The Groebner Factorizer and Polynomial System Solving

Talk given at the Special Semester on Groebner Bases Linz 2006

Hans-Gert Gräbe
Dept. Computer Science, Univ. Leipzig, Germany
http://www.informatik.uni-leipzig.de/~graebe

February 28, 2006

1 Introduction

Let $S := k[x_1, \ldots, x_n]$ be the polynomial ring in the variables $x_1, \ldots, x_n$ over the field k and $B := \{f_1, \ldots, f_m\} \subset S$ be a finite system of polynomials. Denote by $I(B)$ the ideal generated by these polynomials. One of the major tasks of constructive commutative algebra is the derivation of information about the structure of

$$V(B) := \{a \in K^n : \forall\ f \in B \text{ such that } f(a) = 0\},$$

the set of common zeroes of the system B over an algebraically closed extension K of k.

Splitting the system into smaller ones, solving them separately, and patching all solutions together is often a good guess for a quick solution of even highly nontrivial problems. This can be done by several techniques, e.g., characteristic sets, resultants, the Groebner factorizer or some ad hoc methods. Of course, such a strategy makes sense only for problems that really will split, i.e., for reducible varieties of solutions. Surprisingly often, problems coming from “real life” fulfill this condition.

Among the methods to split polynomial systems into smaller pieces probably the Groebner factorizer method attracted the most theoretical attention, see Czapor ([4, 5]), Davenport ([6]), Melenk, Möller and Neun ([16, 17]) and Gräbe ([13, 14]). General purpose Computer Algebra Systems (CAS) are well suited for such an approach, since they make available both a (more or less) well tuned implementation of the classical Groebner algorithm and an effective multivariate polynomial factorizer.

Furthermore it turned out that the Groebner factorizer is not only a good heuristic approach for splitting, but its output is also usually a collection of almost prime components. Their description allows a much deeper understanding of the structure of the set of zeroes compared to the result of a sole Groebner basis computation.

Of course, for special purposes a general CAS as a multipurpose mathematical assistant can't offer the same power as specialized software with efficiently implemented and well adapted algorithms and data types. For polynomial system solving, such specialized software has to implement two algorithmically complex tasks, solving and splitting, and until recently none of the specialized systems (as e.g., GB, Macaulay, Singular, CoCoA, etc.) did both efficiently. Meanwhile, being very efficient computing (classical) Groebner bases, development efforts are also directed, not only for performance reasons, towards a better inclusion of factorization into such specialized systems. Needless to remark that it needs some skill to force a special system to answer questions and the user will probably first try his "home system" for an answer. Thus the polynomial systems solving facility of the different CAS should behave especially well on such polynomial systems that are hard enough not to be done by hand, but not *really hard* to require special efforts. It should invoke a convenient interface to get the solutions in a form that is (correct and) well suited for further analysis in the familiar environment of the given CAS as the personal mathematical assistant.

2 The Groebner Algorithm with Factorization

Define a *zero set with constraints*

$$V(B,C) := \{a \in K^n : \forall\ f \in B\ f(a) = 0 \text{ and } \forall\ g \in C\ g(a) \neq 0\ \},$$

This notion generalizes the notion of $V(B)$.

Note that C may be replaced for theoretical considerations by a single polynomial $c = \prod_{g \in C} g$, but the given notion has some advantage in handling constraints during computations.

The Groebner algorithm with factorization (Groebner factorizer for short) addresses the following

General Problem

Given a system $B = \{f_1, \ldots, f_m\} \subset S$ of polynomials and a set of constraints C find a collection (B_α, C_α) of polynomial systems B_α in "triangular" form (for the moment: being a Groebner basis) and constraint sets C_α such that

$$V(B,C) = \bigcup_\alpha V(B_\alpha, C_\alpha).$$

Using factorization this problem may be solved with the following algorithm

Factorized Groebner Bases FGB(B,C)

(1) During a preprocessing interreduce B and try to factor each polynomial $f \in B$. If f factors, replace B by a set of new problems, one for each factor of f. Update the side conditions and apply the preprocessing recursively. This ends up with a list of interreduced problems with non factoring base elements.

(2) For each basis in the problem list compute its list of critical pairs and put this Groebner task into the task list. Each Groebner task is a triple $T_\alpha = (B_\alpha, C_\alpha, P_\alpha)$ where B_α is a partial Groebner basis and P_α the list of remaining pairs.

(3) Choose a Groebner task from the task list, run it in the usual way but try each reduced (non zero) S-polynomial to factor before it will be added to the polynomial list. If it factors then interrupt, split the task into as many subtasks as there are (different) factors, and put them back into the task list.

(4) If the Groebner task finished, extract the minimal Groebner basis of the subproblem.

If it is not yet reduced apply tail reduction to compute the minimal reduced Groebner basis. This may cause some of the base elements to factor anew. Apply the preprocessing once more.

If the result is stable then put it in the list of results.

Otherwise put the subproblems as new Groebner tasks back into the task list.

Obviously this algorithm terminates and returns a list of pairs (B_α, C_α) with the desired properties. The realization uses the following elementary operations :

1. **Updating after factorization**

   If (B, C) is a problem and $f \in I(B)$ factors as $f = g_1^{a_1} \dots g_m^{a_m}$ then replace the problem by the problem list

   $$\textbf{NewCon}(B, C, \{g_1, \dots, g_m\}) := \{(B \cup \{g_i\}, C \cup \{g_1, \dots, g_{i-1}\}) \mid i = 1, \dots, m\}$$

2. **Inconsistency check**

   (B, C) is inconsistent, i.e. $V(B, C) = \emptyset$, if the normal form $NF(c, B) = 0$ for some $c \in C$.

3. **Subproblem removal check**

   (B_1, C_1) can be removed if there is a problem or partial result (B_2, C_2) such that $V(B_1, C_1) \subset V(B_2)$. This occurs if $NF(f, B_1) = 0$ for all $f \in B_2$. The second problem has to be replaced by $(B_2, C_1 \cap C_2)$.

Both checks use not the full power of information but only sufficient conditions. Indeed, the side condition $C_1 \cap C_2$ in (3) is weaker than the (logical) disjunction of C_1 and C_2. But the latter may not correspond to a main open set in the Zariski topology. Since $C_1 \supset C$ and $C_2 \supset C$ we obtain nevertheless $C_1 \cap C_2 \supset C$. Hence the total set of solutions is not enlarged. This is the main point to use constraint sets and not constraint polynomials.

For (2), we remark that e.g. the full inconsistency check would need a radical membership test, i.e. another (full) Groebner basis calculation. This is impossible in the given frame since all checks must be easy enough not to influence the performance of the main algorithm to heavily. If B is the *Groebner* basis of a *prime* ideal, $NF(c, B) = 0$ for some $c \in C$ is also necessary for $V(B, C) = \emptyset$. Since our connection of factorization and Groebner basis computation leads often to such bases, one should force to progress with a subproblem as deep as possible to take best advantage of the side conditions (and the removal check).

Splitting problems recursively in sets of subproblems yields a tree structure as described in [16] and conceptually used in both the REDUCE and the AXIOM implementations. Using a task list as in the approach described here has two advantages:

- It adds more flexibility in the choice (3) of the next Groebner task to be processed. Especially, splitting one of the leaves of the tree into subproblems one can queue up *all* these problems and continue with a different subproblem. Compared to the recursive approach as e.g. implemented in AXIOM this may lead to a significant speedup, although by the above theoretical remark and empirical observations a depth first strategy has to be preferred.
- One can apply the subproblem removal check not only to the current task, but to all tasks (and results) queued up. This may lead to significant savings cancelling branches in an early stage of the computation.

As explained above we should force a problem sort strategy that mimics a depth first recursion on the corresponding tree. We approximate this strategy sorting the problems by their virtual dimension. This is the dimension of the current Lt-ideal $Lt(B)$. Computing new S-polynomials the Lt-ideal grows and hence its dimension may only decrease. Thus our strategy forces to treat Groebner tasks in greatest progress first.[1].

3 The Preprocessing

We organized the preprocessing in a recursive way factoring each time a single basis element and then interreducing and updating the corresponding subproblems before the preprocessor is called

[1]This is not the whole story since we use the easy linear time dimension algorithm, that works properly only for unmixed ideals, see [11]

on them anew. This has some advantage against the complete factorization of *all* base elements in one step. To see this we restrict our considerations to ideals generated by monomials, this way discussing the influence of common factors occuring in several base elements on the preprocessing, but not the tail reduction effects derived from them.

To compare our recursive approach with a complete factorization of all base elements, note that e.g. Reisner's example, an ideal generated by 10 square-free monomials of degree 3, would split into 3^{10} subproblems with the latter approach, whereas it splits into 10 prime monomial ideals with 46 intermediate subproblems with the former one. This is true for almost all examples containing many splitting basis elements. The following general example illustrates the situation once more :

Example:

$$I_{m,n} := I(x_{2k-1}x_{2l} : 1 \le k \le m, 1 \le l \le n)$$

As easily seen, $I_{m,n} = I(x_{2k-1} : 1 \le k \le m) \cap I(x_{2l} \ : 1 \le l \le n)$ decomposes finally into two subproblems. Factoring all the mn generators at once and combining them to subsystems yields 2^{mn} different ideals (of coordinate hyperspaces). Two of them are the above minimal ones with respect to inclusion.

Using a recursive splitting argument we produce $I(x_2, x_4, \ldots, x_{2n})$ on the main branch and successively $I_0, I_1, \ldots, I_{n-1}$ in the problem list with

$$I_0 = I(x_{2m-1}) + I_{m-1,n}$$

and

$$I_k = I(x_{2m-1}, x_{2n}, \ldots, x_{2(n-k)}) + I_{m-1,n-k} \qquad \text{for } k > 0.$$

Since $I_0 \subset \ldots \subset I_{n-1}$ only one problem survives and $I_{m,n}$ splits recursively generating only mn intermediate subproblems.

4 Solving polynomial systems – A short overview

Let's first collect together some theoretical results and give a survey about the way solutions of polynomial systems may be represented. We only sketch the results below and refer the reader for more details to the papers [13, 14].

Solving systems of polynomial equations in an ultimate way means to find a decomposition of the variety of solutions into irreducible components, i.e., a representation of the radical $Rad\ I(B)$ of the defining ideal as an intersection of prime ideals, and to present them in a way that is well suited for further computations. Usually one tries first to solve this problem over the ground field k, since the corresponding transformations may be performed without introducing new algebraic quantities. Only in a second step, K (or another extension of k) is involved. Since for general univariate polynomials of higher degree, there are no closed formulae for their zeroes using radicals and moreover, even for equations of degree 3 and 4, the closed form causes great difficulties during subsequent simplifications, nowadays in most of the CAS the second step is by default not executed (in exact mode), but encapsulated in the functional symbol $\texttt{RootOf}(p(x), x)$[2], representing the sequence, set, list, etc. of solutions of the equation $p(x) = 0$ for a certain (in most cases not necessarily irreducible) polynomial $p(x)$. Hence we will not address the second step in the rest of this paper, too.

