RACSAM
Rev. R. Acad. Cien. Serie A. Mat.
VOL. **98** (1), 2004, pp. 65–75
Ciencias de la Computación / Computational Sciences

Towards the Automated Synthesis of a Gröbner Bases Algorithm

Bruno Buchberger

Abstract. We discuss the question of whether the central result of algorithmic Gröbner bases theory, namely the notion of S–polynomials together with the algorithm for constructing Gröbner bases using S–polynomials, can be obtained by "artificial intelligence", i.e. a systematic (algorithmic) algorithm synthesis method. We present the "lazy thinking" method for theorem and algorithm invention and apply it to the "critical pair / completion" algorithm scheme. We present a road map that demonstrates that, with this approach, the automated synthesis of the author's Gröbner bases algorithm is possible. Still, significant technical work will be necessary to improve the current theorem provers, in particular the ones in the Theorema system, so that the road map can be transformed into a completely computerized process.

Hacia la síntesis automática de un algoritmo de bases de Gröbner

Resumen. Se aborda la cuestión de si el resultado central de la teoría algorítmica de bases de Gröbner, es decir, la noción de S–polinomio, junto con el algoritmo de construcción de bases de Gröbner basado en S-polinomios, puede obtenerse mediante la "inteligencia artificial", es decir, por un método sistemático de síntesis algorítmica. En concreto, se presenta el método "lazy thinking" para la invención de teoremas y algoritmos, que se aplica al esquema algorítmico de "par crítico/completitud". Se presenta una "hoja de ruta" que demuestra que este enfoque permite la síntesis automática del algoritmo de bases de Gröbner del autor. No obstante, será necesario mejorar los actuales demostradores de teoremas y, sobre todo, los del sistema "Theorema", para que esa "hoja de ruta" se pueda transformar en un proceso completamente computerizado, lo que aún supondrá un trabajo técnico importante.

1. Introduction

In [7, 8] we proposed a method (the "lazy thinking" method) for the automated invention of theorems and algorithms. This method is embedded into a general research plan for automating or, at least, computer-supporting the process of mathematical theory exploration ("mathematical knowledge management") and the implementation of this research plan in the form of the Theorema software system, see [10]. The automated synthesis of simple theorems and algorithms by the lazy thinking method in the frame of Theorema has been demonstrated to be possible, see [7, 8, 11]. The question is how far this approach can carry us. As a kind of non-trivial benchmark, we propose the algorithm in [3] for the construction of Gröbner bases. By now, this algorithm is routinely available in all current mathematical software systems like, for example, Mathematica, see [20], and its theoretical foundation, implementation details, and numerous applications are well documented in the literature, see for example [6, 1], and [15]. However, automatic invention

Presentado por Luis M. Laita.
Recibido: February 29, 2004. *Aceptado*: October 13, 2004.
Palabras clave / Keywords: Groebner Basis, Algorithm Syntesis and Analysis, Symbolic Computation, Automated Reasoning.
Mathematics Subject Classifications: 13P10, 68W30, 68W40, 68N30, 68T15, 68T27.

(synthesis) of this algorithm seems to be far beyond the current capabilities of automated theorem proving and algorithm synthesis techniques although significant progress has been made in the automated verification of the proof of my Gröbner bases algorithm by proof checkers, see [19] and [16]. In this paper, we will demonstrate that an automated synthesis of this (and similar) algorithms along the lines of our "lazy thinking" synthesis method *is* possible. Hence, this is the first time in the literature that a Gröbner bases algorithm is synthesized automatically. Some technical work still has to be done in order to make our concrete automated proving systems, which are an important ingredient in our approach to algorithm synthesis, sufficiently powerful.

2. The Fundamental Problem of Gröbner Bases Theory

We first present the essential ingredients of the Gröbner bases theory introduced in [3]. We present the theory of Gröbner basis in a top-down style starting with the fundamental problem of constructing Gröbner bases, which is the main subject of this paper: We want to find an algorithm Gb such that

$$\forall_F \ (\texttt{is-finite-Gröbner-basis}[F,\ \texttt{Gb}[F]]).$$

Here and in the sequel, F, G range over sets of multivariate polynomials in a fixed number of indeterminates over a fixed coefficient field. Furthermore, f, g, h range over polynomials, p, q, r range over power products, a, b, c range over coefficients, and i, j, k range over integers. f^* etc. range over finite sequences of polynomials, etc. (All formulae in this paper are given in Theorema notation, i.e. they can be processed by the Theorema system, see [10].)

The binary predicate is-finite-Gröbner-basis is defined as follows:

$$\texttt{is-finite-Gröbner-basis}[F,\ G] :\Longleftrightarrow \begin{cases} \texttt{is-finite}[G] \\ \texttt{is-Gröbner-basis}[G] \\ \texttt{ideal}[F] = \texttt{ideal}[G] \end{cases}.$$

(For "is-finite-Gröbner-basis$[F,\ G]$" read "G is a finite Gröbner basis for F".)

