Computer-Assisted Proving by the PCS Method

Bruno Buchberger

Research Institute for Symbolic Computation
Johannes Kepler University, A4040 Linz, Austria

Abstract: In this paper we describe Theorema, a system for supporting mathematical proving by computer. The emphasis of the paper is on showing how, in many situations, proving can be reduced to solving in algebraic domains. We first illustrate this by known techniques, in particular the Gröbner bases technique. Then we go into the details of describing the PCS technique, a new technique that is particularly well suited for doing proofs in elementary analysis and similar areas in which the notions involved, typically, contain alternating quantifiers in their definitions. We conclude with a general view on the interplay between automated proving, solving, and simplifying.

1. Algebra, Algorithms, Proving

1.1 An Algebraic Notion: Gröbner Bases

We start with a typical algebraic definition, the definition of the concept of "Gröbner basis" which, in the notation of our Theorema system [Buchberger et al. 1997, 2000], looks like this:

$$\textbf{Definition}\Big["\text{Gröbner basis}", \text{any}[F],\ \text{is–Gröbner–basis}[F] :\Longleftrightarrow \forall_{f \in \text{Ideal}[F]}\ f \to_F^* 0\Big]$$

Here, $\to_F^*$ denotes the algorithmic reduction (division) relation on polynomials modulo sets F of multivariate polynomials. The notion of Gröbner basis turned out to be important for polynomial ideal theory because many ideal theoretic problems can be solved "easily" if Gröbner bases generating the ideals involved are known and, furthermore, many problems in other areas of mathematics can be reduced "easily" to ideal theoretic problems. (For the theory of Gröbner bases, originally introduced in [Buchberger 1965, 1970], see the textbooks on Gröbner bases, e.g. [Kreuzer, Robbiano 2000] that contains a list of all other current textbooks on Gröbner bases. Tutorials on the applications of Gröbner bases to fields other than polynomial ideal theory can be found in [Buchberger, Winkler 1998]. For example, it turned out that Gröbner bases have important applications also to fundamental problems in systems theory. A recent overview by J. Wood, [Wood 2000], lists the following problems of systems theory that can be solved by the Gröbner bases method:

elimination of variables,
computation of minimal left and right annihilators,

computation of controllable part; controllability test,
observability test,
computation of transfer matrix and minimal realization,
solution of the Cauchy problem for discrete systems,
testing for inclusion; addition of behaviors,
tests for zero/ weak zero / minor primeness; Bezout identity construction,
finite-dimensionality test,
computation of various sets of poles and zeros; polar decomposition,
achievability by regular interconnection,
computation of structure indices.

Because of the many possible applications, the algorithmic construction of Gröbner bases is of considerable importance. In the sequel, we will take the example of Gröbner bases as an example for illustrating various notions of "constructing" in mathematics. The paper will then focus on bringing together two levels of looking to mathematics: The object level, in which we develop an algorithmic theory like Gröbner bases theory using various (automated) proof techniques and the metalevel, in wich we apply algorithmic results from the object level, e.g. Gröbner bases theory, for further automation of proving in other areas of mathematics.

1.2 Inconstructive "Construction" of Gröbner Bases

The problem of "constructing" Gröbner bases can be cast into the form of the following theorem, which in Theorema notation looks as follows:

$$\textbf{Theorem}\Big["\text{Existence of Gröbner bases}",\ \forall_F\ \exists_G\ (\text{Ideal}[G] = \text{Ideal}[F] \wedge \text{is–Gröbner–basis}[G])\Big]$$

A first proof of this theorem could be given by just observing that G:=Ideal[F] has the required properties. Trivially, all polynomials f∈ Ideal[G] = Ideal[Ideal[F]] = Ideal[F] can be reduced to zero modulo G by just subtracting f. However, this is like a bad joke! The "construction" of the ideal generated by F is an infinite process. Thus, this proof of the theorem contains too little information for being algorithmically useful.

