Algebraic Numbers in Symbolic Computations

Hans-Gert Gräbe
Institute of Computer Science
Univ. Leipzig/Germany

September 25, 1998

Abstract

There are many good reasons to teach a course on a systematic introduction to symbolic methods not only to students of mathematics but also to those of technical sciences. The design of such a course meets an essential difficulty since the principles to be demonstrated appear only in non trivial applications in a convincing way, but there is no time to teach the necessary contexts to a large extend. Hence the material intended to demonstrate different effects has to be chosen with great care.

The goal of this paper is to show that for such a course algebraic numbers are not only interesting by their mathematical content but also as a complex target where different concepts and principles of symbolic computations become apparent. Thus they may serve at once as a non trivial application of the basic concepts, notations and principles developed earlier in such a course.

1 Introduction

With increasing computing power desktop computers become more and more indispensable tools for the intellectual work not only of scientists but also of engineers and even the middle technical staff. During these changes computer programs substitute tables and handbooks that accompanied scientific and technical activities for a long time. Computers allow not only a more rapid and flexible access to the same information as before but enable also its more advanced presentation in a generic and dynamic fashion. New tools, based on the capabilities of modern general purpose symbolic systems (CAS for short to remind the computer algebra origin of the systems that I have in mind at this point), may generate algorithmically at runtime very specialized information from a huge knowledge base thus making it possible even to medium gifted technical staff to obtain non standard solutions not explicitly foreseen during design time of the tool. This largely enhances their possibilities compared to print based tools where all useful information had to be compiled in advance. Due to these restrictions print based tools usually present only a small number of standard situations in a rigid format.

There is no doubt that symbolic methods will play a central role both in the presentation and extraction of such knowledge and the forthcoming scientific and technical staff will be faced with the demand to apply these techniques intelligently to its everyday's problems. Thus there is increasing need for including a corresponding component of instruction into university and even possibly high school courses.

Growing efforts to incorporate the use of such tools into regular professional courses, e.g., lectures of mathematics, physics or different engineering courses mark the way the academic community meets this challenge. But even in places missing (yet) such activities students have access to a fast growing literature to acquire these skills, and they do so more and more. Hence we may assume that in a near future students finishing their undergraduate studies will have practical experience using symbolic computer based methods.

Due to the central role of such experience for the future professional life of students graduating not only in mathematics but also in technical sciences it is desirable to continue with a *systematic* introduction into basic concepts, notations and principles behind these methods to qualify the students' exploitation of the power of these tools. Such a course should not concentrate on the more qualified exploitation of a *particular* CAS but, as courses "Algorithms and Data Structures" do for classical computer applications, on the basic instruments and notations common to all or most of the different CAS (at least of second generation).

Since these students usually have a good understanding of the same questions for classical (numerical) computer applications such a course may concentrate on the main notations and principles where symbolic computations differ from numerical ones. These differences constitute a quite complex topic that starts with special design questions resulting from overlapping name and value spaces and extensive use of dynamic data structures, continues with differing notions of functions, the (unknown to classical imperative computer languages) notion of simplification and end up with subtle mathematical questions on simplifications that are allowed and those that are not. A better understanding of these basics usually leads also to a better understanding of the special, in some places quite unexpected behavior of CAS and how it may be avoided or why not.

The design of such a course meets an essential difficulty. On the one side, CAS are multi purpose tools capable of very complicated symbolic computations in very complex mathematical contexts and the principles to be demonstrated appear only in non trivial applications in a convincing way. Such applications can't be well understood without some knowledge of their context. On the other hand, students coming from technical or computer science departments usually have not very advanced mathematical knowledge. Thus the material intended to demonstrate different CAS effects has to be chosen with great care.

The aim of this paper is to show by means of examples that algebraic numbers are a topic that deserves consideration as a complex application area where different concepts and principles of symbolic computations become apparent. These examples and explanations constitute the final part of a course "Intro-

duction to Computer Algebra" taught by the author during the last years.