Now we define the ingredient concepts: First,

$$\texttt{ideal}[F] := \{f \mid f \equiv_F 0\}.$$

(For "ideal[F]" read "the ideal generated by F".)

Here,

$$(h_1 \equiv_F h_2) :\Longleftrightarrow \exists_{h^*,\ f^*} \left(\begin{cases} |h^*| = |f^*| \\ \forall_{i=1,\ldots,|f^*|}\ (f^*)_i \in F \\ h_1 = h_2 + \sum_{i\ =\ 1,\ldots,|f^*|} (h^*)_i (f^*)_i \end{cases} \right).$$

(For "$h_1 \equiv_F h_2$" read "h_1 is congruent to h_2 modulo F". Here, f^* and h^* denote finite sequences of polynomials. $|f^*|$ denotes the length of f^* and $(f^*)_i$ is the i^{th} element of f^*.)

Second,

$$\texttt{is-Gröbner-basis}[G] \Longleftrightarrow \texttt{is-Church-Rosser}[\rightarrow_G].$$

(For "is-Gröbner-basis$[G]$" read "G is a Gröbner basis". Note that, in this paper, following the syntactic conventions in Theorema, see [10], a predicate or function constant like is-Gröbner-basis may occur with various different arities).

Here, $\rightarrow_G$ is the "reduction relation induced by G", which is defined as follows:

$$(h_1 \rightarrow_G h_2) :\Longleftrightarrow \exists_{g \in G} \left(\begin{cases} \texttt{lp}[g] \mid \texttt{lp}[h_1] \\ h_2 = h_1 - (\texttt{lm}[h_1]/\texttt{lm}[g])g \end{cases} \right),$$

where $\mathtt{lm}[f]$ is "the leading monomial of f", $\mathtt{lp}[f]$ is "the leading power product of f", p/q is "the quotient of the two power products p and q", and $p \mid q$ stands for "p divides q". Note that $\mathtt{lm}[f]$ and $\mathtt{lp}[f]$ are defined w.r.t. a fixed "admissible" total ordering $\succ$ of the power products - e.g. the lexical ordering or the total degree lexical ordering - which can then be extended to a partial ordering on the polynomials in a natural way. Admissible orderings are always Noetherian.

Finally, we recall the definition of the Church–Rosser property for binary relations $\rightarrow$ in any domain:

$$\mathtt{is\text{-}Church\text{-}Rosser}[\rightarrow] :\Longleftrightarrow \forall_{f_1,\ f_2}\ \ (f_1 \leftrightarrow^* f_2 \Rightarrow f_1 \downarrow^* f_2).$$

Here $\rightarrow^*$ is the reflexive, transitive closure of $\rightarrow$, $\leftrightarrow^*$ is the reflexive, symmetric, transitive closure of $\rightarrow$, and $\downarrow^*$ is defined as follows:

$$(f \downarrow^* g) :\Longleftrightarrow \exists_h\ \ (f \rightarrow^* h \leftarrow^* g).$$

(For "$f \downarrow^* g$" read "f and g have a common successor".)

3. An Algorithm for the Construction of Gröbner Bases

In [3] an algorithm $\mathtt{Gb}$ that meets the specification

$$\forall_F\ (\mathtt{is\text{-}finite\text{-}Gr\ddot{o}bner\text{-}basis}[F,\ \mathtt{Gb}[F]])$$

was introduced. This algorithm is based on the following theorem in [3]: Let G be finite, then

$$\mathtt{is\text{-}Gr\ddot{o}bner\text{-}basis}[G] \Longleftrightarrow$$

$$\forall_{g_1,g_2 \in G}\ \text{where}[f = \mathtt{lcm}[\mathtt{lp}[g_1], \mathtt{lp}[g_2]], \mathtt{trd}[\mathrm{rd}[f, g_1], G] = \mathtt{trd}[\mathtt{rd}[f, g_2], G]].$$

Here, $\mathtt{lcm}[p,q]$ is the least common multiple of p and q. Furthermore, for finite F,

$$\mathtt{rd}[g,f] = \begin{cases} g - (\mathtt{lm}[g]/\mathtt{lm}[f])f & \Leftarrow \mathtt{lp}[f] \mid \mathtt{lp}[g] \\ g & \Leftarrow \text{otherwise,} \end{cases}$$

$$\mathtt{trd}[g,F] = \mathtt{trd}[g,F,1],$$

$$\mathtt{trd}[g,F,k] = \begin{cases} g & \Leftarrow k > |F| \\ \begin{cases} \mathtt{trd}[\mathtt{rd}[g,F_k],F,1] & \Leftarrow \mathtt{rd}[g,F_k] \prec g \\ \mathtt{trd}[g,F,k+1] & \Leftarrow \text{otherwise} \end{cases} & \Leftarrow \text{otherwise.} \end{cases}$$