1.3 Inconstructive "Construction" of Finite Gröbner Bases

We now move one step further and consider the following, stronger, version of the theorem:

$$\textbf{Theorem}\Big["\text{Existence of finite Gröbner bases}",\ \forall_F\ \exists_G\ (\text{is–finite}[G] \wedge \text{Ideal}[G] = \text{Ideal}[F] \wedge \text{is–Gröbner–basis}[G])\Big]$$

The proof can be given in the following way: We consider the set

$$\text{Contour}[F] := \left\{ \text{LP}[f] \;\middle|\; f \in \text{Ideal}[F] \wedge \neg \exists_{g \in \text{Ideal}[F]} \text{LP}[g] \neq \text{LP}[f] \wedge \text{divides}[\text{LP}[g], \text{LP}[f]] \right\}.$$

(Here, LP[f] denotes the leading power product of the polynomial f.) By Dixon's lemma, this set is finite. Now it is easy to see that

$$G := \{ \text{Select}[t] \mid t \in \text{Contour}[F] \}$$

has the desired properties (where Select is a choice function that, for any power product $t \in$ Contour[F] selects a polynomial in Ideal[F] whose leading power product is t). This proof is much more constructive than the previous one and gives the additional information on the finiteness of G. Hower, although the final outcome of the construction is the finite set G, the construction itself is far from being an algorithm: For constructing Contour[F], one still has to consider the infinitely many polynomials f in the ideal generated by F and, for each such f, one has to check whether any of the infinitely many g in the same ideal satisfies a certain (algorithmic) condition. Also, Select is not an algorithmic function.

1.4 Algorithmic Construction of Finite Gröbner Bases

The algorithmic version of the problem can be formulated as follows:

$$\textbf{Theorem}\Big[\text{"Algorithmic construction of finite Gröbner bases"}, \exists_{\substack{GB \\ \text{is–algorithmic}[GB]}} \forall_F \; (\text{is–finite}[GB[F]] \wedge \text{Ideal}[GB[F]] = \text{Ideal}[F] \wedge \text{is–Gröbner–basis}[GB[F]])\Big]$$

It turns out that an algorithm GB can be established relatively easily as soon as the following theorem is proved:

$$\textbf{Theorem}\Big[\text{"Algorithmic characterization of Gröbner bases"}, \text{any}[F], \text{is–Gröbner–basis}[F] :\Longleftrightarrow \forall_{f,g \in F} \text{SP}[f, g] \to_F^* 0\Big]$$

Here, SP[f,g] is the "S-polynomial of f and g" defined by

$$\text{SP}[f,g] = u.f - v.g,$$

where the monomials u and v are chosen in such a way that u.LP[f]=v. LP[g]=LCM[LP[f],LP[g]]. Note that, if F is finite, the right-hand side of the theorem is an

algorithm! Hence, by this theorem, the inconstructive condition in the original definition of the notion of Gröbner basis can be replaced by an algorithmic condition! The power of the Gröbner basis method is contained entirely in this theorem and its proof, which was originally given in [Buchberger 1965, 1970].

The above theorem can now be turned into an algorithm GB for constructing Gröbner bases by considering all S-polynomials of polynomials in F and reducing them to normal forms. If all these reductions yield zero, by Theorem["Algorithmic characterization of Gröbner bases"], the original F is already a Gröbner basis. Otherwise, one has to adjoin the results of the reductions to the basis and repeat the process until it terminates. (Termination of this process can either be shown by Hilbert's basis theorem or, alternatively, by Dickson's lemma.) For details on the algorithm GB, see the textbooks on Gröbner bases, e.g. [Kreuzer, Robbiano 2000].

1.5 Reduction of Mathematical Problems to the Construction of Gröbner Bases

As a matter of fact, many fundamental problems in various mathematical areas con be reduced to the construction of Gröbner bases and, hence, become algorithmically solvable. In this paper, our focus is not on this well-known reduction of problems *inside* various mathematical areas but on the reduction of *proving about* various mathematical areas to the solution of algebraic problems as, for example, Gröbner bases construction. Thereby, proving in certain areas of mathematics becomes algorithmic. Looking further into the future, also proving *about* Gröbner bases theory may become algorithmic some day. We are currently working on this.

2. Reduce Proving to Algebra: Two Well-Known Methods

2.1 Proving Boolean Combinations of Equalities by the Gröbner Bases Method

The truth of a whole class of formulae can be decided by reducing the decision to the computation of Gröbner bases: This class consists of all universally quantified boolean combinations of equalities between arithmetical terms with variables ranging over the complex numbers (or any other algebraically closed field with algorithmic field operations). In fact, this class of formulae contains the hundreds of interesting geometrical theorems in algebraic formulation using coordinates. Here is an example of such a formula:

Formula["Test", any[x, y],

$$\begin{aligned}&((x^2 y - 3x) \neq 0) \vee ((xy + x + y) \neq 0) \vee \\ &\quad (((x^2 y + 3x = 0) \vee ((-2x^2 + -7xy + x^2 y + x^3 y + -2y^2 + -2xy^2 + 2x^2 y^2) = 0)) \\ &\wedge ((x^2 + -xy + x^2 y + -2y^2 + -2xy^2) = 0))\end{aligned}$$