To convince the students that the presented conclusions are independent of a special CAS I used for the course (due to historical reasons) Derive, Maple, and Reduce, additionally partly MuPAD and Mathematica in a seminar and also (off line) comparisons with Macsyma and Axiom. This on a first glance confusing variety of software used mainly in a demonstration mode allows both clearly to extract the underlying common syntactical structures and to incorporate the heterogeneous students' experience (mainly with Maple, MuPAD and Mathematica).

A good understanding of the special nature of algebraic numbers in symbolic computations is central also by another reason. There is a great difference between the understanding of (non rational) real numbers from the (common to the students) point of view of numerical and that of symbolic computation. The differing understanding of the nature of algebraic numbers is possibly even the branching point between both disciplines. And whereas one may be willing to accept the symbolic nature of transcendent numbers as e or π it is hard to recognize the difference between the "numbers" $\sqrt{2}$ and $2/3$ both containing ciphers and a certain additional symbol. The more since CAS usually know much about square roots as, e.g.,

$$\sqrt{-6\sqrt{2}+11}+\sqrt{6\sqrt{2}+11} = 6 \tag{1}$$

$$\sqrt{-2\sqrt{6}+5}+\sqrt{2\sqrt{6}+5} = 2\sqrt{3} \tag{2}$$

$$(6\sqrt{3}-10)^{\frac{1}{3}}+(6\sqrt{3}+10)^{\frac{1}{3}} = 2\sqrt{3} \tag{3}$$

thus conveying the impression that the arithmetic of algebraic numbers given as nested roots is easy and completely covered by the system.

R. Penrose discusses in chapter 4 of his famous book (Penrose, 1989) the great difficulty to develop a concise understanding of real numbers starting from a numerical point of view taken up by most of the students. On the other hand algebraic numbers are very ubiquitous in symbolic applications and CAS produce them usually one after the other. Hence a systematic introduction into symbolic computation must treat algebraic numbers as one of the central topics.

2 What's that – algebraic numbers ?

Mathematically algebraic numbers are defined as the roots of univariate polynomials and there are many reasons to define them even in such a way also in a CAS. But this is not the form in that such numbers appeared first historically nor in the students' perception. They know algebraic numbers mainly as root expressions and one of the aims of the course will be the explanation of the limitations of such an understanding.

Hence we start with such an intuitive notion of algebraic numbers and first try to understand to what extend algebraic numbers differ from usual, i.e.,

integer and fractional ones. This is a reasonable question since expressions like "52" or "1/4" are also merely symbolic representations. But different to integers and fractional numbers where the CAS (and the students) know how to obtain automatically a canonical form of the result of any arithmetic expression containing such entities expression like $\sqrt{2}$ are only simplified along obvious rules as, e.g.,

$$\begin{aligned} x &:= 2\sqrt{2} + 3\sqrt{3} \\ y &:= 3\sqrt{2} - 2\sqrt{3} \\ x + y &= 5\sqrt{2} + \sqrt{3} \\ x - y &= -\sqrt{2} + 5\sqrt{3} \end{aligned}$$

For more complicated expressions as, e.g.,

$$\sqrt{2} + \sqrt{3} + \frac{1}{\sqrt{2} - \sqrt{3}} \tag{4}$$

we need some more effort to see that the expression is zero (even Maple and Derive don't find out that automatically) . We conclude that algebraic numbers given as root expressions are, by some not yet visible reason, not well suited for computations since even the zero decision problem is not (always) solved automatically.

With such a first inspection in mind it is very surprising that CAS nevertheless seem to know much even about nested roots as pointed out in the introduction. Consider these three identities once more. Knowing the relations

$$\begin{aligned} 11 \pm 6\sqrt{2} &= (3 \pm \sqrt{2})^2, \\ 5 \pm 2\sqrt{6} &= (\sqrt{3} \pm \sqrt{2})^2, \\ 6\sqrt{3} \pm 10 &= (\sqrt{3} \pm 1)^3 \end{aligned}$$

they are evident, but the systems don't know that in advance. Thus let us compare the behavior of different CAS on these examples to obtain more insight into the question. The answers are collected in table 1.

