Proving
by First and Intermediate Principles

Bruno Buchberger
RISC and RICAM, Austria

Invited talk at TYPES Workshop, Nov 1-2, 2004
University of Nijmegen, The Netherlands

⏮ ◀ ▶ ⏭ 2 of XXX

First and Intermediate Principles: A Clarification

Examples of a Provers Based on Intermediate Principles: A Geo Prover and an Analysis Prover

Some Details about Formal Groebner Bases Theory

⏮ ◀ ▶ ⏭ 3 of XXX

First and Intermediate Principles: A Clarification

A Fact

The current reasoning systems are hardly used by the current "working mathematicians".

□

Remedies:

improve the mathematicians

improve the systems.

⏮ ◀ ▶ ⏭ 4 of XXX

The Distinction Between Logicians, Mathematicians, and Computer Scientists is Not Wise

Mathematicians: apply reasoning

Computer scientists: apply reasoning (for algorithms)

Logicians: apply reasoning to reasoning.

Self-application is the nature of intelligence.

Reasoning about reasoning is a natural part of mathematics.

Every reasonable reasoner ("mathematician" in the broad sense) should be a logician, a "mathematician" (in the narrow sense), and a computer scientist.

⏮ ◂ ▸ ⏭ 5 of XXX

My Own Motivation

I am a "working mathematician":
Proof: AMS Subject Classification 13P10 "Gröbner Bases".

I want to get support from formal reasoning system when I am "working".

"Working" in mathematics is not "isolated theorem proving" but "theory exploration".

Thus, I started the Theorema Project in 1996.

⏮ ◂ ▸ ⏭ 6 of XXX

The Goal of Formal Mathematics

Support, by algorithms, the entire process of mathematical theory exploration:

One exploration phase:

Starting from a body of knowledge on some concepts, (e.g. +, * on numbers and functions)

invent (axioms, definitions for) new concepts (operations: predicates, functions), (e.g. limit)

invent and prove properties of notions,

invent problems about notions,

invent methods (algorithms) for problems and prove their correctness,

compute (apply algorithms to data),

organize, store, and retrieve knowledge.

One outcome of formal mathematics: provenly correct mathematical knowledge libraries.

7 of XXX

Current Situation

Current computer algebra systems like *Mathematica*, etc. support some of these aspects.

Current reasoning systems like Coq etc. support some other of these aspects.

Goal: Support all these aspects in one coherent logical and software frame.

Calculemus, MKM, TYPES are research networks that aim at creating integrated systems.

8 of XXX

Some Distinctive Features of Theorema

- prove and compute in one logic ("normal" predicate logic) including functors
- formal text (including the proofs generated) should be "readable"
- should support "easy" proving and allow interaction for "difficult" proving
- emphasizes "special" provers (reasoners): use nontrivial mathematics as "black box" for proving in special theories
- uses Mathematica (as meta-programming language, as front-end, for accessing mathematical algorithms)

Acknowledgement to *Theorema* Group members:

T. Jebelean, T. Kutsia, (K. Nakagawa), F. Piroi, M. Rosenkranz, W. Windsteiger.

PhD students: A. Craciun, L. Kovacs, N. Popov, C. Rosenkranz.

9 of XXX

The Emphasis of This Talk: Special Provers

Proving by "intermediate" principles.

I distinguish:

- proving from first and intermediate principles (knowledge)

- proving by first and intermediate principles (provers)

10 of XXX

First and Intermediate Principles in (Formal) Proving: A First Distinction

Let's fix some logic, e.g. some inference rule system for (first order) predicate logic.

Example of a proposition (to be proved):

$$\underset{f,g:\mathbb{N}\to\mathbb{R}}{\forall}\ \underset{a,b\in\mathbb{R}}{\forall}\left(\begin{array}{l}\mathtt{limit}[f,a]\\ \mathtt{limit}[g,b]\end{array}\Rightarrow \mathtt{limit}[f+g,\ a+b]\right) \qquad (\mathrm{prop})$$

"Proving **from** first priciples": Prove (prop) by the inference rules of predicate logic

- from the axioms of set theory
- and the definitions of the ingredient notions (quite many!)