Attempting to find a full prime decomposition leads to quite difficult computations (the approach described in [9], and refined meanwhile in a series of papers, needs several Groebner basis computations over different transcendental extensions of the ground field), involving generic coordinate changes in the general case. The latter occurs, e.g., if one tries to separate the solution set of $\{x^2 - 2, y^2 - 2\}$ over $\mathbb{Q}$ into the components $\{x^2 - 2, y - x\}$ and $\{x^2 - 2, y + x\}$.

[2]We use this symbol to represent roots of univariate polynomials symbolically in a concise way, that is present in similar versions, but under different names in almost all of the CAS under consideration.

Since the system $\{x^2-2, y^2-2\}$ is already in a form convenient for computational purposes, one may wish not to ask for a completely split, but for a *triangular* solution set. It turns out that every zero dimensional system of polynomials may be decomposed over k into such pieces even without factorization.

4.1 Solving zero dimensional polynomial systems

The notion of triangular systems was introduced for zero dimensional ideals by Lazard in [15] and meanwhile became widely accepted, see the monograph [18].

A set of polynomials $\{f_1(x_1), f_2(x_1,x_2), \ldots, f_n(x_1,\ldots,x_n)\}$ is a (zero dimensional) *triangular system* (reduced triangular set in [15]) if, for $k=1,\ldots,n$, $f_k(x_1,\ldots,x_k)$ is monic (i.e., has an invertible leading coefficient) regarded as a polynomial in x_k over $k[x_1,\ldots,x_{k-1}]$, and the ideal $I=I(f_1,\ldots,f_n)$ is radical, i.e., is an intersection of prime ideals. For such a triangular system, S/I is a finite sum of algebraic field extensions of k. One can effectively compute in such extensions, as was discussed in [15]. Lazard proposed to apply the D5 algorithm to decompose a zero dimensional polynomial system into triangular ones. There is also another approach, suggested in [19].

Proposition 1 *([15, 19]) Let B be a zero dimensional polynomial system.*

1. *If $I(B)$ is prime then a lexicographic Groebner basis of B is triangular.*
2. *For an arbitrary B, there is an algorithm that computes a finite number of triangular systems $T_1,\ldots,T_m$, such that*
$$V(B)=\bigcup_i V(T_i)$$
is a decomposition (over k) of $V(B)$ into pairwise disjoint sets of points.

Note that although such a decomposition may be found not involving any (full) factorization but only gcd computations, it is usually of great benefit to try to factor the system B first into smaller pieces, since the size of the systems drastically influences the computation necessary for a triangulation.

4.2 Polynomial systems with infinitely many solutions

If $P=I(B)$ is a prime ideal of positive dimension, the corresponding variety $V(B)$ may be parameterized by the generic zero of P using reduction to dimension zero.

For a given ideal $I\subset S$, and a subset $V\subset\{1,\ldots,n\}$, the set of variables $(x_v, v\in V)$ is an *independent set* iff $I\cap k[x_v, v\in V]=(0)$. That is the variables $(x_v, v\in V)$ are algebraically independent *mod* I. Hence if $(x_v, v\in V)$ is a maximal (with respect to inclusion) independent set for the ideal $I(B)$, these variables can be regarded as parameters whereas the remaining variables depend algebraically on them.

This corresponds to a change of the base ring $S\longrightarrow\widetilde{S}:=k(x_v, v\in V)[x_v, v\notin V]$, where $\widetilde{S}$ is the ring of polynomials in $x_v, v\notin V$ with coefficients in the function field $k(x_v, v\in V)$. This base ring extension corresponds to a localization at the multiplicative set of nonzero polynomials in the variables $(x_v, v\in V)$. The extension ideal $I\cdot\widetilde{S}$ is a zero dimensional ideal in $\widetilde{S}$ and the algorithms mentioned so far can be applied. For a prime ideal, we get $I=I\cdot\widetilde{S}\cap S$, i.e., the generic zero describes a prime ideal I completely, presenting the quotient ring $Q(S/I)=Q(\widetilde{S}/\widetilde{I})$ as a finite algebraic extension of $k(x_v : v\in V)$. For arbitrary ideals $I(B)$, the localization may cut off some of the components of $V(B)$ that must be parameterized separately. This can be done in an effective way, see [14].

As for zero dimensional ideals, such a presentation is not restricted to prime ideals. In more generality, let $T=\{f_1,f_2,\ldots,f_m\}$ be a set of polynomials in S. We say that T forms a *triangular* *system with respect to the maximal independent set* $(x_v, v \in V)$ of $I = I(T)$ if the extension $\widetilde{T}$ of T to $\widetilde{S} := k(x_v, v \in V)[x_v, v \notin V]$ forms a triangular system for the (zero dimensional) extension ideal $\widetilde{I} := I \cdot \widetilde{S}$.

Such a triangular system may be regarded as a parameterization of $V(I\widetilde{S} \cap S)$, i.e., of the components of $V(T)$ missing the multiplicative set $k[x_v, v \in V] \setminus \{0\}$. Hence a general polynomial system solver should (at least) decompose a given system of polynomials B into triangular systems, defining for each of them the independent variables, extracting the part of $V(B)$ parameterized this way, and describing recursively the remaining part of $V(B)$.

Altogether we can formulate the following

> **Polynomial System Solving Problem**:
>
> *Given a finite set $B \subset S$ of polynomials, find a collection (T_k, V_k) of triangular systems T_k with respect to $V_k \subset \{x_1, \dots, x_n\}$, such that*
>
> - $I_k := I(T_k) \cdot k(x_v, v \in V_k)[x_v, v \notin V_k] \cap S$ *is a pure dimensional radical ideal with V_k as a maximal strongly independent set,*
> - $V(B) = \bigcup V(I_k)$*, and*
> - *this decomposition is minimal.*

Note that due to denominators occuring in $k(x_v, v \in V_k)$, the system T_k yields a parameterization of only "almost all" points of $V(I_k)$. By geometric reasons one cannot expect more, since even simple examples of algebraic varieties show that a rational parameterization may miss some points on the variety, see e.g., [3, chapter 3] for details. Since $V(I_k)$ is the algebraic closure of the parameterized part, for each point $X \in V(I_k)$, there is at least a curve of parameterized points approaching X.

Let's conclude this section with a little example:

Consider the system $B = \{x^3 - y^2,\, x\,y - z\}$. A lexicographic Groebner basis computation yields $G = \{y^5 - z^3,\, x\,z^2 - y^4,\, x^2\,z - y^3,\, x\,y - z, x^3 - y^2\}$. Hence (z) is a maximal independent set by [11] and $G' = \{y^5 - z^3,\, x\,(z^2) - y^4\}$ a triangular system with respect to (z) (and a minimal Groebner basis of $I(G) \cdot \widetilde{S}$). Rewinding this tower of extensions over K, we get as parameterization of $V(B)$ the five branches

$$\{(z^{\frac{2}{5}}, z^{\frac{3}{5}}, z) \;:\; z \in K\}$$

corresponding to the different choices of the branch of $\sqrt[5]{z}$. Note that these branches may be parameterized by a unique formula

$$V(B) = \{(t^2, t^3, t^5) \;:\; t \in K\},$$

i.e., the variety is rational. It's a deep geometric question to decide whether an irreducible variety admits a rational parameterization and in the case it does to find one, see e.g., [20]. Since none of the CAS under consideration invokes such algorithms with their solver, we will not address this question here. Note that moreover in most applications, the structure of the prime components is very easy. Often they may be either *regularly (polynomially) parameterized* (reg. par.)

$$\{x_k - p_k(x_1, \dots, x_d) \mid k = d+1, \dots, n\},\; p_k \in k[x_1, \dots, x_d],$$

rationally parameterized (rat. par.)

$$\{x_k - p_k(x_1, \dots, x_d) \mid k = d+1, \dots, n\},\; p_k \in k(x_1, \dots, x_d)$$

or are zero-dimensional *primes in general position* (g.p.)

$$\{x_k - p_k(x_n) \mid k = 1, \dots, n-1\} \cup \{p_n(x_n)\},\; p_k \in k[x_n].$$

5 A first example

To get a first impression about the power of the polynomial system solver implemented in different CAS, let's `Solve` a very simple one dimensional system like the Arnborg example $A4$, cf. [6].

$$\begin{aligned}
&vars := \{w, x, y, z\}\\
&polys := \{wxy + wxz + wyz + xyz, wx + wz + xy + yz,\\
&\qquad\qquad w + x + y + z, wxyz - 1\}
\end{aligned}$$

The Groebner factorizer yields easily the following (already prime) decomposition into two one dimensional components:

$$\big\{\{w + y, x + z, yz - 1\}, \{w + y, x + z, yz + 1\}\big\}$$

Macsyma returns as answer the list

$$\begin{aligned}
\{\{&w = -\%\mathrm{r}5, x = \tfrac{1}{\%\mathrm{r}5}, y = \%\mathrm{r}5, z = -\tfrac{1}{\%\mathrm{r}5}\},\\
&\{w = -\%\mathrm{r}6, x = -\tfrac{1}{\%\mathrm{r}6}, y = \%\mathrm{r}6, z = \tfrac{1}{\%\mathrm{r}6}\},\\
&\{w = -i, x = i, y = i, z = -i\}, \{w = i, x = -i, y = -i, z = i\},\\
&\{w = -1, x = 1, y = 1, z = -1\}, \{w = 1, x = -1, y = -1, z = 1\},\\
&\{w = -i, x = -i, y = i, z = i\}, \{w = i, x = i, y = -i, z = -i\},\\
&\{w = 1, x = 1, y = -1, z = -1\}, \{w = -1, x = -1, y = 1, z = 1\}\}
\end{aligned}$$

that contains the two expected one dimensional components together with a list of embedded points.
Mathematica returns the following result (slightly improved in 5.2)

$$\begin{aligned}
\{\{&w \to -\tfrac{1}{z}, y \to \tfrac{1}{z}, x \to -z\}, \{w \to \tfrac{1}{z}, y \to -\tfrac{1}{z}, x \to -z\},\\
&\{x \to -\tfrac{1}{y}, z \to \tfrac{1}{y}, w \to -y\}, \{x \to \tfrac{1}{y}, z \to -\tfrac{1}{y}, w \to -y\}\}
\end{aligned}$$

together with a warning

```
Solve::svars: Warning: Equations may not give solutions
     for all "solve" variables.
```

We obtain four instead of two solution sets, possibly according to the fact whether $z \neq 0$ or $y \neq 0$ in $y\,z - 1$, respectively $y\,z + 1$. Such a distinction is unnecessary, since $y\,z = 1$ implies $y, z \neq 0$.
MuPAD, Maple and Reduce return the solution in the expected form

$$\{\{x = -z, y = z^{-1}, w = -z^{-1}, z = z\}, \{x = -z, y = -z^{-1}, w = z^{-1}, z = z\}\}$$

Calling `solve(polys,vars)` with Axiom 2.0 ends up with the message

```
Error detected within library code:
   system does not have a finite number of solutions
```

whereas a direct call of the Groebner factorizer

    groebnerFactorize polys

returns (almost) the expected answer

$$[[1], [z + x, y + w, wx + 1], [z + x, y + w, wx - 1]]$$

6 Solving zero dimensional systems

As explained above, zeroes of univariate polynomials of degree five don't admit closed form representations in radicals in general and even for polynomials of degree 3 and 4, these expressions are usually so difficult that they cause great trouble during simplification of derived expressions. Moreover the evident degree reduction rule for a single `RootOf` symbol leads to a *normal representation* of rational expressions containing this symbol. Hence it has become a certain standard to represent algebraic numbers even of small degree occuring in the solution set of a polynomial system through a `RootOf` construct.