We see that most of the systems are strong in this area but don't invoke that knowledge automatically. With regard to (4) the automatic simplification of nested roots in Derive and Maple is rather obscure. But even for Axiom and Reduce, that have no such facilities built in, one may easily design rules that allow to perform these simplifications, of course developing the corresponding mathematical background first.

3 Algebraic numbers and mathematical exactness

One may wonder about the third result in table 1 returned by Maple and Macsyma. Are they strong only for square roots? The reason is another one – mathematical correctness. Computed over the complex numbers we remember

CAS	(1)	(2)	(3)
Axiom	—	—	—
Derive	automatically	automatically	automatically
Macsyma	`denest_sqrt`	`denest_sqrt`	—
Maple	automatically	automatically	—
Mathematica	`RootReduce`	`RootReduce`	`RootReduce`
MuPAD	`radsimp` or simplify(_,sqrt)	`radsimp` or simplify(_,sqrt)	`radsimp` or simplify(_,sqrt)
Reduce	—	—	—

Table 1: Simplification of nested roots

that root expressions are many-valued functions and there is no canonical way to choose one of the branches. This is a great step towards the algebraic understanding of algebraic numbers: The symbolic expression doesn't identify the underlying algebraic object uniquely. This ambiguity may be resolved for real numbers but there are many reasons to assume the argument and return type of root expressions to be complex numbers. It would be nice to restrict the domain of definition by the user if possible, e.g., choosing the real branch of the root only, but I found no possibility to convince any of the systems under consideration to do that in this case.

4 Algebraic numbers, variables, and kernels

Let's have a closer look, how the different systems compute with algebraic numbers. Despite (4) simple square roots are not very interesting since any root of a polynomial of degree 2 may be expressed arithmetically through the polynomial's coefficients and a single root expression $\sqrt{D}$ in a well known way. So let us examine the possibility of the different CAS to compute with algebraic numbers of degree 3.

To be fair, we will use algebraic numbers produced by the CAS themselves and perform some computations on these numbers. We prepared the following task:

> Take a polynomial
>
> $$f(x) = x^3 + p\,x + q$$
>
> of degree 3 with integer coefficients, compute its zeroes x_1, x_2, x_3 and subsequently compose and expand the expression
>
> $$(x - x_1)(x - x_2)(x - x_3)$$
>
> (that should simplify to the original polynomial $f(x)$).

If this task is executed successfully test moreover whether the CAS can sum up powers of the roots

$$s_k := x_1^k + x_2^k + x_3^k$$

that are known to yield again integers.

Here is the result of the first part of this task produced by Derive (version 3.04)

```
1: x^3+x+1
```

```
2: factor complex #1: (0.4 s)
```

$$\begin{aligned}
&\left(x + \left(\tfrac{1}{18}\sqrt{93} + \tfrac{1}{2}\right)^{1/3} - \left(\tfrac{1}{18}\sqrt{93} - \tfrac{1}{2}\right)^{1/3}\right) \\
&\quad \cdot \left(x + \left(\tfrac{1}{144}\sqrt{93} - \tfrac{1}{16}\right)^{1/3} - \left(\tfrac{1}{144}\sqrt{93} + \tfrac{1}{16}\right)^{1/3}\right. \\
&\qquad \left. -i\left(\left(\tfrac{3}{128}\sqrt{93} + \tfrac{29}{128}\right)^{1/6} + \left(\tfrac{29}{128} - \tfrac{3}{128}\sqrt{93}\right)^{1/6}\right)\right) \\
&\quad \cdot \left(x + \left(\tfrac{1}{144}\sqrt{93} - \tfrac{1}{16}\right)^{1/3} - \left(\tfrac{1}{144}\sqrt{93} + \tfrac{1}{16}\right)^{1/3}\right. \\
&\qquad \left. +i\left(\left(\tfrac{3}{128}\sqrt{93} + \tfrac{29}{128}\right)^{1/6} + \left(\tfrac{29}{128} - \tfrac{3}{128}\sqrt{93}\right)^{1/6}\right)\right)
\end{aligned}$$

```
3: expand #2 (1.3 s)
```