"B1"]

By the following call, Theorema generates the proof of this formula together with explanatory text that, in fact, explains the general reduction method to Gröbner bases computations for the particular input formula:

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

The output of this call is the proof text in Appendix A1. that may well be sufficient for the reader to understand also the general method behind this reduction of proving to the construction of certain Gröbner bases. (The nested brackets at the right-hand margin of the proof text, in an on-line version of the proof, can be used to open and close entire sequences of cells for structured browsing of the proof text.)

2.2 Proving First-Order Formulae over the Reals by Cylindrical Algebraic Decomposition

Another well-known, and rather sophisticated, method for reducing proving to the solution of a certain algebraic problem is Collins' method for proving *first-order formulae over the reals* by cylindrical algebraic decomposition of $\mathbb{R}^n$. Collins' theorem on which the method hinges, essentially, states that, given a finite set of multivariate polynomials F, $\mathbb{R}^n$ decomposes into finitely many "cells" so that, on each of them, each $f \in F$ stays sign-invariant. Given F, Collins' algorithm then finds at least one "sample" point in each cell. In an oversimplified presentation, the truth of any first-order predicate formula over the reals can then be decided in the following way:

- First construct sample points in each cell in a sign-invariant decomposition of $\mathbb{R}^n$ for the set F of polynomials occurring in the formula.
- Partial formulae of the form $(\forall x \in \mathbb{R})\ P[x]$ can then be decided by checking $P[\alpha_1] \wedge \ldots \wedge P[\alpha_k]$ for the finitely many sample points α_i .
- Similarly, $(\exists x \in \mathbb{R})\ P[x]$ can be decided.

For the details, which are in fact quite involved, see the original paper [Collins 1975] and the recent collection of articles on Collins' method [Caviness, Johnson 1999]. Collins's algorithm, as the algorithm for constructing Gröbner bases, is now available in (some of) the mathematical software systems, like Mathematics, and can, hence, be routinely applied for proving.

3. Reduce Proving to Algebra: PCS, a New Method

3.1 The Purpose of the Method

We focus now on presenting a new method (the "PCS Method") for reducing proving in certain areas of mathematics to the solution of algebraic problems: This method is a heuristic method, which

- aims at automatically generating "natural" proofs that can easily be understood and checked by a human reader,

- is particularly suited for formulae involving concepts defined by formulae containing "alternating quantfiers" (i.e. alternations of the quantifiers $\forall$ and $\exists$), as this is typically the case in proofs in analysis, and
- allows the reduction of higher-order formulae (i.e. formulae containing variables on functions and predicates) to solving first-order formulae, in particular quantifier-free first-order formulae (that can then be solved, for example, by Collins' method).

The method is a heuristic method in the sense that we are not (yet) able to give a completeness result for the method. As a practical result we observed, however, that most of the typical propositions in elementary analysis textbooks (which are still considered to present quite some challenge for automated theorem proving) can be automaticall proved by the method and that, actually, these proofs can be found with very little search in the search space. Applications of the approach to areas in mathematics other than analysis, e.g. to areas involving set theory, are currently investigated, see [Windsteiger 2001].

3.2 An Example

The PCS method is best explained in an example. Consider, for example, the usual definition of "sequence f has limit a", which in Theorema notation looks as follows:

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

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

Note that 'f' is a function variable, i.e. a higher-order variable, as can be seen by its occurence in the term 'f[n]'. Thus, propositions involving this notion cannot be treated by first-order methods like Collins' method. A typical theorem on the notion 'limit' is, for example,

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

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

Of course, this proposition cannot be proved without further knowledge on the notions occurring in the proposition and the definition of limit as, for example, the notions '<' and '| |'. It turns out that, actually, very littel knowledge suffices for proving the proposition:

Definition["+:", any[f, g, x],

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

Lemma["|+|", any[x, y, a, b, δ, ϵ],

$$(|(x + y) - (a + b)| < (\delta + \epsilon)) \quad \Longleftarrow \quad (|x - a| < \delta \wedge |y - b| < \epsilon)]$$

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

In Theorema, we have language constructs for packing formulae into "knowledge bases" that can be nested to arbitrary depth. In our simple case, we define the theory "limit" by entering

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

Now, the following call will generate a proof of the proposition:

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

The proof generated is shown in Appendix A2. Note that the entire proof, including the explanatory English text, is found and displayed by the PCS method completely automatically, i.e. with no user interaction.