Functional symbols such as `RootOf` are ubiquitous objects in symbolic computations to construct new symbolic expressions from old ones. The target of the simplification system of a CAS is to make the inner world of the symbolic structure of the arguments cooperate with the outer world of the environment of the functional symbol, e.g., expanding arguments over function symbols, collecting expressions together, or applying rules to combinations of symbols.

In this context, the `RootOf` symbol plays a special role, since, different to most other functional symbols, it stands for a *compound data structure* and should expand in a proper way in different situations, i.e., expand partly or completely into the promised data structure under substitution of values for parameters, during approximate evaluation, etc.

The first three examples show how different CAS manage this difficulty.

EXAMPLE 1:

$$\begin{aligned}&vars := \{x, y, z\}\\&polys := \{x^2 + y + z - 3, x + y^2 + z - 3, x + y + z^2 - 3\}\end{aligned}$$

This system has 8 different complex solutions. The set of solutions is stable under cyclic permutations of the coordinates as are the equations. Note that the univariate polynomial of least degree in z (and, by symmetry, also in x and in y) belonging to $I(polys)$ has only degree 6.

Systems that involve factorization have no trouble decomposing this system. The CAS under consideration answer in the following way:

Maple:

$$\begin{aligned}sol := \{&\{y = -3, x = -3, z = -3\}, \{z = 1, x = 1, y = 1\},\\&\{z = \mathtt{RootOf}(\_Z^2 - 2), y = \mathtt{RootOf}(\_Z^2 - 2),\\&\qquad x = -\mathtt{RootOf}(\_Z^2 - 2) + 1\},\\&\{z = \mathtt{RootOf}(\_Z^2 - 2), y = -\mathtt{RootOf}(\_Z^2 - 2) + 1,\\&\qquad x = \mathtt{RootOf}(\_Z^2 - 2)\},\\&\{x = -\mathtt{RootOf}(-2\,\_Z - 1 + \_Z^2) + 1, z = \mathtt{RootOf}(-2\,\_Z - 1 + \_Z^2),\\&\qquad y = -\mathtt{RootOf}(-2\,\_Z - 1 + \_Z^2) + 1\}\}\end{aligned}$$

Axiom:

$$\begin{aligned}sol := [&[x = 1, y = 1, z = 1],\ [x = -3, y = -3, z = -3],\\&[x = -z + 1, y = -z + 1, 2z^2 - 2z - 1 = 0],\\&[x = z, y = -z + 1, z^2 - 2 = 0],\ [x = -z + 1, y = z, z^2 - 2 = 0]]\end{aligned}$$

Macsyma and Reduce:

$$\begin{aligned}sol := \{&\{x = \sqrt{2} + 1, y = -\sqrt{2}, z = -\sqrt{2}\}, \{x = \sqrt{2}, y = \sqrt{2}, z = -\sqrt{2} + 1\},\\&\{x = \sqrt{2}, y = -\sqrt{2} + 1, z = \sqrt{2}\}, \{x = -\sqrt{2} + 1, y = \sqrt{2}, z = \sqrt{2}\},\\&\{x = -\sqrt{2}, y = \sqrt{2} + 1, z = -\sqrt{2}\}, \{x = -\sqrt{2}, y = -\sqrt{2}, z = \sqrt{2} + 1\},\\&\{x = 1, y = 1, z = 1\}, \{x = -3, y = -3, z = -3\}\}\end{aligned}$$

Mathematica and MuPAD have some trouble to extract the solution set in such clarity but leave nested roots.

$$\Big\{ [x = 1, y = 1, z = 1]\,,$$
$$[x = -3, y = -3, z = -3]\,,$$
$$\left[x = \sqrt{2}, y = \sqrt{2}, z = 1 - \sqrt{2}\right],$$
$$\left[x = -\sqrt{2}, y = -\sqrt{2}, z = \sqrt{2} + 1\right],$$
$$\left[x = \frac{\sqrt{4\sqrt{2}+9}}{2} + 1/2, y = 1/2 - \frac{\sqrt{4\sqrt{2}+9}}{2}, z = -\sqrt{2}\right],$$
$$\left[x = \frac{\sqrt{9-4\sqrt{2}}}{2} + 1/2, y = 1/2 - \frac{\sqrt{9-4\sqrt{2}}}{2}, z = \sqrt{2}\right],$$
$$\left[x = 1/2 - \frac{\sqrt{4\sqrt{2}+9}}{2}, y = \frac{\sqrt{4\sqrt{2}+9}}{2} + 1/2, z = -\sqrt{2}\right],$$
$$\left[x = 1/2 - \frac{\sqrt{9-4\sqrt{2}}}{2}, y = \frac{\sqrt{9-4\sqrt{2}}}{2} + 1/2, z = \sqrt{2}\right] \Big\}$$

We conclude, that (for multivariate systems) Maple and Axiom use the `RootOf` notation even for algebraic numbers of degree 2 whereas Reduce, Macsyma and Mathematica promise to handle such square root symbols as "normal" numbers. But even simple square roots of integers may already cause trouble as in the Mathematica output of the solution above. Although Reduce has no problem handling the square roots here, in general it often runs into trouble with the default setting of the switch `rationalize` to off. Correct (in the given context) simplifications of square roots of complicated symbolic expressions cause problems to all the CAS under consideration, see e.g., example 10 below.

Note that although Maple offers quite powerful algorithms to deal with the `RootOf` symbol, the return data type design of `RootOf`$(P(x), x)$ is not satisfactory. Due to the above syntax, it promises to return *one* of the roots of $P(x)$ instead all of them. Hence different calls to `RootOf` should choose such a root independently. This is required for different elements of *sol*, but not for different calls of `RootOf` inside the same element of *sol*. The procedure `allvalues` tries to expand the different roots properly either in radicals (up to degree 4) or to approximate them numerically. It may be called with a second parameter d to indicate that the same `RootOf` symbol in different places has to be expanded identically[3]. Issuing

```
map(allvalues,sol,d);
```

we obtain the same answer as Macsyma and Reduce for this example. Note that this approach runs into trouble with nested `RootOf` expressions in example 4 below.

The concisest solution for these problems is offered by Mathematica: It's `Roots` command returns a (promise of a) logical conjunction of zeroes that is handled properly by subsequent substitutions, approximate evaluations, etc. The `ToRules` operator transforms such a logical expression into a *Sequence*, i.e., (a promise of) comma separated expressions that automatically expand in the argument list of functional symbols with variable argument list length (`Set`, `List`, etc.) whenever possible. For example, the parametric equation $f = x^5 - 5x^3 + 4x + a$ yields with

```
Solve[f==0,x]
```

[3] `d` was changed to `dependent` in Maple version V.4.

the answer

$$\{\texttt{ToRules}[\texttt{Roots}[4\,x - 5\,x^3 + x^5 == -a, x]]\}$$

that expands under the substitution `%/.a` $\rightarrow$ `0` automatically into

$$\{\{x \rightarrow 2\}, \{x \rightarrow 1\}, \{x \rightarrow 0\}, \{x \rightarrow -1\}, \{x \rightarrow -2\}\}$$

Reduce uses the `RootOf` symbol in a sense like Maple, but binds its value for a single solution to a variable, hence resolving the ambiguity discussed above in a proper way. Expansion into a list of individual solutions is done in two steps. First, the `RootOf` symbol is changed automatically into the `one_of` symbol with the desired list as argument. Second, the user may call the command `expand_cases` to expand this functional expression properly inside a list of solutions. For the polynomial f, we get successively

solve(f,x);

$$\{x = \texttt{root\_of}(a + \_x^5 - 5 \cdot \_x^3 + 4 \cdot \_x, \_x)\}$$

sub(a=0,ws);

$$\{x = \texttt{one\_of}(\{2,\ 1,\ 0,\ -1,\ -2\})\}$$

expand_cases(ws);

$$\{x = 2,\ x = 1,\ x = 0,\ x = -1,\ x = -2\}.$$

Axiom follows another strategy. It doesn't introduce `RootOf` symbols in the output of the `solve` command, but returns a list of (simplified) systems of equations instead, that may be further simplified with the usual system command. To extract, e.g., (real) approximate solutions from the above symbolic answer one may call

realsol:=concat([solve(u,0.001) for u in sol])

and then check the answer by

[[subst(u,v) for u in polys] for v in realsol].

A great difficulty for the CAS that use a `RootOf` notion remains the proper simplification of expressions involving such symbols. A good benchmark for these capabilities is the resubstitution test, i.e., the test whether the CAS may prove that the produced solutions satisfy the initial equations. In our experiments, only Maple was able to do these simplifications with satisfactory success, possibly after a subsequent call to the algebraic simplifier `evala`, that is not invoked with `simplify`.

Example 2:

$vars := \{x, y\}$
$polys := \{x^4 + y + 1, y^4 + y + 1\}$

This example has 16 different complex solutions, none of them being real. It demonstrates both the difference between an early use of `RootOf` expressions as in Axiom, Maple and Reduce, full radical expansion of solutions of equations up to degree 4 as in Mathematica, and the difference between triangular systems and a complete decomposition into isolated prime components[4].

To begin with, note that for $x > y$, the given system of equations is already in triangular form. Moreover, it is easily seen that $x^4 - y^4$ belongs to $I(polys)$, i.e., the solution set may be decomposed into

$$V(polys) = V(x+y, y^4+y+1) \cup V(x-y, y^4+y+1) \cup V(x^2+y^2, y^4+y+1).$$

Reduce ends up with the original triangular systems

$$sol := \{\{x = \texttt{RootOf}(x^4+x+1, x), y = \texttt{RootOf}(x+y^4+1, y)\}\}$$

whereas Maple and Axiom find the decomposition (probably by some ad hoc method).