"Proving **from** intermediate priciples": Prove (prop) by the inference rules of predicate logic

- (from the axioms of set theory)
- (the definitions of the ingredient notions)
- and intermediate knowledge on intermediate notions (already proved or assumed), e.g. knowledge on the operations on real numbers and the operations on real sequences.

11 of XXX

Proving from first principles is, in principle, possible.

However, proving from first principles is unpractical.

The necessity of building up mathematical knowledge in "layers" seems to be clear:

- Every time a new notion is introduced, a new "layer" of knowledge starts.
- One layer of knowledge should be "completed" before a new layer is started.
- This strategy, typically, generates "short and easy" proofs.

One could call this strategy "proving **from** intermediate principles".

However, in this talk, the emphasis is on another, additional, type of proving **by** intermediate principles.

12 of XXX

Proving "by" First and Intermediate Principles:

This distinction refers to the inference techniques which we allow in a proof step.

This distinction has also to do with the distinction between the "proof checking" and the "proof generating" philosophies.

There are basically two extremes in the approach to future formal mathematics:

- (formal) proving by generated proof checkers (by "first principles"),
- (formal) proving by proved proof generators (by "intermediate principles").

Please note the parsing!

There are all possible variants conceivable in between the two extremes.

In my personal view, the proved proof generators approach is more promising for making our systems attractive for the "working mathematician".

13 of XXX

The (Generated Proof) Checkers Approach

A. For the fixed inference rule system, write a proof checker that checks whether each step of a given proof is correct.

B. Individual proofs for (interesting) propositions (higher up in the mathematical knowledge hierarchy) can now be proposed by the user and checked.

C. Since this is practically unattractive (unfeasible), various proof generators (of low and high sophistication) can be written that produce "proofs". However, these proofs are only considered as "proof proposals". They are only accepted if checked by the proof checker.

Ad A: The correctness of the checker can be seen "by inspection" because it is "small" (de Brujn's principle). (Alternatively, we could just say: The implementation *is* the inference system.)

14 of XXX

The Proved (Proof Generators) Approach

A. The basis is also a proof checker (which is nothing else than saying that the basis is an - implemented - inference system P).

B. Of course, one could propose individual proofs to the checker.

C. However, typically, for certain fixed "theories" (knowledge bases), special proof generators are written that use "black box inference rules" that are only valid in the specific theories.

D. The correctness of a proof generator must be proved.

E. The proofs produced by proved proof generators need not be checked any more by the initial proof checker!

15 of XXX

Remarks about the Proved Proof Generators Approach

Ad D: Correctness of special prover Q for theory T: One has to prove that, for all knowledge bases K and formulae F, if $K \vdash_Q F$ then $K \cup T \vdash_P F$.

The (formal) proof of the correctness of Q must be done by using a prover whose correctness was already shown (i.e. at the beginning by using P alone).

Note that, typically, special provers are not just "black box versions of repeated applications of the inference rules in P". (Example: AC simplification by variable count.)

Proving the correctness of provers in a given logic and than applying them needs "reflection".

16 of XXX

Arguments in Favor of the Proved (Proof Generators) Approach

- The generated proofs checker approach does not seem to be practically feasible and attractive for formally developing interesting parts of mathematics. (Compare the reports of people who, for example, had the Groebner bases main theorem checked by a proof checker.)
- The proved proof generator approach seems to be what mathematicians have been doing all the time (although sometimes with low formal quality and, mostly, hiddenly).
- In fact, as soon as one's attention is open for the proved proof generator approach, one detects its application in nearly every section of mathematical papers and textbooks.
- It seems that the past success of building up mathematical theories without algorithm support mainly relies on the power of extending the proving systems while proving.

In order to make the approach practically attractive:

- the systems must be "opened" in the sense that
- the users may not only use the provers (proof generators) provided by the system
- but also must get the possibility to program their own proof generators and prove their correctness.

17 of XXX

Example of a Provers Based on Intermediate Principles: A Geo Prover

Reduction of geo proving to Groebner bases computation.

