈g1, g2〉, p̄〉] =
    F,g1,g2,p̄
         where[f = lc[g1, g2],
               h1 = trd[rd[f, g1], F], h2 = trd[rd[f, g2], F],
           ⎧A[F, 〈p̄〉]                              ⇐  h1 = h2
           ⎨A[F ⌢ df[h1, h2],                      ⇐  otherwise   ]
           ⎩  〈p̄〉 ≍ 〈〈F_k, df[h1, h2]〉      |      〉]
                                     k=1,…,|F|
```

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

51 of 63

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.)

52 of 63

Details

First, it can be proved (independent 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

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

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.

53 of 63

Using Available Knowledge

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

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

Newman's lemma (1942):

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

54 of 63

The (Automated) Proof Attempt

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

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

We have to find a polyonomial g such that

$$f1 \to_G^* g,$$
$$f2 \to_G^* g.$$

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

```
lp[g1] | p,
f1 = rd[p, g1],
lp[g2] | p,
f2 = rd[p, g2].
```

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

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

where

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

55 of 63

Case h1=h2

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

(Note that here we used the requirements that lc[g1,g2] is reducible w.r.t. g1 and g2. The other cases are easy.)

Hence, by elementary properties of polynomial reduction,

$$\underset{a,q}{\forall} (a\, q\, lc[g1, g2] \to_{g1} a\, q\, rd[lc[g1, g2], g1] \to_G^* a\, q\, trd[rd[lc[g1, g2], g1], G] = a\, q\, trd[rd[lc[g1, g2], g2], G] \leftarrow_G^* a\, q\, rd[lc[g1, g2], g2] \leftarrow_{g2} a\, q\, lc[g1, g2]).$$

Now we are stuck in the proof.

56 of 63

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

$$\underset{p,g1,g2}{\forall} \left(\left(\begin{cases} lp[g1] \mid p \\ lp[g2] \mid p \end{cases} \right) \Rightarrow \left(\underset{a,q}{\exists} (p = a\, q\, lc[g1, g2]) \right) \right), \quad (\text{lc requirement})$$

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.

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

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

and the requirements:

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

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!

59 of 63

Case h1≠h2

In this case, df[h1,h2]∈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

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

60 of 63

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,

$$\underset{f,g}{\forall} (f \downarrow_{\{f-g\}}{}^{*} g).$$

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

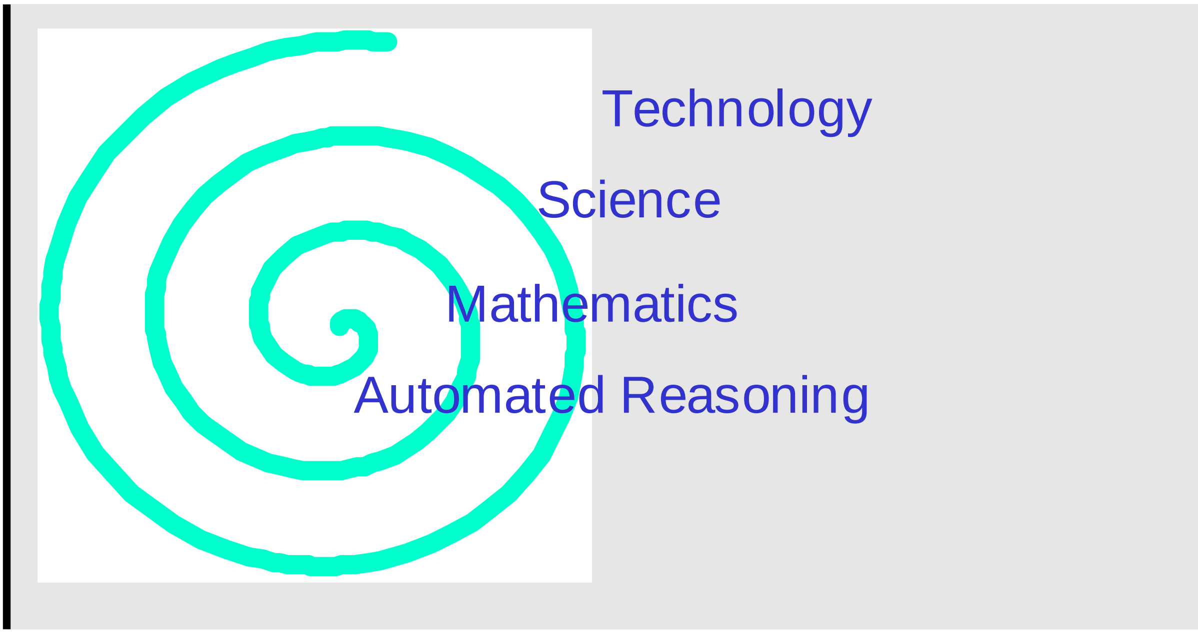
61 of 63

Goals, Obstacles, Chances

The Mathematcial Theory Exploration Cycle

Example: Groebner Bases Algorithm Synthesis

Conclusion



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.