Mathematica computes 16 explicit solutions *sol*, very complicated formulae, that it isn't able to handle further symbolically. For example, the test

```
polys/.sol//Simplify
```

fails to return 0. Checking the numerical approximations `N[sol]` with

```
polys/.N[sol]
```

we detect 8 of the 16 approximations to be wrong.[5]

Macsyma resolves all that "trouble" in a different way: If results become too complicated it switches automatically to numerical solutions. For this purpose there are several root finding devices as e.g., for real and complex roots, for counting roots inside a real interval, for approximation of roots by different methods, etc. No `RootOf` philosophy is supported. Of course, such an approach doesn't work for equations with parametric coefficients as e.g., for $f = x^5 - 5x^3 + 4x + a$. For those systems, Macsyma follows the same philosophy as Axiom, i.e., returns simplified equations that may be subsequently handled with the Lisp-like system language.

Let's digress for a moment to the *numerical solving capabilities for zero dimensional systems* offered by the other CAS in the case the user isn't satisfied with the symbolic answer (as it probably will be the case in our situation).

As we already mentioned, **Mathematica** is best suited for such approximate solutions, since the numerical evaluator may be involved in a unified way in almost all situations. Besides the easy (but in this case wrong) approximate evaluation of the symbolic results themselves, one can also solve the system approximately with

```
NSolve[polys,vars]
```

to obtain 16 correct solutions.

For **Axiom** we may proceed as above. Starting from the symbolic solution

```
sol:=solve(polys,vars)
```

the complex system's solver

```
csol:=concat([complexSolve(u,0.00001) for u in sol])
```

[4]MuPAD returns the strange answer

$$\{[y = \texttt{RootOf}(-y^2 + \texttt{RootOf}(x+y^2+1, y)), x = \texttt{RootOf}(x+x^4+1)]\}.$$

[5]Note that this default behaviour may be turned off by

```
SetOptions[Roots,Cubics-¿False,Quartics-¿False]
```

Then the resulting expression is evaluated numerically in a proper way.

may be involved. It returns 16 complex solutions that may be checked to be correct in the same manner as above

[[subst(u,v) for u in polys] for v in csol]

Reduce invokes approximate algorithms, switching with `on rounded` to floating point arithmetic. Calling `solve(polys,vars)` directly returns the 16 complex solutions. The situation becomes more difficult starting from the symbolic solution `sol`. Switching to floating point arithmetic, the `root_of` expressions are changed into `one_of` expressions

$$\begin{aligned}\{\{x = \texttt{one\_of}(\{0.934099289461 * i + 0.727136084491, \ldots\}),\\ y = \texttt{one\_of}(\{(-(x+1))^{0.25} * i, \ldots\})\}\},\end{aligned}$$

that may be expanded via `Expand_Cases` into 16 complex solutions

$$\{\{x = 0.934099289461 * i + 0.727136084491, y = (-(x+1))^{0.25} * i\}, \ldots\}$$

In each of them y is expressed not as a complex number but as a formula, and even resubstitution doesn't simplify these expressions to complex numbers since Reduce can't compute powers of complex numbers with real exponents.

Maple's `fsolve` returns only a single numerical approximate solution instead of all of them. Expanding the `RootOf` expressions with `allvalues` fails in reasonable time, since it tries to express them in radicals. There is no way to avoid this.

MuPAD has not yet facilities to extract numerical solutions from the above system.

Different to the single polynomial case, **Macsyma** also doesn't offer a special routine to find all solutions of a polynomial *system* numerically beyond the mixed symbolic-numerical approach supplied with `solve` as described above.

Example 3: The Katsura example K4, [2, p. 91]

$$\begin{aligned}&vars := \{u_0, u_1, u_2, u_3\}\\ &polys := \{u_0 + 2u_1 + 2u_2 + 2u_3 - 1, u_0^2 + 2u_1^2 + 2u_2^2 + 2u_3^2 - u_0,\\ &\qquad 2u_0u_1 + 2u_1u_2 + 2u_2u_3 - u_1, 2u_0u_2 + u_1^2 + 2u_1u_3 - u_2\}\end{aligned}$$

Its zero set decomposes into two points with rational coordinates and another one with coordinates depending on an algebraic number of degree 6. Reduce, Maple and Axiom return almost immediately the correct answer

$$\begin{aligned}&[[u_0 = 1,\ u_1 = 0,\ u_2 = 0,\ u_3 = 0],\ [u_0 = \tfrac{1}{3},\ u_1 = 0,\ u_2 = 0,\ u_3 = \tfrac{1}{3}],\\ &\quad [u_0 = \frac{-381533328\ u_3^5 + 97717752\ u_3^4 + 12529296\ u_3^3 - 5057432\ u_3^2 + 7598\ u_3 + 147793}{168945},\\ &\quad u_1 = \frac{-5452920\ u_3^5 + 1977048\ u_3^4 + 589356\ u_3^3 - 177864\ u_3^2 - 17866\ u_3 + 4768}{11263},\\ &\quad u_2 = \frac{272560464\ u_3^5 - 78514596\ u_3^4 - 15104988\ u_3^3 + 5196676\ u_3^2 + 95246\ u_3 - 60944}{168945},\\ &\quad 42768\ u_3^6 - 16848\ u_3^5 - 432\ u_3^4 + 904\ u_3^3 - 72\ u_3^2 - 12\ u_3 + 1 = 0]],\end{aligned}$$

that expands numerically into the 2 rational, 4 real and 2 complex conjugate solutions. This can be proved with Axiom, Maple and Reduce as indicated during the discussion of example 2.

Mathematica returns a complicated formula with five (!) nested `Roots` expressions to determine the four indeterminates, that nevertheless correctly evaluates numerically to the 8 solutions of the system under consideration.

Macsyma finds the two rational and one further real solution.

More detailed timings are collected in the following table.

	symbSolve		numSolve1		numSolve2	
	time	structure	time	structure	time	structure
Axiom	10.8	(2x1 1x6)	134.6	8 sol.	176.2	8 sol.
Macsyma	10.2	3 sol.	—		—	
Maple	1.6	(2x1 1x6)	—		0.3	8 sol.
Mathem.	0.4	5 nested `Roots`	0.5	6 sol.	0.1	8 sol.
Reduce	0.6	(2x1 1x6)	2.6	8 sol.	0.5	8 sol.

Table 1 a: Solving example 3

The first two columns (symbSolve) are devoted to the symbolic solver's behaviour. The structure column collects information about the degrees of the different branches. The latter columns describe the numeric solver's output (numSolve1) and the numeric approximation of the symbolic results (numSolve2) as far as they may be computed. All timings are given in seconds of CPU-time as reported by the corresponding CAS.

Let's conclude this section with some more difficult examples, collecting the results in Tables 1 b – 1 e.

EXAMPLE 4:

$$\begin{aligned}&vars := \{x, y, z\}\\&polys := \{x^3 + y + z - 3, x + y^3 + z - 3, x + y + z^3 - 3\}\end{aligned}$$

The result contains several branches of degree 6, some of them are quadratic extensions of cubic ones. This explains the different branching of Reduce compared to Maple and Axiom.

	symbSolve		numSolve1		numSolve2	
	time	structure	time	structure	time	structure
Axiom	61.9	(1x1 1x2 4x6)	383	27 sol.	251	27 sol.
Macsyma	39.9	25 sol.	—		—	
Maple	2.8	(1x1 1x2 4x6)	—		1.8	39 sol.
Mathem.	160.7	11 branches	0.54	18 sol.	0.4	21 sol.
Reduce	1.7	(3x1 2x3 3x6)	14.4	27 sol.	1.5	27 sol.

Table 1 b: Solving example 4

Note that both Maple and Mathematica expand their symbolic results into a wrong number of solutions. Maple expands one of the branches of degree 6 improperly: A `RootOf` symbol of degree 2 nested with another `RootOf` symbol of degree 3 is expanded into 18 instead of 6 solutions, probably due to a wrong application of `allvalues`. Surprisingly enough, resubstitution proves only 6 of these solutions to be wrong.

EXAMPLE 5: The Arnborg example A5, [6], also known as "cyclic 5"

$$\begin{aligned}&vars := \{v, w, x, y, z\}\\&polys := \{v + w + x + y + z,\ v\,w + v\,z + w\,x + x\,y + y\,z,\\&\qquad v\,w\,x + v\,w\,z + v\,y\,z + w\,x\,y + x\,y\,z,\\&\qquad v\,w\,x\,y + v\,w\,x\,z + v\,w\,y\,z + v\,x\,y\,z + w\,x\,y\,z,\ v\,w\,x\,y\,z - 1\}\end{aligned}$$

This is already a quite hard example. The ideal is radical and of degree 70, i.e., has 70 different complex solutions.

	symbSolve		numSolve1		numSolve2	
	time	structure	time	structure	time	structure
Axiom	34 677	(5x2 15x4)	49 276	70 sol.	202	70 sol.
Maple	740	(5x2 10x4 2x16)	—		zero divide error	
Reduce	13.5	(10x1 12x4 1x12)	82.4	70 sol.	`RootOf` not properly expanded	

Table 1 c: Solving example 5

Macsyma and Mathematica were unable to crack this example. MuPAD returned a strange result, consisting of a single branch containing an equation of degree 15 for z and an equation of degree 2 depending on z for y. Altogether this may be expanded into at most 30 solutions. For numSolve2, Reduce couldn't resolve the complicated `RootOf` expression properly. Maple couldn't expand the symbolic branches, too, but crashed with a zero divide error.

Example 6:

$$vars := \{x, y, z\}$$
$$polys := \{x^3 + y^2 + z - 3, x + y^3 + z^2 - 3, x^2 + y + z^3 - 3\}$$

This is a slightly more complicated example than example 1 or example 4 from a series of systems that are symmetric under cyclic permutations.

	symbSolve		numSolve1		numSolve2	
	time	structure	time	structure	time	structure
Axiom	47.7	(1x1 1x2 1x24)	> 25 000		> 25 000	
Macsyma	36.7	25 sol.	—		—	
Maple	11.3	(1x1 1x2 1x24)	—		5.6	27 sol.
Mathem.	4396	(3x1 1x24)	593	3 sol.	0.6	27 sol.
Reduce	21.1	(3x1 1x24)	60.2	27 sol.	13.5	27 sol.

Table 1 d: Solving example 6

Example 7: The Katsura example K5, [2, p.91]

$$vars := \{u_0, u_1, u_2, u_3, u_4\}$$
$$\begin{aligned} polys := \{&u_0 + 2u_1 + 2u_2 + 2u_3 + 2u_4 - 1, \\ &u_0^2 + 2u_1^2 + 2u_2^2 + 2u_3^2 + 2u_4^2 - u_0, \\ &2u_0u_1 + 2u_1u_2 + 2u_2u_3 + 2u_3u_4 - u_1, \\ &2u_0u_2 + u_1^2 + 2u_1u_3 + 2u_2u_4 - u_2, \\ &2u_0u_3 + 2u_1u_2 + 2u_1u_4 - u_3\}\end{aligned}$$

	symbSolve		numSolve1		numSolve2	
	time	structure	time	structure	time	structure
Axiom	80.0	(2x1 1x2 1x12)	> 25 000		> 25 000	
Maple	127.5	(2x1 1x2 1x12)	—		0.9	16 sol.
Reduce	27.2	(4x1 1x12)	34.1	16 sol.	2.4	16 sol.