F is supposed to be represented as finite sequence, so that F_k refers to the k^{th} element of F. The algorithm $\mathtt{rd}$ reduces a polynomial g modulo a polynomial f to a polynomial $\prec g$ iff $\mathtt{lp}[f] | \mathtt{lp}[g]$. For "$\mathtt{rd}[g,\ f]$" read "the result of reducing g modulo f in one step". The algorithm $\mathtt{trd}$ applies the step $\mathtt{rd}$ iteratively modulo all polynomials in F until it arrives at a polynomial that cannot be further reduced modulo F. Note, also, that this algorithm always terminates because the ordering $\succ$ is Noetherian. For "$\mathtt{trd}[g,F]$" read "the result of totally reducing g modulo F".

Note that the above criterion for deciding the Gröbner basis property is *algorithmic* (in case G is finite): Just consider the finitely many pairs g_1, g_2 of polynomials in G and check whether or not the two reductions described in the criterion yield identical results.

(Usually, the criterion is given in the following form:

$$\mathtt{is\text{-}Gr\ddot{o}bner\text{-}basis}[G] \Longleftrightarrow \forall_{g_1,g_2 \in G}\ (\mathtt{trd}[\mathtt{sp}[g_1,g_2],G] = 0),$$

where

$$\mathtt{sp}[g_1, g_2] = \text{where}[f = \mathtt{lcm}[\mathtt{lp}[g_1], \mathtt{lp}[g_2]], \mathtt{rd}[f, g_1] - \mathtt{rd}[f, g_2]].$$

$\mathtt{sp}[g_1, g_2]$ is called the "S–polynomial of g_1 and g_2", whereas the two polynomials $\mathtt{rd}[f, g_1]$ and $\mathtt{rd}[f, g_2]$ are called the "critical pair of g_1 and g_2". In this paper, we use the criterion in the first form because it lends itself better to the generalization to an "algorithm scheme" that is applicable in many domains, even in domains in which we do not have a zero and a substraction operation.)

The transition from this *algorithmic criterion* to an *algorithm* Gb for *constructing* Gröbner bases (for finite inputs F) is not difficult any more and was also introduced in [3]:

$$\mathtt{Gb}[F] = \mathtt{Gb}[F, \mathtt{pairs}[F]]$$

$$\mathtt{Gb}[F, \langle\ \rangle] = F$$

$$\mathtt{Gb}[F, \langle\langle g_1, g_2\rangle, \overline{p}\ \rangle] =$$

$$\text{where}[f = \mathtt{lcm}[\mathtt{lp}[g_1], \mathtt{lp}[g_2]], h_1 = \mathtt{trd}[\mathtt{rd}[f, g_1], F], h_2 = \mathtt{trd}[\mathtt{rd}[f, g_1], F],$$

$$\begin{cases} \mathtt{Gb}[F, \langle\overline{p}\ \rangle] & \Longleftarrow h_1 = h_2 \\ \mathtt{Gb}[F \frown (h_1 - h_2), \langle\overline{p}\ \rangle \asymp \big\langle\langle F_k, h_1 - h_2\rangle\ \big|_{k=1,\ldots,|F|}\big\rangle] & \Longleftarrow \mathit{otherwise} \end{cases}].$$

Here, pairs$[F]$ is the tuple of all pairs of elements in F. The overbarred variables like $\overline{p}$, in Theorema notation, stand for arbitrarily many elements, see [10]. $F \frown h$ is F with h appended and $\asymp$ is the notation for concatenation. Roughly, the algorithm *Gb* checks for all pairs of polynomials g_1, g_2 in F whether the results h_1 and h_2 of the total reduction of $\mathtt{rd}[\mathtt{lcm}[\mathtt{lp}[g_1], \mathtt{lp}[g_2]], g_1]$ and $\mathtt{rd}[\mathtt{lcm}[\mathtt{lp}[g_1], \mathtt{lp}[g_2]], g_2]$, respectively, are identical. If $h_1 \neq h_2$ then the difference $h_1 - h_2$ is added to the basis and the process is repeated. (Termination is guaranteed by Dickson's lemma, see [13].)

¿From the perspective of algorithm synthesis, the question now is: By which systematic (and hopefully algorithmic) methods can one invent a criterion and/or a construction algorithm of the above type? In other words, can the invention of the essential notion of algorithmic Gröbner bases theory, namely the notion of "S–polynomial" (or, equivalently, "critical pair") together with the fundamental theorem on the relation between Gröbner bases and S-polynomials, be obtained by a systematic (algorithmic) process on the meta–level of mathematics? Again in other words, can the invention of the notion of S–polynomial and the pertinent theorem be obtained by "artificial intelligence"? Before we discuss this question, we provide a brief summary of our "lazy thinking" method for theorem and algorithm synthesis introduced in [7].