$$\begin{aligned}
&x^3 + x\left(\tfrac{1}{1152}\sqrt{93} + \tfrac{29}{3456}\right)^{1/3} + x\left(\tfrac{29}{3456} - \tfrac{1}{1152}\sqrt{93}\right)^{1/3} + x\left(\tfrac{3}{128}\sqrt{93} + \tfrac{29}{128}\right)^{1/3} \\
&\quad + x\left(\tfrac{29}{128} - \tfrac{3}{128}\sqrt{93}\right)^{1/3} - x\left(\tfrac{1}{18}\sqrt{93} + \tfrac{29}{54}\right)^{1/3} - x\left(\tfrac{29}{54} - \tfrac{1}{18}\sqrt{93}\right)^{1/3} \\
&\quad + x + \left(\tfrac{7}{7776}\sqrt{93} + \tfrac{5}{576}\right)^{1/3} + \left(\tfrac{1}{31104}\sqrt{93} - \tfrac{1}{3456}\right)^{1/3} + \left(\tfrac{7}{288}\sqrt{93} + \tfrac{15}{64}\right)^{1/3} \\
&\quad + \left(\tfrac{1}{1152}\sqrt{93} - \tfrac{1}{128}\right)^{1/3} + \left(\tfrac{1}{486}\sqrt{93} + \tfrac{1}{54}\right)^{1/3} - \left(\tfrac{1}{31104}\sqrt{93} + \tfrac{1}{3456}\right)^{1/3} \\
&\quad - \left(\tfrac{7}{7776}\sqrt{93} - \tfrac{5}{576}\right)^{1/3} - \left(\tfrac{1}{1152}\sqrt{93} + \tfrac{1}{128}\right)^{1/3} - \left(\tfrac{7}{288}\sqrt{93} - \tfrac{15}{64}\right)^{1/3} \\
&\quad - \left(\tfrac{1}{486}\sqrt{93} - \tfrac{1}{54}\right)^{1/3}
\end{aligned}$$

Even another call to `simplify` doesn't change the result. One may wonder about this but there is a simple explanation that the students may guess with their knowledge about the simplification mechanism of CAS obtained in an earlier part of the course. Nested root expressions as, e.g., $(U)^{1/3}$, are (conceptually) internally represented as `Power(U,1/3)` and thus much closer to functional expressions as $\sin(2\,x - y)$, $\cos(x^2 + y^2)$ etc., than to numbers. Hence the general simplification mechanism for functional expressions applies to them (and may be studied on this target in more detail).

For efficiency reasons this general simplification mechanism usually consists of a (cheap since efficiently decidable) polynomial or rational simplificator combined with a (in full generality not decidable) rule based simplification system

associated with functional symbols. A functional expression is simplified at three layers: The outer layer is a polynomial simplification of the context that regards the expression as a generalized variable, called *kernel*, the inner layer a polynomial simplification of the arguments of the expression and the middle one a rule based simplification combining the inner and the outer world otherwise separated by the function name as by a "wall".

Hence in our example expression `#3` containing several root symbols is internally not represented as a polynomial in x with number coefficients, but as a (multivariate) polynomial in several kernels. Different to polynomials in indeterminates, these kernels are algebraically (and here even linearly) dependent. Indeed, a more detailed inspection "by hand" shows that all these kernels are merely different multiples of two different third roots. Probably Derive "forgot" this common origin and considers all the roots as independent. Note that, once forgotten, these dependencies can't be recovered if root expressions are considered as multi-valued.

We conclude that it is very important to keep the number of different but algebraically dependent kernels as small as possible, e.g., keeping track of common subexpressions.