Table 1 e: Solving example 7

Macsyma and Mathematica were unable to crack this example.

7 Zero dimensional systems with parameters

As explained in section 4, zero dimensional polynomial systems with parameters play an important intermediate role solving polynomial systems with infinitely many solutions. Let's therefore study the behaviour of the different CAS on this topic separately.

Considering, e.g., the parametric (linear) system in the variables $\{x, y\}$

$$polys := \{a\,x + y - 1,\, x + a\,y - 1\}$$

one may pose the following two different problems:

(1) Find a description of the solution set for each possible value of the parameter a.

(2) Find a description of the "generic" solution set valid for "almost all" values of a.

Both problems may be solved with Groebner bases with respect to special term orders. For the first problem, one has to decompose the variety of solutions in (x, y, a) into components and for each of them to express (x, y) through the parameter a whenever a is not fixed (and possibly to understand the fiber structure of this component). This may be done with respect to a term order where a is lexicographically less than both x and y. The Groebner factorizer yields the simple solution

$$\{\{x = y = \tfrac{1}{a+1}\}\}$$

for $a \neq 1$ (i.e., the empty set for $a = -1$) and the one parameter solution

$$\{\{x = 1 - y, y = y\}\}$$

for $a = 1$. It should be possible to find this decomposition with

```
solve(polys,{x,y,a})
```

For the second problem, we solve the system over the coefficient field $k = \mathbf{Q}(a)$ of rational functions in the parameter a. This should be possible to realize with

```
solve(polys,{x,y})
```

We tested the different CAS on the following examples of zero dimensional (non-linear) systems for a solution in the sense of (2):

EXAMPLE 8: A parametric version of A4, with parameter z.

$$\begin{aligned}
&vars := \{w, x, y\}\\
&polys := \{wxy + wxz + wyz + xyz, wx + wz + xy + yz,\\
&\qquad\qquad w + x + y + z, wxyz - 1\}
\end{aligned}$$

EXAMPLE 9: Raksanyi's example, [2, example 6], with parameters a, u, v, w.

$$\begin{aligned}
&vars := \{x, y, z, t\}\\
&polys := \{t - (a - v), x + y + z + t - (u + w + a),\\
&\qquad\qquad xz + xt + yz + zt - (ua + uw + wa), xzt - (uwa)\}
\end{aligned}$$

This system has three rationally parameterized solutions.

EXAMPLE 10: The ROMIN 3R-robot [10] with parameters a, b, c, l_1, l_2

$$\begin{aligned}
&vars := \{s_1, s_2, s_3, c_1, c_2, c_3, d\}\\
&polys := \{a + ds_1, -b + c_1 d, c_2 l_2 + c_3 l_3 - d, -c + l_2 s_2 + l_3 s_3,\\
&\qquad\qquad c_1^2 + s_1^2 - 1, c_2^2 + s_2^2 - 1, c_3^2 + s_3^2 - 1\}
\end{aligned}$$

This system has 4 solutions, two in each branch of $d = \pm\sqrt{a^2 + b^2}$.

Here are the results of our computations for the examples 8 – 10 :

	Example 8	Example 9	Example 10	
	time	time	time	structure
Axiom	0.7	7.9	49.3	1 sol. of degree 4
Macsyma	0.5	0.5	1183	4 explicit sol.
Maple	0.4	0.2	1.5	empty[6]
Mathematica	0.13	0.25	11.5	4 explicit sol.
Reduce	0.6	0.16	369	1 sol. of degree 4

Table 2: Solving zero dimensional systems with parameters

Let's add some remarks about the output. For examples 8 and 9, all systems produced the expected rationally parameterized branches, only Mathematica's output for example 8 contained repetitions. For example 10, Axiom returned the shortest form with two quadratic equations (for c_3 and $d^2 - a^2 - b^2 = 0$) not resolved since it doesn't expand quadratic `RootOf` symbols. Macsyma produced 4 solutions with many complicated `sqrt` symbols that it wasn't able to handle during resubstitution. Similarly for Mathematica, but it could check the result `sol` to be correct with

polys/.sol//Simplify//Together

Reduce returned a single solution in terms of rational expressions in a, b, c, l_3 and a `RootOf`-expression for c_3 instead of d (although suggested in *vars* to consider d as the lowest variable) that it could not handle in a subsequent resubstitution step.

8 Systems with infinitely many solutions

Let's now analyze the behaviour of the different CAS under consideration to solve polynomial systems with solution sets of positive dimension.

The first example was contributed by one of our students who tried to study the extrema of $f(x, y) = x^3 y^2 (6 - x - y)$. It may easily be solved by hand, but already causes trouble trying to be solved automatically:

[6]But note that Maple V.4 computed within 1.68 s on a Sun UltraSparc correctly a single solution of degree four similar to that of Axiom and handled it successfully during resubstitution.

Example E1:

$vars := \{x, y\}$
$polys := \{x^2y^2(-4x - 3y + 18),\ x^3y(-2x - 3y + 12)\}$

Another quite impressive example was posted by E. Krider on June 1, 1996 in the news group `sci.math.symbolic`. It behaves like many examples arising from applications that, in contrast to their heavy input size, become tame after inter-reduction and splitting, since the individual components tend to be prime and may be presented in a simple (but not too simple) form.

Example Kri: Krider's example:

$P_0 := -6cg^2upv^5 - 2(2cd + a + bd + b + cd^2 + wbf - w + 2wcdf + 2wcf + w^2bg + 2w^2cg + 2w^2cdg + w^2cf^2 + 2w^3cfg + cg^2w^4)upv - 2(bg + 2cg + 2cdg + cf^2 + 6wcfg + 6cg^2w^2)upv^3 - 2cu^3pv;$

$P_1 := -6cg(2f + 5gw)upv^5 - 2(2cd + a + bd + b + cd^2 + wbf - w + 2wcdf + 2wcf + w^2bg + 2w^2cg + 2w^2cdg + w^2cf^2 + 2w^3cfg + cg^2w^4)wupv - 2(bf - 1 + 2cdf + 2cf + 3wbg + 6wcg + 6wcdg + 3wcf^2 + 12w^2cfg + 10cg^2w^3)upv^3 - 2wcu^3pv;$

$P_2 := -6(bg + 2cg + 2cdg + cf^2 + 10wcfg + 15cg^2w^2)upv^5 - 2(2cd + a + bd + b + cd^2 + wbf - w + 2wcdf + 2wcf + w^2bg + 2w^2cg + 2w^2cdg + w^2cf^2 + 2w^3cfg + cg^2w^4)w^2upv - 2(a + b - 3w + 6wcf + 15cg^2w^4 + 6w^2cf^2 + 12w^2cg + 6w^2bg + 3wbf + 12w^2cdg + 20w^3cfg + 6wcdf + bd + cd^2 + 2cd)upv^3 - 30cg^2upv^7 - 2w^2cu^3pv - 2cu^3pv^3;$

$M_0 := -6cg^2upv^5 - 2(a + bd + cd^2 - w + wbf + 2wcdf + w^2bg + 2w^2cdg + w^2cf^2 + 2w^3cfg + cg^2w^4)upv - 2(bg + 2cdg + cf^2 + 6wcfg + 6cg^2w^2)upv^3 - 2cu^3pv;$

$M_1 := -6g(bg + 3cdg + 3cf^2 + 15wcfg + 15cg^2w^2)upv^5 - 2(a + bd + cd^2 - w + wbf + 2wcdf + w^2bg + 2w^2cdg + w^2cf^2 + 2w^3cfg + cg^2w^4)(d + fw + gw^2)upv - 2(-f - 3gw + 3cdf^2 + 3wcf^3 + 18w^2cdg^2 + 30w^3cfg^2 + 6gwbf + 18gwcdf + 18gw^2cf^2 + ga + bf^2 + 6w^2bg^2 + 15cg^3w^4 + 2gbd + 3gcd^2)upv^3 - 30cg^3upv^7 - 2(b + 3cd + 3wcf + 3w^2cg)u^3pv - 6cgu^3pv^3;$

$M_2 := -2(a + bd + cd^2 - w + wbf + 2wcdf + w^2bg + 2w^2cdg + w^2cf^2 + 2w^3cfg + cg^2w^4)(d + fw + gw^2)^2upv - 2(-6wgd + 120w^3cdfg^2 + 12wcdf^3 + 60w^4cdg^3 + 40w^3cf^3g + 90w^4cf^2g^2 + 84w^5cfg^3 + 6agfw + 18bdgfw + 18bdg^2w^2 + 36cd^2gfw + 36cd^2g^2w^2 + 18w^2bf^2g + 30w^3bfg^2 + 72w^2cdf^2g + af^2 - 10g^2w^3 - 3wf^2 - 2df + 3bd^2g + 3bdf^2 + 4cd^3g + 6cd^2f^2 - 12gfw^2 + 3wbf^3 + 15w^4bg^3 + 6w^2cf^4 + 28cg^4w^6 + 2agd + 6ag^2w^2)upv^3 - 6(12gcdf^2 + 20gwcf^3 + 15g^2wbf + 60g^2wcdf + 60g^3w^2cd + 90g^2w^2cf^2 + 140g^3w^3cf + g^2a - 5g^2w - 2gf + cf^4 + 3g^2bd + 6g^2cd^2 + 15g^3w^2b + 70g^4cw^4 + 3gbf^2)upv^5 - 210cg^4upv^9 - 30g^2(bg + 4cdg + 6cf^2 + 28wcfg + 28cg^2w^2)upv^7 - 2(a + 3bd + 6cd^2 - w + 3wbf + 12wcdf + 3w^2bg + 12w^2cdg + 6w^2cf^2 + 12w^3cfg + 6cg^2w^4)u^3pv - 6(bg + 4cdg + 2cf^2 + 12wcfg + 12cg^2w^2)u^3pv^3 - 36cg^2u^3pv^5 - 6cu^5pv;$

$vars := \{a, b, c, d, f, g, u, v, w, p\};$
$polys := \{P_0, P_1, P_2, M_0, M_1, M_2\};$

The other examples we used to test the different solvers are well documented elsewhere. The complete sources are available from our Web site

`http://www.informatik.uni-leipzig.de/~compalg`

Example G1: [7, eq. (4)], see also [13, example G1]

Example G6: [7, eq. (8)], see also [13, example G6]

Example Go:
The (quasi)homogenized version of Gonnet's example from [2], see also [13, example Go].