4. The Lazy Thinking Approach for Theorem and Algorithm Synthesis

The problem consists in synthesizing an algorithm A that satisfies the specification ("correctness theorem")

$$\forall_x\ \mathtt{P}[x, \mathtt{A}[x]],$$

where P is the given problem specification. (We assume that we are given a knowledge base, i.e. a collection of true statements about P and the auxiliary functions and predicates needed for the definition of P.)

In the "lazy thinking" synthesis approach, we first choose an algorithm scheme for A from a library of algorithm schemes. An algorithm scheme is a formula in which the unknown function A is (recursively) defined in terms of unknown auxiliary operations and A. Now we substitute the scheme chosen into the correctness theorem and start an attempt to prove the theorem. Typically, this proof will fail because nothing is known on the auxiliary operations. Next, we analyze the failing proof and, by some algorithmic heuristics (which is explained in [8]) generate requirements (specifications) on the auxiliary operations that would make the proof of the correctness theorem work.

By this, we reduce the synthesis of an algorithm A, that meets the given specification P, to the synthesis of algorithms for auxiliary operations that meet the *generated* specifications. This recursive synthesis process stops when we arrive at specifications for auxiliary operations which are met by operations which are already available in the knowledge base. This synthesis method is automatic in as much as the proving process and the requirements generation process are automated. For examples of a completely automated synthesis of algorithms using this "lazy thinking" approach see [8] and [11].

Mutatis mutandis, this lazy thinking approach can also be applied to the synthesis (invention) of theorems (lemmata): If we want to prove a theorem `T` from a knowledge base of statements which we presuppose on the operations occurring in `T` we may apply an (automated) proof method until we will be stuck at a failing proof situation. In such a situation, by a systematic (algorithmic) lemmata invention method, we conjecture a lemma `L` whose truth could make the proof of `T` succeed. An attempt is then started for proving `L`, which may either succeed in which case we return to the proof of the main theorem `T` or it may again be stuck at a failing proof situation in which case we again call the lemma conjecture algorithm. In this way, we generate a "cascade" of partial proofs which may finally result in a proof of the initial theorem and, at the same time, will generate a hierarchy of additional lemmata, see [7] for an example.

5. On the Way to Synthesizing a Gröbner Bases Algorithm

Applying the lazy thinking algorithm synthesis paradigm to the synthesis of a Gröbner bases algorithm, we could now choose various algorithm schemes from a library of fundamental algorithm schemes, like "divide–and–conquer", "interpolation", "projection" etc. and set up the corresponding correctness proof attempts. (In fact, it would be an interesting study to investigate which one of these schemes leads to a feasible algorithm for the Gröbner bases problem. This study would be particularly interesting given the fact that, so far, no algorithm essentially different from the author's "critical pair / completion" algorithm has been found for the Gröbner bases construction problem.) The most promising algorithm scheme candidate is the "critical pair / completion scheme", which was distilled from the author's Gröbner bases algorithm, the Knuth–Bendix algorithm [14], Robinson's resolution algorithm [18], and other algorithms and informally formulated in [5].

Here, we propose the following formal presentation of this scheme, which 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:

$$\forall_{f,g} \;\; (f \geq \mathtt{rd}[f,\, g]).$$

Given a reduction operation `rd`, we can define `tdr` (total reduction) in exactly the way shown above for the special case of polynomial reduction and, by the above property of `rd`, it is clear that this algorithm always terminates.

Then the critical pair completion algorithm scheme is as follows:

$$\begin{aligned}
&\mathtt{Gb}[F] = \mathtt{Gb}[F, \mathtt{pairs}[F]]\\
&\mathtt{Gb}[F, \langle\,\rangle] = F\\
&\mathtt{Gb}[F, \langle\langle g_1, g_2\rangle, \overline{p}\,\rangle] =\\
&\quad \text{where}[f = \mathtt{lc}[g_1, g_2],\; h_1 = \mathtt{trd}[\mathtt{rd}[f, g_1], F],\; h_2 = \mathtt{trd}[\mathtt{rd}[f, g_2], F],\\
&\quad \left\{\begin{array}{ll} \mathtt{Gb}[F, \langle\overline{p}\,\rangle] & \Longleftarrow h_1 = h_2 \\ \mathtt{Gb}[F \frown \mathtt{df}[h_1, h_2],\; \langle\overline{p}\,\rangle \asymp \big\langle\langle F_k, \mathtt{df}[h_1, h_2]\rangle \,\big|_{k=1,\ldots,|F|}\big\rangle] & \Longleftarrow \text{otherwise} \end{array}\right.].
\end{aligned}$$

with *unknown* auxiliary functions `lc` and `df` where, in addition, we require that

$$\begin{aligned}
&\mathtt{rd}[\mathtt{lc}[g_1, g_2], g_1] \prec \mathtt{lc}[g_1, g_2],\\
&\mathtt{rd}[\mathtt{lc}[g_1, g_2], g_2] \prec \mathtt{lc}[g_1, g_2].
\end{aligned}$$

(A more general scheme that does not include this additional requirement could be formulated but, in this paper, we don't want to introduce additional technical complications which we do not have in the case of the polynomial domain.)