The main intermediate principle (black-box inference rule) used:

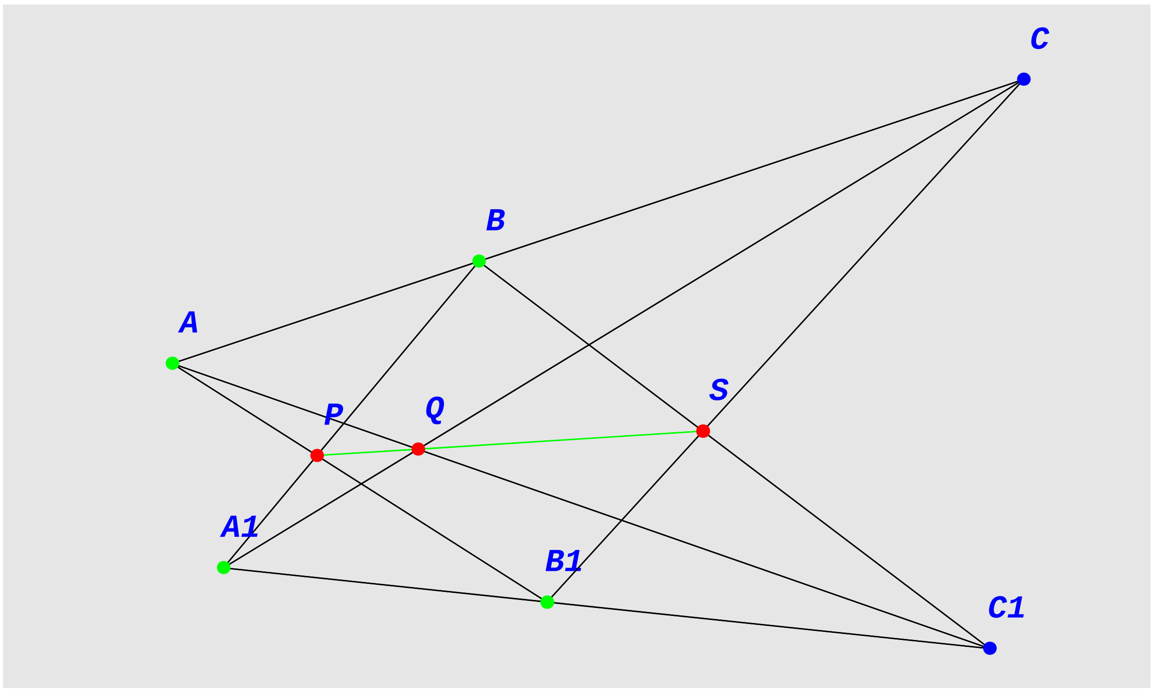
- Groebner bases computation.

Other intermediate principles used:

- simplification of arithmetical expressions to polynomial form,
- elimination of negation of equalities (Rabinovic trick).

18 of XXX

Example of a Geometrical Theorem: Pappus' Theorem



- Textbook formulation:

Let A,B, C and A1,B1, C1 be on two lines and P = AB1 ⋂ A1B, Q = AC1 ⋂ A1C, S = BC1 ⋂ B1C. Then P, Q, and S are collinear.

⏮ ◀ ▶ ⏭ 19 of XXX

Transformation into a Theorem on (Complex) Numbers by Coordinatization:

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

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

Using this coordinate system we have the following points:

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

The algebraic form of the above proposition:

(1)

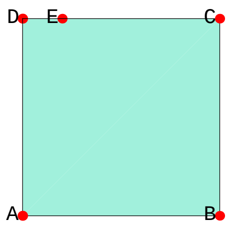
$$\forall_{u_1,u_2,u_3,u_4,u_5,u_6,u_7,x_1,x_2,x_3,x_4,x_5,x_6,x_7} (u_3 u_4 + -u_2 u_5 + -u_3 u_7 + u_5 u_7 + u_2 x_1 + -u_4 x_1 == 0 \wedge u_5 x_2 + -u_4 x_3 == 0 \wedge -u_1 u_2 + u_1 x_2 + -u_3 x_2 + u_2 x_3 == 0 \wedge x_1 x_4 + -u_7 x_5 == 0 \wedge -u_2 u_6 + -u_3 x_4 + u_6 x_4 + u_2 x_5 == 0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

Hence, essentially, we have to prove a universally quantified Boolean combination of polynomial equalities over the (complex) numbers.