Example	# eq.	# vars	# sol	Dimensions	Structure
A4	4	4	2	2x1	all rat. par.
E1	2	2	3	2x1 1x0	all reg. par.
G1	13	7	9	1x3 3x2 5x1	all reg. par.
G6	4	4	8	1x2 7x1	all reg. par.
Kri	6	10	6	3x9 3x4	see below
Go	19	18	7	1x7 1x6 2x5 3x4	see below
G7	12	10	20	4x6 4x5 11x4 1x3	see below

Table 3 : Input and output characteristics

Example G7: [8, eq. (6)], see also [13, example G7]

For the convenience of the reader, we collected in Table 3 some input and output characteristics of the systems under consideration as they may be computed using, e.g., our Reduce package CALI [12]. The number of solutions returned by the different CAS is not invariant, since it may vary due to different parameterizations. Indeed, even the simple system $\{y^2 - x\}$ may be parameterized either as $\{y = \pm\sqrt{x}, x = x\}$ or as $\{x = y^2, y = y\}$. `#sol` reports the number of isolated primes (over k), a geometric invariant. In the column *dimensions* an entry AxB indicates A components of dimension B among these primes.

The form of the output of the latter three examples has a more difficult structure: The 9-dimensional branches in Krider's example are $\{u = 0\}, \{v = 0\}$ and $\{p = 0\}$, whereas the four dimensional branches correspond to the different factors of $u^4 - 16$. Each of them contains another polynomial in f of degree 2. Hence by our experience obtained so far, we would expect that they are completely decomposed by Reduce into 8 rationally parameterized branches whereas Maple and Axiom will return 3 branches instead.

For Gonnet's example, the components may be rationally parameterized, but this is not obvious from the Groebner bases in the output collection of the Groebner factorizer regardless of the fact that they are already primes.

For G7 we refer to the end of this section.

Let us first report about the CAS that failed to give satisfactory answers for higher dimension:

MuPAD: Even for the very simple system E1 it returns the strange answer

$$\{[x = \texttt{RootOf}(-12x^3y + 2x^4y + 3x^3y^2, x), y = 0]\}$$

The polynomial system solver of the version 1.2.9 (seriously improved in release 1.3, see below) computes a single Groebner basis and extracts from the result a presentation of the solution that is not correct except for very simple cases.

Mathematica: We tested it with some of the easier examples above and got the following behaviour:

- For E1, it reports after 0.2 s. 25 zero dimensional solutions.
- For the example A4, see above.
- For G1, it reports after 1025 s. a list of about 10000 solutions with many repetitions of dimension ≤ 1 that we did not try to analyze.
- For Gonnet's example, it reports after 58.2 s. 20 solutions, all of dimension 4, containing only two really different ones (the same effect as for A4).

- The same applies to G6 : After 23.6 s. there were returned 6 one dimensional solutions, missing $\{\lambda_3 = \lambda_4 = 0\}$ and $\{\lambda_4 = \lambda_5 = 0, \lambda_1 = 1\}$.
- Kri and G7 it was unable to solve.

Macsyma: For A4, see above. The result of E1 is the expected one. All other nonzero dimensional examples in our test suite it was unable to solve in reasonable time (but see the report about the new version of the solver below).

Axiom: As already seen above with A4, the solver returns an error message for systems with infinitely many solutions. The Groebner factorizer can be accessed directly to decompose the system into pieces. With

```
groebnerFactorize polys
```

for A4 and E1, we get the answers

$$[[1],\ [z+x,\ y+w,\ w\,x+1],\ [z+x,\ y+w,\ w\,x-1]]$$

and

$$[[y-2,\ x-3],\ [y,\ x-6],\ [y],\ [1],\ [x]]$$

In both cases, superfluous (embedded) branches occur in the output collection. This is probably due to the recursive implementation of the Groebner factorizer. An early elimination of such branches may lead to a significant speed up of the computations, as shown by the example Go. Here Axiom returns 192 branches, but only 7 of them are really necessary. The same holds for Maple's Groebner factorizer implementation.

Due to our observations so far, we tried to calculate the examples mentioned above with the `Solve` facility of **Maple** and **Reduce** and the `groebnerFactorize` facility of **Axiom**. In Table 4 we collected the results of these experiments. The first column contains the corresponding computation (CPU-)time in seconds as reported from the system, the second the number of final branches in the answer. Since both Axiom and Maple usually return also embedded solutions that are completely covered by other branches, we report both the number of branches returned by the system and the number of essential branches among them.

Example	Reduce		Axiom		Maple	
	time	branches	time	branches	time	branches
G1	1.7	9	7.5	16/9	6.8	15/9
G6	0.75	8	13.5	12/8	15.7	8
Go	46.0	9	2022	192/7	2.3	10/7
Kri	12.3	11	5011	60/7	64.3	19/19
G7	125	33	1350	266/22	67.2	77/24

Table 4 : Run time experiments with different CAS

Some words about the quality of the output for the more advanced examples. As already explained above, the polynomial system solver passes through two phases: it first decomposes the system into smaller, almost prime components, and then tries to parameterize them. The second pass is not executed by Axiom's solver. For Gonnet's example, Maple was sufficiently smart to find the rational parameterization of all components (but couldn't remove embedded solutions), whereas Reduce introduced square root symbols for the parameterization of the components of dimension four. Both CAS successfully resubstituted their results into the polynomial system to be solved.

For Krider's example, Reduce returned the expected answer whereas Maple recognized the special biquadratic structure of the four dimensional branches and split them in an early stage of the computation. It returned 4 branches for each of the factors $u+2$ and $u-2$ and 8 branches for the factor u^2+4, thus splitting primes over $\mathbb{Q}$ into collections of primes over $\mathbb{Q}(i, \sqrt{2})$. Maple resubstituted its results successfully whereas Reduce couldn't handle its output during resubstitution.

For G7, the 20 prime components over $\mathbb{Q}$ were split during parameterization into smaller components over extension fields, introducing several square root symbols. Reduce returned 23 rational branches, 6 branches containing square roots of integers and 4 branches containing square roots of more complicated symbolic expressions. All of them could be managed to simplify to zero during resubstitution. Maple couldn't simplify one of the expressions obtained with a symbolic expression's square root during resubstitution.

9 Conclusions

Among the current versions of the general purpose CAS under consideration only Reduce (3.6) and Maple (V.3) offer satisfactory solve functionality for more advanced polynomial systems. Reduce was usually faster for those examples where it didn't try to introduce square roots into the representation of the solution. Maple (and Axiom) split all examples in a correct way, but usually returned superfluous components that were completely covered by other branches. Note the seriously improved behaviour of Maple compared to version V.2 as reported in [13]. Maple was the only system that, with some additional help, could handle `RootOf` symbols introduced during the solution process in a subsequent resubstitution step in a proper way.

Axiom (2.0) can decompose systems well (but not very fast), but there is not yet a facility integrated into the solver that allows systems with an infinite set of solutions to be handled.

Macsyma (420) and Mathematica (2.2) have serious problems, especially with higher dimensional systems, whereas the polynomial system solver of MuPAD (1.2.9) is in a very rudimentary state (but note that all three CAS improved their solvers meanwhile).

The algebraic solvers of Axiom, Maple, and Reduce are centered around an implementation of the Groebner factorizer whereas the other systems use different techniques, including the computation of (classical) Groebner bases. The latter are often less effective since they interweave factorization and Groebner basis computation in a less intrinsic way compared to the Groebner factorizer. For example, Macsyma's solver first computes a classical Groebner basis and calls the factorizer only on the resulting polynomials to split the system into smaller pieces. Afterwards, it applies resultant based elimination techniques to extract a triangular form for each of these components.

10 What's going on ?

As already explained in the introduction, the present paper can give no more than a snapshot of the state of the implementation of symbolic solving methods in the different CAS under consideration. Let's nevertheless add some remarks about developments going on or (almost) finished that we became aware of during the two years of preparation of this final version of our report.

First, due to the "general nonsense overhead", the implementational restrictions caused by the underlying (higher symbolic) programming language, a rigid hierarchy of code transparency, and the (mostly undocumented) hidden dependencies between different parts, general purpose CAS are not well suited for solving difficult advanced systems effectively. For *really hard systems*, the use of specially designed implementations is inevitable.

Such highly specialized, very effective systems (to name some of the widely used systems centered around the Groebner algorithm: CoCoA, GB, Macaulay, Singular) are top software products in the sense that they are on the top of a whole development pyramid and offer optimized implementations of advanced algorithms tested and refined formerly in more flexible (and thus less efficient) symbolic computation environments.

Besides the efforts of the big CAS to carry over the corresponding algorithmic knowledge (in a more or less efficient manner) into their own systems, recent research (e.g., the projects PoSSo, Frisco, OpenMath, Math-ML) is directed towards concepts of distributed computing that allows the advantages of the different special implementations to be combined directly. Opposite to the aim of general purpose CAS to *localize* the global power of problem solving competency on a single (desktop) computer, these efforts are directed towards *globalization* of the different specific local problem solving competencies into a network reaching far beyond the possibilities of classical scientific communication. They are conducted by efforts to develop methods, software, and interfaces that allow easy access to this network (at least) from the scientific community, thus leaving the general purpose CAS the role of advanced desk top calculators with merely an interface to this network. This makes it obsolete for them to pursue ultimate state of the art problem solving facilities, but increases the importance of easy handling, correctness and usefulness of results for small and medium sized problems when it is inconvenient and probably also too costly to contact the network for an answer.

Since these efforts are part of the beginning of general changes in the public information system that will influence human life in a very unpredictable way, it's hard to predict this development even in a near but not very near future. I don't dare to add my own predictions beyond the problem description so far.

Second, there are developments connected with next versions, releases, patches, etc. Only such changes will be reported below. We had the opportunity to work with beta releases or newly released versions of Macsyma (421, with new versions of the modules `algsys` and `triangsys`), Maple (V.4), Mathematica (3.0), and MuPAD (1.3 and 1.4.0). We acknowledge the kind support by Macsyma Inc., Wolfram Research Inc., and also the MuPAD development group supplying us with their development versions.

Macsyma: A new implementation of the modules `algsys` and `triangsys` offers also a `RootOf` symbol that doesn't expand algebraic numbers by default except for those of degree two. This follows Reduce's philosophy already discussed above. The new solver triangulates the given polynomial system using pseudo division and characteristic sets as proposed by D. Wang in [21] and [22]. This is a very interesting approach since it is the only general system solver implementation that completely avoids Groebner basis computations[7]. Such an approach is often superior compared to the traditional one for systems that are (almost) complete intersections, see, e.g., the timings for example 10 compared to the Groebner factorizer based solver of Reduce and the former Macsyma implementation.

A new operator `root_values` allows `RootOf` symbols to be expanded either symbolically (if possible) or numerically. This supplies the following functionality for `numSolve2`:

```
numSolve2(sol):=root_value(sol,sfloat);
```

To get sufficient accuracy for some of the examples (marked with *) one has to use double precision `dfloat` instead of single precision `sfloat`.