The problem of synthesizing a Gröbner bases algorithm can now be also stated by asking whether we can *automatically* arrive at the idea that

$$\mathtt{lc}[g_1,\ g_2]\ =\ \mathtt{lcm}[\mathtt{lp}[g_1],\ \mathtt{lp}[g_2]]$$

and

$$\mathtt{df}[h_1,\ h_2]\ =\ h_1\ -\ h_2$$

are suitable functions that specialize the algorithm scheme to an algorithm that constructs a Gröbner basis for the input F.

Using this scheme, we now go into the correctness proof of

$$\forall_F\ (\mathtt{is\text{-}finite\text{-}Gröbner\text{-}basis}[F,\ \mathtt{Gb}[F]]).$$

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

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

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

$\mathtt{is\text{-}finite}[G]$,

$\mathtt{ideal}[F]\ =\ \mathtt{ideal}[G]$,

and

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

For lack of space in this paper, we only deal with the third, most important, property. (Finiteness of G can again be proved by Dickson's Lemma *after* lc and df will have been synthesized! The property $\mathtt{ideal}[F]=\mathtt{ideal}[G]$ is of course important but, from the methodological point of view, does not add any more challenges to the synthesis.)

In principle, Noetherian induction on polynomials w.r.t. the Noetherian ordering $\succ$ would be an appropriate proof method for proving $\mathtt{is\text{-}Church\text{-}Rosser}[\rightarrow_G]$. However, Newman's Lemma, [17], for all Noetherian reduction relations $\rightarrow$ replaces this induction by showing, once and for all, that the following equivalence

$$\mathtt{is\text{-}Church\text{-}Rosser}[\rightarrow]\Longleftrightarrow\forall_{f,f_1,f_2}\left(\left(\left\{\begin{array}{l} f\rightarrow f_1 \\ f\rightarrow f_2\end{array}\right.\right)\Rightarrow f_1\downarrow^* f_2\right)$$

is true. Using Newman's Lemma, the correctness proof (attempt) now proceeds as follows (using various elementary properties of the arithmetical operations and the reduction operation on polynomials, which we cannot repeat in this short paper, see any of the textbooks on Gröbner basis or [6]). First, we observe that, using the elementary properties of reduction, we can simplify the test for checking the Church–Rosser property further by proving that the first universal quantifier needs only range over all power products, i.e. by proving

$$\texttt{is-Church-Rosser}[\to_G] \iff \forall_p\ \forall_{f1,f2}\ \left(\left(\left\{\begin{array}{l} p \to f_1 \\ p \to f_2 \end{array}\right.\right) \Rightarrow f_1 \downarrow^* f_2\right).$$

Let now the power product p and the polynomials f_1, f_2 be arbitrary but fixed and assume

$$\begin{array}{l} p \to_G f_1, \\ p \to_G f_2. \end{array}$$

We have to find a polynomial g such that

$$\begin{array}{l} f_1 \to_G^* g, \\ f_2 \to_G^* g. \end{array}$$

¿From the assumption we know that there exist polynomials g_1 and g_2 in G such that

$$\begin{array}{l} \texttt{lp}[g_1] | p, \\ f_1 = \texttt{rd}[p,\ g_1], \\ \\ \texttt{lp}[g_2] \mid p, \\ f_2 = \texttt{rd}[p,\ g_2]. \end{array}$$

¿From the final situation in the algorithm scheme we know that for these g_1 and g_2

$$\bigvee \left\{\begin{array}{l} h_1 = h_2 \\ \texttt{df}[h_1,\ h_2] \in G, \end{array}\right.$$

where

$$\begin{array}{l} h_1 := \texttt{trd}[f_1',\ G],\ f_1' := \texttt{rd}[\texttt{lc}[g_1,\ g_2],\ g_1], \\ h_2 := \texttt{trd}[f_2',\ G],\ f_2' := \texttt{rd}[\texttt{lc}[g_1,\ g_2],\ g_2]. \end{array}$$

Case $h_1 = h_2$: In this case

$$\begin{array}{l} \texttt{lc}[g_1,\ g_2] \to_G \texttt{rd}[\texttt{lc}[g_1,\ g_2],\ g_1] \to_G^* \texttt{trd}[\texttt{rd}[\texttt{lc}[g_1,\ g_2],\ g_1],\ G] = \\ \texttt{trd}[\texttt{rd}[\texttt{lc}[g_1,\ g_2],\ g_2],\ G] \leftarrow_G^* \texttt{rd}[\texttt{lc}[g_1,\ g_2],\ g_2] \leftarrow_G \texttt{lc}[g_1,\ g_2]. \end{array}$$