20 of XXX

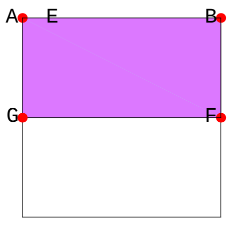
Example: Trisection an Angle by Origami (jointly with T. Ida)

By origami, an angle can be trisected. The method of construction is due to H. Abe. We want to prove that the construction yields the desired trisection:

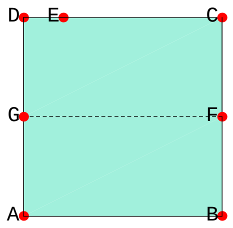


Step 2

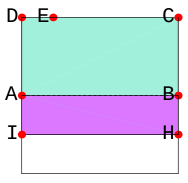
The problem is to trisect the angle ∠EAB.



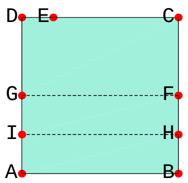
Step 3



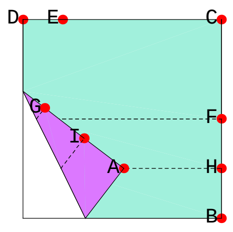
Step 4



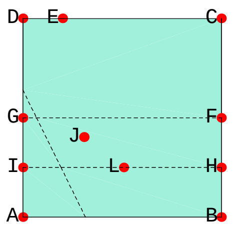
Step 5



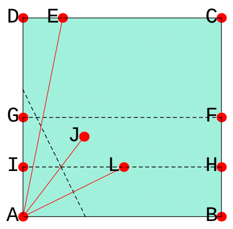
Step 6



Step 8



Step 9



Step 10

We now have to prove:

∡EAB /3= ∡JAB/2 = ∡LAB.

The Theorem After Coordinatization (Done Automatically in Theorema):

(Formula (TrisectingAngle): (1))

$$
\begin{aligned}
&\underset{a5,a6,a7,a8,b5,b6,b7,b8,c5,c6,c7,c8,r,x1,x10,x2,x3,x4,x5,x6,x7,x8,x9,y1,y10,y2,y3,y4,y5,y6,y7,y8,y9,\eta1,\eta2,\eta3,\eta4,\eta5,}{\forall}\\
&((c8 = 0) \wedge (a5 * r = 0) \wedge (c5 + \tfrac{1}{2} * b5 * r = 0) \wedge (-1 * r + x4 = 0) \wedge\\
&\quad (x5 = 0) \wedge (-1 * r + x7 = 0) \wedge (x8 = 0) \wedge (b5 * (r + (-1) * x1) + a5 * y1 = 0) \wedge\\
&\quad (c5 + \tfrac{1}{2} * a5 * (r + x1) + \tfrac{1}{2} * b5 * y1 = 0) \wedge (-1 * b7 * x10 + a7 * y10 = 0) \wedge\\
&\quad (c7 + \tfrac{1}{2} * a7 * x10 + \tfrac{1}{2} * b7 * y10 = 0) \wedge (b6 * (r + (-1) * x2) + a6 * y2 = 0) \wedge\\
&\quad (c6 + \tfrac{1}{2} * a6 * (r + x2) + \tfrac{1}{2} * b6 * y2 = 0) \wedge (c8 + a8 * x3 + b8 * y3 = 0) \wedge\\
&\quad (c5 + a5 * x4 + b5 * y4 = 0) \wedge (-1 * b6 * x5 + a6 * y5 = 0) \wedge (c5 + a5 * x5 + b5 * y5 = 0) \wedge\\
&\quad (c6 + \tfrac{1}{2} * a6 * x5 + \tfrac{1}{2} * b6 * y5 = 0) \wedge (b7 * (x5 + (-1) * x6) + a7 * (-1 * y5 + y6) = 0) \wedge\\
&\quad (c7 + \tfrac{1}{2} * a7 * (x5 + x6) + \tfrac{1}{2} * b7 * (y5 + y6) = 0) \wedge (c6 + a6 * x7 + b6 * y7 = 0) \wedge\\
&\quad (c6 + a6 * x8 + b6 * y8 = 0) \wedge (b7 * (x8 + (-1) * x9) + a7 * (-1 * y8 + y9) = 0) \wedge\\
&\quad (c7 + \tfrac{1}{2} * a7 * (x8 + x9) + \tfrac{1}{2} * b7 * (y8 + y9) = 0) \wedge (-1 + r * \eta1 = 0) \wedge\\
&\quad (-1 + (a5^2 + b5^2) * \eta2 = 0) \wedge (-1 + (a6^2 + b6^2) * \eta3 = 0) \wedge\\
&\quad (-1 + (a7^2 + b7^2) * \eta4 = 0) \wedge (-1 + (a8^2 + b8^2) * \eta5 = 0) \wedge\\
&\quad (c8 + a8 * \mu1 + b8 * \mu2 = 0) \wedge (b7 * (x5 + (-1) * \mu1) + a7 * (-1 * y5 + \mu2) = 0) \wedge\\
&\quad (c7 + \tfrac{1}{2} * a7 * (x5 + \mu1) + \tfrac{1}{2} * b7 * (y5 + \mu2) = 0) \wedge (-1 * b7 * \mu3 + a7 * \mu4 = 0) \wedge\\
&\quad (c6 + a6 * \mu3 + b6 * \mu4 = 0) \wedge (c7 + \tfrac{1}{2} * a7 * \mu3 + \tfrac{1}{2} * b7 * \mu4 = 0) \Rightarrow\\
&\ (-3 * x10^2 * x3 * y10 + x3 * y10^3 + x10^3 * y3 + (-3) * x10 * y10^2 * y3 = 0) \wedge\\
&\quad (-2 * x10 * x9 * y10 + x10^2 * y9 + (-1) * y10^2 * y9 = 0))
\end{aligned}
$$