3.3 The "PCS" Method: Reduce Proving over Real Functions to Solving over Real Numbers

We now explain the essential parts of the PCS method by going through the example proof:

First, the goal of the proof and the available knowledge base is echoed.

P-phase ("predicate logic Proving"): In steps (1) and (2), the typical natural deduction steps of predicate logic are applied. However, all formulae involving equivalences, equalties, and implications are not yet used in this phase.

C-phase ("symbolic Computing"): Equivalences, equalities, and implications are used for "rewriting", i.e. "symbolic computation". Notably, definitions are used in this way. Thus, in the example, (3), (5), and (7) are produced.

In the C-phase, we also apply *Skolemization* on formulae in the knowledge base that start with ∀ ∃. This kind of "restricted Skolemization" is essential for being able, later, to construct solving terms for existential goals, see formula (16). Note that, in contrast to the resolution method, we do not at all expand the entire goal and knowledge base to clause form and do *not* apply Skolemization to the entire resulting formula. We think that the appropriate "dose" of Skolemization is essential for producing natural proofs. In the example, Skolemization is applied to (3) and (5) for producing (4) and (6).

Now we are back in a P-phase, in which we produce (8) and (9).

S-phase ("Solving"): Now, formula (9) is an existential formula. In such a situation we introduce "find constants" like ' N_2*' that allow us to speak about objects, which we do not yet know and, formally, to get rid of the existential quantifier so that we can go on with working on the inner parts of the goal formula.

Using a definition in a C-step, we arrive at (11). This is a crucial situation in the proof: The current goal formula is close to the Lemma["|+|"] in the knowledge base. However, we cannot obtain the goal from the lemma by substitution. Here, we apply what we call "semantic rewriting": Although there is no substitution for δ and ε such that $\delta+\varepsilon$ results in the constant ε_0 , one can state the the goal would be implied by knowing that there existed δ and ε such that $\delta+\varepsilon= \varepsilon_0$ and the premises of Lemma["|+|"] held. Thus, we are able to get over this situation on the expense of introducing additional existential quantifiers or, in other words, on the expense of having to find not only N_2^* but also a δ_0^* and an ε_1^*. In the example, semantic rewriting brings us from (11) over (12) to (13).

We are now again in a C-phase, in which a couple of rewriting steps using formulae in the knowledge base simplifies the "find problem" to the form (15) in which it now becomes apparent that, actually, the problem can be split into two independent "find problems":

- the problem of finding δ_0^* and ε_1^*, which is a problem that does not involve any higher order concepts (in our case, a problem on real numbers) and
- the problem of finding N_2^*, which can be solved by pure predicate logic as soon as δ_0^* and an ε_1^* are known.

The problem of finding δ_0^* and ε_1^* can be solved by calling any algebraic algorithm for solving equations and inequalities over the reals, for example by Collins' method. The method does not only return one particular solution but (either reports that no solution exists or) returns a general solution: In our example it suggests that one may take any δ_0^* between 0 and ε_0 (which is > 0 by assumption (8)) and $\varepsilon_1^* = \varepsilon_0 - \delta_0^*$.

Finally, the solving term

$$\max[\, N_0\,[\delta_0^*],\ N_1\,[\,\varepsilon_0 - \delta_0^*]]$$

for N_2^* is produced. Note that this term contains much more constructive information than is usually given in proofs presented in textbooks of elementary analysis: It tells us that, if we know a function N_0 that gives the index bound for f in dependence on the distance between sequence values and limit and similarly a function N_1 for g, we can explicitly compute the index bound for the sum sequence f+g: If we want to be sure that (f+g)[n] stays closer to a+b than a given $\varepsilon_0>0$, take any δ_0^* between 0 and ε_0 and take

$n>\max[N_0 [\delta_0^*], N_1 [\varepsilon_0 - \delta_0^*]]$. In case N_0 and N_1 are algorithms, this procedure is an algorithm. In other words, the above proof does not only prove the theorem but it automatically synthesizes an algorithm for the index bound of the sum of sequences.

3.4 A General View

The PCS method heavily interrelates proving, solving, and simplifying (computing). In fact, we believe that the interrelation of proving, solving, and simplifying is fundamental for the future of algorithmic mathematics. Proving, solving, and simplifying seem to be the three basic mathematical activities that, roughly, correspond to how the free variables in formulae are treated:

An algorithm $\mathcal{A}$ is a *prover* for the theory T iff, for all formulae F and knowledge bases K,

if $\mathcal{A}$[F,K] = "proved" then $T \cup K \models (\forall x)\ F$
(where x are the free variables in F).