Hence, by elementary properties of polynomial reduction,

$$\begin{array}{l} \forall_{a,\ q}\ \Big(a\,q\,\texttt{lc}[g_1,\ g_2] \to_G a\,q\,\texttt{rd}[\texttt{lc}[g_1,\ g_2],\ g_1] \to_G^* a\,q\,\texttt{trd}[\texttt{rd}[\texttt{lc}[g_1,\ g_2],\ g_1],\ G] = \\ \quad a\,q\,\texttt{trd}[\texttt{rd}[\texttt{lc}[g_1,\ g_2],\ g_2],\ G] \leftarrow_G^* a\,q\,\texttt{rd}[\texttt{lc}[g_1,\ g_2],\ g_2] \leftarrow_G a\,q\,\texttt{lc}[g_1,\ g_2]\Big). \end{array}$$

Now we are stuck in the proof.

However, looking at all the temporary assumptions which we have now, and using the requirements conjecturing heuristics described in [8], we see that we could proceed successfully with the proof if $\texttt{lc}[g_1,\ g_2]$ would satisfy the following requirement

$$\begin{array}{l} \forall_{p,\ g_1,\ g_2}\ \left(\left(\left\{\begin{array}{l} \texttt{lp}[g_1] \mid p \\ \texttt{lp}[g_2] \mid p \end{array}\right.\right) \Rightarrow \left(\exists_{a,\ q}\ (p = a\,q\,\texttt{lc}[g_1,\ g_2])\right)\right), \\ \forall_{g_1,\ g_2}\ \left(\left\{\begin{array}{l} \texttt{lp}[g_1] \mid \texttt{lc}[g_1, g_2] \\ \texttt{lp}[g_2] \mid \texttt{lc}[g_1, g_2] \end{array}\right.\right). \end{array} \qquad \text{(lc requirement)}$$

With such an lc, we then would have

$$\begin{aligned}
&p \to_G \ \mathtt{rd}[p,\ g_1] \ = \\
&\quad = a\, q\, \mathtt{rd}[\mathtt{lc}[g_1,\ g_2],\ g_1] \ \to_G^* \ a\, q\, \mathtt{trd}[\mathtt{rd}[\mathtt{lc}[g_1,\ g_2],\ g_1],\ G] \ = \\
&\quad = a\, q\, \mathtt{trd}[\mathtt{rd}[\mathtt{lc}[g_1,\ g_2],\ g_2],\ G] \ \leftarrow_G^* \ a\, q\, \mathtt{rd}[\mathtt{lc}[g_1,\ g_2],\ g_2] \ = \\
&= \mathtt{rd}[p,\ g_2] \ \leftarrow_G \ p
\end{aligned}$$

and, hence,

$$\begin{aligned}
&f_1 \ \to_G^* \ a\, q\, \mathtt{trd}[\mathtt{rd}[\mathtt{lc}[g_1,\ g_2],\ g_1],\ G], \\
&f_2 \ \to_G^* \ a\, q\, \mathtt{trd}[\mathtt{rd}[\mathtt{lc}[g_1,\ g_2],\ g_1],\ G],
\end{aligned}$$

i.e. we have found a suitable g.

Note that the (lc requirement) is now completely independent of the Gröbner bases construction problem from which we started! Of course, it is now easy to find an lc that satisfies (lc requirement), namely

$$\mathtt{lc}[g_1,\ g_2] \ = \ \mathtt{lcm}[\mathtt{lp}[g_1],\ \mathtt{lp}[g_2]].$$

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

Case $\mathtt{df}[h_1,\ h_2] \in G$:
In this part of the proof we are basically stuck right at the beginning.

As a new requirement generation technique (which we did not yet consider in [8]), we can try to reduce this case to the first case, which would generate the following requirement

$$\forall_{h_1,\ h_2} \ h_1 \downarrow^*_{\{\mathtt{df}[h_1,\ h_2]\}} h_2 \quad \text{(df requirement)}.$$

Looking to the knowledge base of elementary properties of polynomial reduction, it is now easy to find a function df that satisfies (df requirement), namely

$$\mathtt{df}[h_1,\ h_2] \ = \ h_1 \ - \ h_2$$

because, in fact,

$$\forall_{f,\ g} \ f \ \downarrow^*_{\{\text{f-g}\}} \ g.$$

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

6. Conclusion

We sketched how the author’s Gröbner bases algorithm, which hinges on the crucial concept of S–polynomials can be synthesized “automatically”, i.e. how the most essential concept of algorithmic Gröbner bases theory, namely the concept of S–polynomial (or, equivalently, the concept of “critical pair”) can be invented “automatically”. Here we put “automatically” into quotation marks because a complete automation would require

a. the complete automation of such proofs (proof attempts) as done in the previous section, and

b. the complete automation of the generation of the requirements for the subalgorithms lc and df.