,

Hence, essentially, we have to prove a universally quantified Boolean combination of polynomial equalities over the (complex) numbers.

22 of XXX

A Prover for Univ Quantified Boolean Combinations of Poly Equalities:

In the Theorema implementation of the prover, the proof method is explained in each proof generated by the prover.

```
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 y²) = 0))
          ∧ ((x² + -x y + x² y + -2 y² + -2 x y²) = 0))
 ]
```

The following call of the Groebner bases prover generates the proof of the above proposition completely automatically:

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

```
- ProofObject -
```

⏮ ◀ ▶ ⏭ 23 of XXX

Example of a Prover Based on Intermediate Principles: An Analysis Prover

The PCS method (BB 2001): A heuristic prove method for fomulae with

- variables for functions and for reals,
- involving notions whose definition needs "alternating quantifiers".

Structure of the method:

1. Use predicate logic rules (except "simplification" by =, ⇒, ⇔)	("Prove")
2. Simplify (using "semantic unification")	("Compute")
3. Solve inequalities.	("Solve")

⏮ ◀ ▶ ⏭ 24 of XXX

The main intermediate principle (black-box inference rule) used:

- an (exact) inequalities solver over the reals (e.g. Collins' algorithm).

Other intermediate principles used:

- simplification of arithmetical expressions to polynomial form.

Again, in the Theorema implementation of the prover, the proof method is explained in each proof generated by the prover.

⏮ ◀ ▶ ⏭ 25 of XXX

An Example

Definition["limit:", any[f, a],

$$\text{limit}[f, a] \Leftrightarrow \underset{\epsilon>0}{\underset{\epsilon}{\forall}}\ \underset{N}{\exists}\ \underset{n\geq N}{\underset{n}{\forall}}\ |f[n] - a| < \epsilon\Big]$$

Proposition["limit of sum", any[f, a, g, b],

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

```
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)]
```

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

```
Theory["limit"]
```

26 of XXX

Call of the Prover

The following call of the PCS prover generates the proof of the above proposition completely automatically:

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

```
- ProofObject -
```

27 of XXX

Some Details about Formal Groebner Bases Theory

The exploration of Groebner bases theory is (one of) my benchmarks for the support we obtain from formal math systems.

Go in layers (using the "form intermediate principles" and the "by intermediate principles" approach):

- from coefficient domains and power product domains
- to the polynomial ring (functor) with reduction relation,
- definition of Groebner bases (with or without reference to ideal theory - in which case set theory is needed),

- proof of the main theorem of algorithmic Groebner bases theory (the theorem on S-polynomials),
- algorithmic construction of Groebner bases,
- computation of Groebner bases,
- application of Groebner bases (for this, the various application areas must be formalized),
- application as a special prover for proving theorems in geometry.

I proposed this benchmark 1996 at the 1st Calculemus Meeting in Rome.