(Complete provers would have to produce "proved" iff $T \cup K \models (\forall x)\ F$.)

An algorithm $\mathcal{A}$ is a *solver* for the theory T iff, for all F and K,

if $T \cup K \models (\exists x)\ F$ then $T \cup K \models (\forall y)\ \mathcal{A}[F,K] \circ F$

(where x are the free variables in F and y are the free variables in the substitution $\mathcal{A}$[F,K] which has the form $x \rightarrow S$, where S is a term).

An algorithm $\mathcal{A}$ is a *simplifier* for the theory T iff, for all F and K,

$T \cup K \models (\forall x)\ (\mathcal{A}[F,K] \Leftrightarrow F)$ (and $\mathcal{A}$[F,K] "is simpler than" F)
(where x are the free variables in F).

Accordingly, Theorema is a (growing) library of

- *elementary* provers, solvers, simplifiers for various theories and
- *reduction* methods that reduce proving, solving, simplifying in various theories to proving, solving, simplifying in other theories

together with tools for producing explanatory text and nice syntax and managing mathematical knowledge bases. Theorema is programmed in Mathematica and also heavily uses the front-end of Mathematica. Hence, Theorema runs on all platforms on which Mathematica is available. In the future, the interplay between proving, solving, and simplifying will be made the main structuring design feature of Theorema. As a consequence, we will give the user more explicit control over this interplay.

4. References

[Buchberger 1965, 1970] B. Buchberger. An Algorithmic Criterion for the Solvability of Algebraic Systems of Equations (German). *Aequationes mathematicae* 4/3, pp. 374-383, 1970. (English translation in: [Buchberger, Winkler 1998], pp. 535-545.) This is the journal publication of the PhD thesis: B. Buchberger, *On Finding a Vector Space Basis of the Residue Class Ring Modulo a Zero Dimensional Polynomial Ideal* (German), University of Innsbruck, Austria, 1965.

[Buchberger et al. 1997] B. Buchberger, T. Jebelean, F. Kriftner, M. Marin, D. Vasaru, 1997. An Overview on the Theorema project. In: *Proceedings of ISSAC'97* (International Symposium on Symbolic and Algebraic Computation, Maui, Hawaii, July 21-23, 1997), W. Kuechlin (ed.), ACM Press 1997, pp. 384-391.

[Buchberger et al. 2000] B. Buchberger, C. Dupre, T. Jebelean, F. Kriftner, K. Nakagawa, D. Vasaru, W. Windsteiger, 2000. The Theorema Project: A Progress Report. In: *8th Symposium on the Integration of Symbolic Computation and Mechanized Reasoning*, St. Andrews, Scotland, August 6-7, M. Kerber and M. Kohlhase (eds.), available from Fachbereich Informatik, Universität des Saarlandes, Germany, pp. 100-115.

[Buchberger, Winkler 1998] B. Buchberger, F. Winkler (eds.). *Gröbner Bases and Applications*. Proc. of the International Conference "33 Years of Groebner Bases", London Mathematical Society Lecture Note Series, 251, Cambridge University Press, 552 p.

[Caviness, Johnson 1998] B. F. Caviness, J. Johnson (eds.) *Quantifier Elimination and Cylindrical Algebraic Decomposition*. Springer, Wien – New York, 1998, 431 p.

[Collins 1975] G. E. Collins. Quantifier Elimination for the Elementary Theory of Real Closed Fields by Cylindrical Algebraic Decomposition. In: H. Brakhage (ed.), *Automata Theory and Formal Languages*, Lecture Notes in Computer Science 33, pp. 134-183. Reprinted in [Caviness, Johnson 1998], pp. 85-121.

[Kreuzer, Robbiano 2000] M. Kreuzer, L. Robbiano. *Computational Commutative Algebra 1*. Springer, Berlin – Heidelberg – New York, 2000, 321 p.

[Windsteiger 2001] W. Windsteiger. *A Set Theory Prover Within Theorema*. PhD Thesis, RISC Institute, University of Linz, Austria, to appear, 2001.

[Wood 2000] J. Wood. Modules and Behaviours in nD Systems Theory. *Multidimensional Systems and Signal Processing* 11, pp. 11-48, 2000.