We are very optimistic that a. can soon be achieved because similar proofs are already in the reach of automated theorem proving systems like Theorema. As for b., surprisingly, relatively simple rules can cope with the requirement generation for subalgorithms and were already implemented in [8, 11]. The new rule needed in the second case of the above proof attempt is not yet implemented but we foresee no big difficulties in the implementation.

Thus, summarizing, for the first time, the automated synthesis of the author's Gröbner bases algorithm (and similar algorithms), which seemed to be far outside the reach of current synthesis methods, seems to be possible. The "lazy thinking" synthesis method which we use for this purpose is based on two key ideas:

A. the use of algorithm schemes and

B. the automated analysis of failing proof attempts and the automated generation of (lemmata and) subalgorithm requirements based on these attempts.

Of course, the algorithm schemes already capture essential ideas of the algorithm "invention". However, it would be very silly *not* to use the condensed expertise for algorithm design contained in algorithm schemes, which either can be distilled from examples of known successful algorithms or can be built up systematically by syntactical enumeration of all possible right–hand sides in recursive function definitions. It is of course a different question how the invention of an algorithm happened at a time when the pertinent algorithm scheme was *not* yet known as was the case when the author came up with the Gröbner bases algorithm based on S–polynomials in the PhD thesis 1965 (published in [3]).

Anyway, it also should be emphasized that, even if we did not aim at a complete automation of our lazy thinking approach to algorithm synthesis, we think that the approach gives a good heuristics for the systematic invention of algorithms by *human* mathematicians or, in other words, it gives a good explanation of how, maybe, invention often happens in human mathematicians. In fact, paraphrasing A. and B. didactically, one could say that A. suggests to use, "by analogy", algorithmic ideas which already have proven useful in other contexts and B. suggests to "learn from failure". Both suggestions are well known as "didactic principles". The point is that, in our lazy thinking approach to algorithm synthesis, we make these principles concrete and algorithmic.

Note also that the degree of automation achievable in the lazy thinking approach to algorithm synthesis depends crucially also on the degree of completeness of the available knowledge base on the ingredient concepts of the theory: The more complete the knowledge base is, the easier is it to find and structure the attempts of the correctness proof on which the algorithm synthesis is based. For example, of course, Newman's lemma as part of the knowledge base in our above synthesis makes the whole proof technically *much* easier and focuses our attention on the essential stumbling block of the proof. Therefore, as a fundamental and natural strategy in systematic theory exploration we postulate that, before one starts with the synthesis of an algorithm for a non-trivial problem, one should first "completely" explore the underlying theory, i.e. one should fill up the knowledge base of properties of the underlying concepts as completely as possible. Some first ideas about systematic theory exploration are given in [7] and [9].

Note: Among the various other systematic ("artificial intelligence") approaches to algorithm synthesis, the approach of [2] seems to be particularly interesting for the Gröbner bases construction problem. Given a problem specification $\mathtt{P}$, this approach starts from a (constructive) proof of the existence formula

$$\forall_F \ \exists_G \ \mathtt{P}[F, G]$$

and tries to extract an algorithm from a successful proof. For the Gröbner bases case, the inconstructive proof of the existence of Gröbner bases using the contour argument, which has first been given in [4], seems to be a promising candidate for starting the investigation along this approach. Of course, the crucial challenge is how to turn the inconstructive proof into a constructive one without already providing the key idea of the "critical pair" (or, equivalently, the "S–polynomial") of two polynomials.

Acknowledgement. Sponsored by FWF (Österreichischer Fonds zur Förderung der Wissenschaftlichen Forschung; Austrian Science Foundation), project SFB 1302 ("Theorema") of the SFB 013 (Special Research Area "Scientific Computing") and RICAM (Radon Institute for Computational and Applied Mathematics, Austrian Academy of Science, Linz). The author is also grateful to Helmut Schwichtenberg for an invitation to the Mathematical Institute of the University of Munich in December 2003 and stimulating discussions on the subject of this paper. Sincere thanks also to Markus Rosenkranz, Florina Piroi and Laura Kovács for helpful discussions, proof reading, and translating the Theorema version of the paper into LaTeX.

References

[1] T. Becker and V. Weispfenning. 1993. *Gröbner Bases: A Computational Approach to Commutative Algebra*. Springer, New York.

[2] U. Berger, W. Buchholz and H. Schwichtenberg. 2002. Refined Program Extraction from Classical Proofs. *Ann. Pure Appl. Logic*, **114**, pp. 3–25.

[3] B. Buchberger. 1970. Ein algorithmisches Kriterium für die Lösbarkeit eines algebraischen Gleichungssystems (An Algorithmical Criterion for the Solvability of Algebraic Systems of Equations). *Aequationes Mathematicae*, **4/3**, pp. 374–383. (English translation in: [12], pp. 535–545). Published version of the PhD Thesis of B. Buchberger, University of Innsbruck, Austria, 1965.