28 of XXX

Part of this program has been carried out:

- In Theorema:

  - Implementation of geo proofs generator using Groebner bases as an "intermediate inference rule".

  - Polynomial rings and "Groebner rings" as functors, computation in logic.

  - Implementation of inductive proofs generator that comes close to the inductive proofs needed in Groebner bases theory.

  - Automated invention of main theorem of Groebner bases theory (modulo an inductive proof generator).

  - Groebner bases algorithm cannot yet be "lifted" from the status of being knowledge to the status of being an inference rule. ("Reflection" necessary.)

- By L. Thery in Coq and by J.L. Ruiz-Reina et al. in ACL2:

  - Checked proof of main theorem of algorithmicGroebner bases theory (BB theorem on S-polys).

  - However, proofs need thousands of lines of user provided proof code.

  - Groebner bases algorithm cannot yet be "lifted" from the status of being knowledge to the status of being an inference rule.

- In Mizar (by C. Schwarzweller and P. Rudnicki) and Nuprl (by P.B. Jackson): Preparation towards proof checked algorithmic Groebner bases theory.

- H. Persson tried a Groebner bases algorithm synthesis in intuitionistic type theory but it is based on a fundamental misunderstanding of GB theory.

But still a long (?) way to go.

29 of XXX

Self-Elimination

By recent work on algorithmic algorithm synthesis, I am now able to automatically invent my own Groebner bases algorithm, in particular the central notion of S-polynomial.

This new synthesis method is based on the idea of "analysis of failing correctness proofs".

Details see:

B. Buchberger. Towards the Automated Synthesis of a Gröbner Bases Algorithm.

RACSAM (Review of the Royal Spanish Academy of Science), to appear.

⏮ ◀ ▶ ⏭ 30 of XXX

Input to the "lazy thinking" synthesis algorithm: The problem specification for Groebner bases construction.

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

Knowledge base contains "complete" knowledge on the auxiliary notion:

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

etc.

⏮ ◀ ▶ ⏭ 31 of XXX

Second input to the synthesis algorithm: algorithm scheme "Completion":

$$\begin{array}{l}
A[F] = A[F, \mathrm{pairs}[F]]\\
A[F, \langle\rangle] = F\\
\\
A[F, \langle\langle g1, g2\rangle, \bar{p}\rangle] =\\
\quad \mathrm{where}\Big[f = \mathrm{lc}[g1, g2],\ h1 = \mathrm{trd}[\mathrm{rd}[f, g1], F],\ h2 = \mathrm{trd}[\mathrm{rd}[f, g2], F],\\
\qquad \begin{cases} A[F, \langle\bar{p}\rangle] & \Leftarrow\ h1 = h2\\ A\Big[F \frown \mathrm{df}[h1, h2],\ \langle\bar{p}\rangle \asymp \Big\langle\langle F_k, \mathrm{df}[h1, h2]\rangle \underset{k=1,\ldots,|F|}{|}\Big\rangle\Big] & \Leftarrow\ \mathrm{otherwise}\end{cases}\Big]
\end{array}$$

The red algorithms are the unknown subalgorithms.

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

$$\underset{f,g}{\forall}\,(f \succeq \mathrm{rd}[f, g]).$$

⏮ ◀ ▶ ⏭ 32 of XXX

Output from the synthesis algorithm: a specification of the auxiliary algorithms lc and df:

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

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

⏮ ◀ ▶ ⏭ 33 of XXX

Summary

□

- I think that our reasoning systems must get to a significantly higher level of automation in order to become attractive for "working mathematicians".
- The resulting proofs must be presented in the form of proofs in ordinary math texts.
- Coherent theory exploration rather than isolated theorem proving must be in the foreground.
- "From" and "by" "Intermediate principles" play an important role in the structuring and automation process.

□

Bourbakism of 21st century:

- includes algorithmic mathematics # computer-free math
- allows the quick build-up of changing "views" of mathematical areas # one fixed view

⏮ ◀ ▶ ⏭ 34 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.

■ 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 2004]

B. Buchberger. The Four Parallel Threads of Formal Theory Exploration.

Technical Report of the SFB (Special Research Area) Scientific Computing, Johannes Kepler University, Linz, in preparation.

[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), to appear, 10 pages.