We tested the new version Macsyma 421 on a Sun UltraSparc 1 and collected in Table 5 a the behaviour of the new solver on our zero dimensional examples. Example 5 and 7 remained out of the scope of the solver.

Altogether one may conclude that Macsyma's solve facility was seriously improved in both the symbolic and the numerical directions, although, as for Maple, we found no way to get numerical approximations for the solutions of a polynomial system directly. Note that all solutions not containing `RootOf` symbols were simplified correctly during resubstitution, even the answer to example 10 that contains many complicated square root expressions, but expressions containing

[7] D. Wang developed a package for Maple's share library that uses the same approach, see below.

Example	symbSolve		numSolve2		
	time	structure	time	structure	resubst.
1	0.3	(8x1)	0	8 sol.	ok.
2	0.04	(4x4)	0.05	16 sol.	ok.
3	0.79	(2x1 1x6)	0.36	8 sol.	*
4	0.57	(3x1 2x3 3x6)	0.59	27 sol.	ok.
6	4.36	(3x1 1x24)	73.1	5 sol.	*
8	0.2	2 rat. par.	—		
9	0.2	3 rat. par.	—		
10	4.6	4 branches	—		

Table 5 a: Zero dimensional systems with Macsyma 421

Example	Time	Structure/Comments	Resubst.
A4	0.6	14 solutions, 12 of them embedded.	ok.
E1	0.1	6 solutions, 3 of them embedded	ok.
G1	243 151	44 sol. with dim=(2x3 15x2 27x1)	42 of 44
Go	8.1	13 sol. with dim=(1x7 1x6 3x5 8x4)	ok.

Table 5 b: Higher dimensional systems with Macsyma 421

`RootOf` symbols are not reduced with respect to the corresponding characteristic polynomial and hence resubstitution fails.

The situation is much worse with systems of positive dimension. This is partly due to the fact that the pseudo division approach has serious problems detecting embedded solutions[8]. With A4 and E1 we got almost the same results as earlier (some more embedded components). With G1 we got 44 solutions, where 42 of them proved to be correct during resubstitution (note that the number of three dimensional components differs from Table 3 due to the parameterization). The remaining components contain the (suspicious since decomposable) `RootOf` expression

$$l_3 = \pm\,\mathtt{RootOf}\left(\frac{1243351456137\,\sqrt{2}\,\sqrt{7}\,r_1^{\frac{15}{2}}\,l_3^3}{32},\ l_3\right)$$

Table 5 b reports the behaviour of the new solver on our higher dimensional examples. Examples G6, Kri, and G7 remained beyond the scope of Macsyma.

Note that in general the results are expressed through rational expressions in the algebraic numbers introduced with the `RootOf` symbols, but neither numerators nor denominators are reduced with respect to the characteristic polynomials of these numbers.

Maple: Version V.5, distributed since April 1998, contains a completely rewritten Groebner package. This influences the performance and behaviour of the solver, of course. Table 6 contains the results of sample computations with the new Maple version on a Sun UltraSparc and some of our examples.

The second line of the table lists the CPU-time (in seconds) to compute the solution of the given example with the `solve` command and the third line lists the CPU-time for the resubstitution test on the (algebraic) output. It was unable to finish example 5 after 12 h CPU-time. The resubstitution test was completed successfully except for G7 where only another call to `simplify` led to the desired zero result. The last line of the table contains both the number of solutions with

[8] Note that for components that don't admit a regular parameterization, this remains also a theoretically difficult question.

Example	4	6	7	10	G1	G6	Go	Kri	G7
c-time	1.05	2.38	21.3	0.8	2.4	2.5	1.2	1.9	16.2
r-time	0.1	46.4	1907	0.3	0.3	0.1	0.1	0.4	0.5
# sol.	6/39	3/27	4/–	1/4	10	8	7	6/11	33/38

Table 6: Sample computations with Maple V.5

Example	Time	Remarks on the output
A4	1.0	8 embedded solutions as with Macsyma 420
ex. 1	0.25	correct but unsimplified
ex. 3	2.15	non-reduced rational expressions in algebraic numbers
ex. 4	18.1	degree=(9x1 3x6)[9]
ex. 5	–	error: object too large
ex. 6	8.1	degree=(3x1 1x24)
ex. 7	114	degree=(4x1 1x12)
ex. 9	0.3	3 sol.
ex. 10	21.0	4 sol.
E1	0.1	2 embedded solutions
G1	13.9	14 sol. with dim=(1x3 5x2 8x1)
G6	3.4	10 sol. with dim=(1x2 9x1)
Go	72.9	18 sol. with dim=(1x7 3x6 7x5 7x4)
Kri	19.1	19 sol. with dim=(3x9 8x4 8x3)

Table 7: The solver of the `charsets` package of D. Wang

`RootOf` expressions and the number of expanded solutions, i.e., either symbolic expansions (roots up to degree 4) or numerical approximations (zero dimensional solutions of higher degree) if they were different. Note that the number of branches for example 4 is still wrong. For example 7, the expansion failed with the error "`in allvalues/genall division by zero`".

In some zero dimensional cases, Maple V.5 now returns rational instead of (fully simplified) polynomial `RootOf` expressions as possible for algebraic extensions. This concerns, e.g., example 6 and is in the spirit of the Generalized Shape Lemma in [1, 2.9.]. These rational expressions with coefficients of moderate size expand with a subsequent call to `simplify` to the former polynomials with huge coefficients.

Note that there are two more Maple packages that implement tools for the solution of polynomial systems. One of them is the package `moregroebner` by K. Gatermann that extends the classical Groebner algorithm to modules and more flexible term orderings. The other one is D. Wang's implementation of the Ritt-Wu characteristic set method in the package `charsets` of the shared library `algebra`. The latter offers its own solver `csolve` with a performance slightly better than Macsyma. But also in this implementation the Ritt-Wu method has some problems detecting and removing embedded solutions. Table 7 contains more detailed results of our sample computations.

Mathematica: In late summer 1996, Wolfram Research Inc. launched version 3.0 of Mathematica with serious improvements in almost all parts of the CAS. The improvements concerning the area of polynomial system solving are mainly related to a new `RootOf` philosophy: `RootOf`$(f(x))$ now contains additionally a counter to address the different roots of $f(x)$ individually thus resolving the data type design trouble explained above.

`Solve[f(x) == 0,x]` returns a substitution list with exactly $deg(f)$ items, possibly with repetitions, that are either of the form `Root[f(x),k]` if $f(x)$ is irreducible or indecomposable (i.e., not

[9] One of the four solutions of degree 6, see Table 1 b, is a nested tower of degrees 2 and 3 respectively, that is resolved by Cardano's formula into 6 branches of degree 1.

of the form $f(x) = g(h(x))$) or are simplified by the obvious rules if $f(x)$ is reducible or of small degree. For example, for $poly = x^5 - x + 1$

    Solve[poly == 0,x]

now yields

$$\{\{x \to \texttt{Root}[1 - \#1 + \#1^5, 1]\}, \{x \to \texttt{Root}[1 - \#1 + \#1^5, 2]\},\\ \{x \to \texttt{Root}[1 - \#1 + \#1^5, 3]\}, \{x \to \texttt{Root}[1 - \#1 + \#1^5, 4]\},\\ \{x \to \texttt{Root}[1 - \#1 + \#1^5, 5]\}\}$$

The different roots of $f(x)$ are distinguished by their approximate complex values. There is a great variety of functions to deal with such algebraic numbers as, e.g., summation, parametric differentiation, computation of minimal polynomials of derived expressions, computation of primitive elements, etc.

Together with this new representation of algebraic numbers, the simplifier was improved for such objects. This yields for the output of example 1 expressions without nested roots that may be transformed into the simple form returned by Macsyma and Reduce with another application of the new operator `FullSimplify`. For example 2, Mathematica returns a list of 16 complicated radical expressions, that are properly simplified symbolically under resubstitution and also numerically. For example 3, the result consists of 8 explicit solutions where 6 of them differ only in the component number of the corresponding `Root` expressions as expected.

With the new `RootOf` syntax and the improved algebraic simplifier, Mathematica has no more problems substituting `RootOf` symbols into algebraic expressions and simplifying them. All resubstitution tasks for the examples, where the solution contains unresolved `RootOf` expressions, were executed with full success.

But also the new version couldn't solve examples $4 - 7$ within reasonable time and space. There is also only little progress solving the polynomial systems of positive dimension. Now the system E1 is solved properly, but with repetitions of the partial solutions $\{x \to 0\}$ and $\{y \to 0\}$. For G1, the system returned after 1796 s. 478 solutions, among them 388 times the partial solution $\{\lambda_1 = \lambda_2 = \lambda_4 = \lambda_5 = \lambda_6 = \lambda_7 = 0\}$. For Gonnet's example, it returns after 64 s. 9 solutions, two of them being really different. Kri and G7 remain unsolved.

MuPAD: Since version 1.2.9, MuPAD was seriously improved. Version 1.3 already produced correct answers for most of our zero dimensional examples implementing Groebner factorizer based methods into the solver.

The syntax of `solve` was extended with a third optional parameter

    sol:=solve(polys,vars,options)

These options may be

- `MaxDegree` to control the maximum degree of irreducible polynomials whose roots are given in closed form (if possible). The default is 2 (as in Reduce).
- `BackSubstitution` to enable a back-substitution step to be performed on the solution. The default is FALSE, since this usually leads to coefficient size explosion.

The new version offers also a (direct) numerical solver *numSolve1* via

    float(hold(solve)(polys,vars))

and the numerical expansion *numSolve2* of symbolic solutions with the (yet undocumented) function `allvalues` that (in version 1.4) expands a single solution tuple into a set of numerical approximations. Hence

Ex.	symbSolve		numSolve1		numSolve2	
	time	structure	time	structure	time	structure
1	1.95	(8x1)	1.34	8 sol.	0.01	8 sol.
2	0.53	(4x4)	3.06	16 sol.	0.61	16 sol.
3	3.02	(2x1 1x6)	3.27	8 sol.	0.37	8 sol.
4	4.18	(3x1 2x3 3x6)	5.19	27 sol.	1.19	27 sol.
5	84.1	(10x1 10x4 1x20)	96.0	70 sol.	5.20	70 sol.
6	12.3	(3x1 1x24)	31.7	27 sol.	19.8	27 sol.
7	64 437	(4x1 1x12)	64 421	16 sol.	1.49	16 sol.

Table 8: The behaviour of MuPAD 1.4 for zero dimensional examples without parameters

```
map(sol,op@allvalues)
```

will expand a set *sol* of symbolic solutions numerically.

This leads to satisfactory results for all zero dimensional and easy general systems in our test suite. In Table 8, we collected some data of the computations we did with MuPAD 1.4. They report correct output characteristics and reasonable timings.