[4] B. Buchberger. 1983. *Miscellaneous Results on Gröbner Bases for Polynomial Ideals II*. Technical Report 83–1, Dept. of Computer and Information Sciences, Univ. of Delaware, Newark, Delaware.

[5] B. Buchberger. 1987. History and Basic Features of the Critical-Pair/Completion Procedure. *J. Symbolic Computation*, **3**, 1/2, pp. 3-38.

[6] B. Buchberger. 1998. *Introduction to Gröbner Bases*. In: [12], pp. 3–31.

[7] B. Buchberger. 2000. Theory Exploration with Theorema. In: T. Jebelean, V. Negru and A. Popovici (eds.), *Analele Universitatii din Timisoara*, Seria Matematica-Informatica, Vol. XXXVIII, Fasc.2, ISSN 1124-970X, Mirton Publisher Timisoara, pp. 9–32. (Proceedings of SYNASC 2000, 2nd International Workshop on Symbolic and Numeric Algorithms in Scientific Computing, October 4–6, 2000, Timisoara, Romania.)

[8] B. Buchberger. 2003. Algorithm Invention and Verification by Lazy Thinking. In: T. Jebelean, V. Negru and A. Popovici (eds.), *Analele Universitatii din Timisoara, Seria Matematica-Informatica*, Vol. XLI, special issue on Computer Science, ISSN 1124-970X, Mirton Publisher Timisoara, pp. 41-70. (Proceedings of SYNASC 2003, 5th International Workshop on Symbolic and Numeric Algorithms in Scientific Computing, October 1–4, 2003, Timisoara, Romania.)

[9] B. Buchberger. 2004. Algorithm Supported Mathematical Theory Exploration: A Personal View and Strategy. In: B. Buchberger and J. Campbell (eds.) *Proceedings of AISC 2004* (7 th International Conference on Artificial Intelligence and Symbolic Computation, September 22-24, 2004, RISC, Johannes Kepler University, Austria), Springer Lecture Notes in Artificial Intelligence, Vol. 3249, Springer, Berlin, Heidelberg, pp. 236-250.

[10] B. Buchberger, C. Dupre, T. Jebelean, F. Kriftner, K. Nakagawa, D. Vasaru and W. Windsteiger 2000. 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.

[11] B. Buchberger and A. Craciun 2003. Algorithm Synthesis by Lazy Thinking: Examples and Implementation in Theorema. *Electronic Notes in Theoretical Computer Science*, Vol. 93, 18 February, 2004, pp. 24-59, (Proceedings of the Mathematical Knowledge Management Workshop, Edinburgh, Nov. 25, F. Kamareddine (ed.)), http://www.sciencedirect.com/science/journal/15710661.

[12] B. Buchberger and F. Winkler (eds.). 1998. *Gröbner Bases and Applications* (Proceedings of the International Conference ”33 Years of Gröbner Bases”, 1998, RISC, Austria). London Math. Soc. Lecture Note Series, **251**, Cambridge University Press.

[13] L.E. Dickson. 1913. Finiteness of the Odd Perfect and Primitive Abundant Numbers With n Distinct Prime Factors. *Amer. J. Math.*, **35**, pp. 413–422.

[14] D.E. Knuth and P.B. Bendix. 1970. Simple Word Problems in Universal Algebra. In: J. Leech (ed.) *Computational Problems in Abstract Algebras*, Pergamon Press, pp. 263–297.

[15] M. Kreuzer and L. Robbiano L. 2000. *Computational Commutative Algebra*. Springer-Verlag, Berlin, Heidelberg, New-York.

[16] I. Medina-Bulo, F. Palomo-Lozano, J.A. Alonso-Jimenez and J.L. Ruiz-Reina. 2004. Verified Computer Algebra in ACL2 (Gröbner Bases Computation). In: B. Buchberger and J. Campbell (eds.) *Proceedings of AISC 2004* (7 th International Conference on Artificial Intelligence and Symbolic Computation, September 22-24, 2004, RISC, Johannes Kepler University, Austria), Springer Lecture Notes in Artificial Intelligence, Vol. 3249, Springer, Berlin, Heidelberg, pp. 171-184.

[17] M.H.A. Newman. 1942. On Theories With a Combinatorial Definition of Equivalence. *Annals of Mathematics*, **43**, pp. 233–243.

[18] J.A. Robinson. 1965. A Machine–oriented Logic Based on the Resolution Principle. *Journal of the ACM*, **12**, pp. 23–41.

[19] L. Théry, L. 2001. A Machine-Checked Implementation of Buchbergers's Algorithm. *Journal of Automated Reasoning*, **26**, pp.107-137.

[20] S. Wolfram. 1996. *The Mathematica Book*. Wolfram Media.

Bruno Buchberger
Research Institute for Symbolic Computation
Johannes Kepler University
A 4232 Schloss Hagenberg, Austria
buchberger@risc.uni-linz.ac.at