The timings suggest, that NumSolve1 does probably the same as SymbSolve followed by NumSolve2. Resubstitution of the numerical values proved the results to be correct, but resubstitution of the symbolic results failed, if the solution contained `RootOf` symbols, since they are not simplified even according to the obvious degree reduction rules.

The parametric zero dimensional systems in examples 8 - 10 were also solved successfully. Even for example 10, we got (after 52 s.) four solutions with quite simple square root expressions. Note that `vars` is a *set*, hence the variable order is chosen by the system, so the computations executed by the different systems may vary.

The situation is much worse for systems with infinitely many solutions. E1, A4 and Kri are solved correctly. For the latter, the result (returned after 124 s.) had the same form (11 branches) as produced by Reduce.

For G1, the system returned after 190 s. 16 solutions with 5. . . 7 entries each. Hence, (since there are 7 variables) one would expect solutions of dimension 0. . . 2 with the missing variables as parameters. But a more detailed analysis of the output shows that some of the solutions contain an entry $l_4 = l_4$ thus binding a really free parameter. Four of the solutions contained even (one branch of) the very suspicious expression

$$l_4 = \pm\frac{\sqrt{30}}{15}\sqrt{\frac{15}{2}\,l_4^2}.$$

Altogether we've got 2 solutions of dimension 2, 13 (including the 4 suspicious ones) of dimension 1 and another of dimension 0, thus missing at least the 3-dimensional component.

The same applies to G6: We've got (after 24 s.) 11 rationally parameterized solutions, 7 of dimension 1 and 4 of dimension 0. The zero dimensional solutions turned out to be embedded; the 2-dimensional solution was missing.

Go and G7 remained unsolved after more than 24 h computing time.

References

[1] Mariemi et al. Alonso. Zeroes, multiplicities and idempotents for zero dimensional systems. In T.Recio L.Gonzalez-Vega, editor, *Proc. MEGA-94*, number 43 in Prog. Math., pages 1–15.

Birkhäuser, 1996.

[2] W. et al. Boege. Some examples for solving systems of algebraic equations by calculating Gröbner bases. *J. Symb. Comp.*, 2:83 – 98, 1986.

[3] D.A. Cox, J.B. Little, and D.B. O'Shea. *Ideals, Varieties and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Undergraduate Texts in Mathematics. Springer, New York, 1992.

[4] S. R. Czapor. Solving algebraic equations via Buchberger's algorithm. In *Proc. EUROCAL'87*, volume 378 of *Lect. Notes Comp. Sci.*, pages 260 – 269, 1987.

[5] S. R. Czapor. Solving algebraic equations : Combining Buchberger's algorithm with multivariate factorization. *J. Symb. Comp.*, 7:49 – 53, 1989.

[6] James H. Davenport. Looking at a set of equations. Technical Report 87 - 06, Univ. Bath, Comp. Sci. Dept., 1987.

[7] V. P. Gerdt, N. V. Khutornoy, and A. Yu. Zharkov. Solving algebraic systems which arise as necessary integrability conditions for polynomial nonlinear evolution equations. In Shirkov, Rostovtsev, and Gerdt, editors, *Computer algebra in physical research*, pages 321 – 328. World Scientific, Singapore, 1991.

[8] V. P. Gerdt and A. Yu. Zharkov. Computer classification of integrable coupled KdV-like systems. *J. Symb. Comp.*, 10:203 – 207, 1990.

[9] P. Gianni, B. Trager, and G. Zacharias. Gröbner bases and primary decomposition of polynomial ideals. *J. Symb. Comp.*, 6:149 – 167, 1988.

[10] M.J. Gonzalez-Lopez and T. Recio. The ROMIN inverse geometric model and the dynamic evaluation method. In Cohen, editor, *Computer algebra in industry*, pages 117 – 141. John Wiley, 1993.

[11] Hans-Gert Gräbe. Two remarks on independent sets. *J. Alg. Comb.*, 2:137 – 145, 1993.

[12] Hans-Gert Gräbe. CALI – a Reduce package for commutative algebra, version 2.2.1. Available via WWW from `http://www.informatik.uni-leipzig.de/~compalg`, 1995.

[13] Hans-Gert Gräbe. On factorized Gröbner bases. In Fleischer, Grabmeier, Hehl, and Küchlin, editors, *Computer algebra in science and engineering*, pages 77 – 89. World Scientific, Singapore, 1995.

[14] Hans-Gert Gräbe. Triangular systems and factorized Gröbner bases. In *Proc. AAECC-11*, volume 948 of *Lect. Notes Comp. Sci.*, pages 248 – 261, 1995.

[15] Daniel Lazard. Solving zero dimensional algebraic systems. *J. Symb. Comp.*, 13:117 – 131, 1992.

[16] H. Melenk, H.-M. Möller, and W. Neun. Symbolic solution of large chemical kinematics problems. *Impact of Computing in Science and Engineering*, 1:138 – 167, 1989.

[17] Herbert Melenk. Practical applications of Gröbner bases for the solution of polynomial equation systems. In Shirkov, Rostovtsev, and Gerdt, editors, *Computer algebra in physical research*, pages 230 – 235. World Scientific, Singapore, 1991.

[18] B. Mishra. *Algorithmic Algebra*. Springer, New York, 1993.

[19] H.-M. Möller. On decomposing systems of polynomial equations with finitely many solutions. *J. AAECC*, 4:217 – 230, 1993.

[20] Rafael Sendra and Franz Winkler. Parameterization of algebraic curves over optimal field extensions. *J. Symb. Comp.*, 23:197 – 207, 1997.

[21] Dongming Wang. An elimination method based on Seidenberg's theory and its applications. In Eyssette and Galligo, editors, *Computational Algebraic Geometry*, pages 301 – 328. Birkhäuser, Basel, 1993.

[22] Dongming Wang. An elimination method for polynomial systems. *J. Symb. Comp.*, 16:83 – 114, 1993.

Appendix: Some code fragments

Assuming that `vars` and `polys` are defined as in the text, we collected for the different CAS the commands to be issued for timing, linear output printing (to analyze the output) and the code fragments to compute the list of symbolic solutions `sol`, to compute the list of numerical solutions `numsolve1(polys)`, to convert `sol` into a list of approximate solutions `numsolve2(sol)`, and to check each of them by resubstitution.

Axiom:

The following command computes a (substitution) list of solutions:

```
sol:=solve(polys,vars);
```

Numerical solutions may be found with the procedures

```
numsolve1(polys) == complexSolve(polys,0.0001);
numsolve2(sol) == concat([complexSolve(u,0.0001) for u in sol]);
```

The following procedure executes the resubstitution task

```
resubst(sol,polys) == [[subst(u,v) for u in polys] for v in sol];
```

To get CPU time reported, issue

```
)set message time on
```

at the beginning of the session. There is no natural way to tell the system to return *linear* output in human readable form (only coercion to InputForm returns linear output, but in a Lisp like notation).

Macsyma:

For linear output and timing, issue

```
showtime:true$
display2d:false$
```

at the beginning of the session. Solutions may be found with

```
sol:solve(polys,vars);
```

and resubstitution may be tested with the procedure

```
resubst(sol,polys):=
    map(lambda([elem],ratsimp(subst(elem,polys))),sol)$
```

Mascyma 421 allows one also to produce numerical approximations from a symbolic solution:

```
numsolve2(sol):=root_value(sol,sfloat)$
```

Maple:

The following code refers to version V.5. Timings may be obtained from time stamps at the beginning and the end of a procedure, e.g., with the instructions

```
tt:=time(); sol:=[solve(polys,vars)]; time()-tt;
```

Note that `solve` returns an expression sequence, that for convenience was subsequently transformed into a list. Then

```
nops(sol);
lprint(sol);
```

allows one to count the number of solutions and to print out all of them in a linear format. The procedures

```
expandsol := proc(sol) map(allvalues,sol,dependent) end;
numsolve2 := proc(sol) map(evalf,expandsol(sol)) end;
resubst:=proc(sol,polys) map(simplify@subs,sol,polys) end;
```

expand `RootOf` symbols, compute numerical approximations and execute the resubstitution task, respectively.

Mathematica:

Mathematica doesn't support the shortcut that the solver interpretes a polynomial input f as the equation $f = 0$. Hence one must first transform a list of polynomials into a list of equations. This can be done by mapping the pure function `#==0&` onto the list `polys`.

Timings may be obtained by calling `Timing` with the procedure to be timed as an argument. Using the special postfix notation of Mathematica for procedures with a single argument

```
res=Solve[#==0&/@ polys,vars]//Timing
time=res[[1]]
sol=res[[2]]
```

then computes a record `res` with the elapsed time as first entry and the solution as the second one. Similarly, the procedures

```
numsolve1[polys_]:=NSolve[#==0&/@ polys,vars]//Timing
numsolve2[sol_]:=sol//N//Timing
resubst[sol_,polys_]:=polys/.sol//Simplify
```

compute numerical approximations and execute the resubstitution task, respectively. Linear output may be produced with `sol//InputForm`.

MuPAD:

In MuPAD 1.4. timings and results for `symbSolve`, `numSolve1` and `numSolve2` may be obtained in a similar way as in Mathematica:

```
tt:=time((sol:=solve(polys,vars))); sol;
tt:=time((sol1:=numsolve1(polys,vars))); nops(sol1);
tt:=time((sol2:=map(sol,op@allvalues))); nops(sol2);
```

remember the elapsed time and return also the set of solutions (respectively their number). Here `numsolve1`, the numeric solver, should be defined as

```
numsolve1:=proc(polys,vars)
    begin float(hold(solve)(polys,vars)) end_proc;
```

In most cases, the resubstitution test may be done with the following procedures:

```
mysubs:=proc(a,b) begin simplify(subs(b,op(a))) end_proc;
resubst:=proc(sol,polys) begin map(sol,mysubs,polys) end_proc;
```

Linear output may be produced by setting `PRETTY_PRINT:= FALSE`.

Reduce:

As in Macsyma, linear output and timing are achieved through global switches

```
off nat; on time;
```

The list of symbolic solutions may be found with

```
sol:=solve(polys,vars);
```

To produce numerical solutions, one should switch to rounded mode, e.g., with the procedures

```
procedure numsolve1(polys);
    << on rounded; write solve(polys,vars); off rounded; >>;
procedure numsolve2(sol);
    << on rounded;
       write for each x in expand_cases(sol) collect sub(x,x);
       off rounded;
    >>;
```

Note that the (at a first glance) suspicious `sub(x,x)` in the second procedure performs back substitution. The resubstitution task may be executed with the procedure

```
procedure resubst(sol,polys);
    for each u in sol collect sub(u,polys);
```

In some places this doesn't handle back substitution satisfactory and you have to use instead

```
procedure resubst1(sol,polys);
    for each u in sol collect sub(u,sub(u,polys));
```