The common origin of all these summands is also evident from *Cardano's formula* for the zeroes of the reduced cubic polynomial $x^3 + p\,x + q$

$$x = \sqrt[3]{-\frac{q}{2} + \sqrt{D}} + \sqrt[3]{-\frac{q}{2} - \sqrt{D}} \qquad \text{with } D = \left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3$$

The other systems take into consideration this rule. Consider, for example, Maple (version V.5):

```
s:=[solve(x^3+x+1,x)];
```

$$[-1/6\,\%1 + 2\,\%2,\ 1/12\,\%1 - \tfrac{1}{\%1} + 1/2\,I\sqrt{3}\,(-1/6\,\%1 - 2\,\%2),\ 1/12\,\%1 - \tfrac{1}{\%1} - 1/2\,I\sqrt{3}\,(-1/6\,\%1 - 2\,\%2)]$$

$$\%1 := \sqrt[3]{108 + 12\sqrt{93}}$$
$$\%2 := \frac{1}{\sqrt[3]{108 + 12\sqrt{93}}}$$

The result was formulated using new variables %1 and %2 that refer to a common subexpression (and its inverse) occuring in different places of the formula. Now we apply selectors and constructors of the CAS language to express the next step of our task:

```
product(x-op(i,s),i=1..3);
```

$$(x + 1/6\,\%1 - 2\,\%2)\,(x - 1/12\,\%1 + \%2 - 1/2\,I\,\sqrt{3}\,(-1/6\,\%1 - 2\,\%2))\,(x - 1/12\,\%1 + \%2 + 1/2\,I\,\sqrt{3}\,(-1/6\,\%1 - 2\,\%2))$$

```
p:=expand(%);
```

$$1/2 + x + x^3 + \frac{1}{18}\sqrt{93} - \frac{8}{108 + 12\sqrt{93}}$$

At this point we may remember that `expand` is a polynomial normal form operator of the outer simplification layer, not involving simplifications of algebraic numbers that are part of the middle simplification layer. The result is the same for an expression in independent variables x and $u = \%1$. We may prove this:

```
s1:=subs(%1=u,%2=1/u,s);
```

$$s1 := [-1/6\,u + 2\,u^{-1}, 1/12\,u - u^{-1} + 1/2\,I\sqrt{3}\left(-1/6\,u - 2\,u^{-1}\right), \\ 1/12\,u - u^{-1} - 1/2\,I\sqrt{3}\left(-1/6\,u - 2\,u^{-1}\right)]$$

```
expand(product(x-op(i,s1),i=1..3));
```

$$x + x^3 + \frac{1}{216}\,u^3 - 8\,u^{-3}$$

A subsequent simplification of nested roots finally involves the middle simplification layer and yields the original polynomial:

```
simplify(p);
```

$$x^3 + x + 1$$

For the sums of powers we get

```
a:=[seq(simplify(sum(s[i]^k,i=1 .. 3)),k=2 .. 9)];
```

$$[-2, -3, 2, 5, 6\,\tfrac{29+3\sqrt{3}\sqrt{31}}{\left(9+\sqrt{3}\sqrt{31}\right)^2}, -7, -36\,\tfrac{29+3\sqrt{3}\sqrt{31}}{\left(9+\sqrt{3}\sqrt{31}\right)^2}, 144\,\tfrac{135+14\sqrt{3}\sqrt{31}}{\left(9+\sqrt{3}\sqrt{31}\right)^3}]$$

By some reason not all expressions are fully simplified. Note that Maple, as most of the other CAS, doesn't incorporate the full power of algebraic simplifications into `simplify` but has a special algebraic evaluation operator `evala` that involves a stronger, but in some cases more tedious application of the knowledge about algebraic numbers.

```
map(evala,a);
```

```
[-2, -3, 2, 5, 1, -7, -6, 6]
```

In a similar fashion works MuPAD (version 1.3.1):

```
s:=solve(x^3+x+1,x);
s1:=_mult(x-op(s,i) $i=1..3);
expand(s1);
```

$$x^3 + 3x\sqrt[3]{\left(\frac{\sqrt{31}\sqrt{108}}{108} + 1/2\right)}\sqrt[3]{\left(\frac{\sqrt{31}\sqrt{108}}{108} - 1/2\right)} + 1$$

```
simplify(%);
```

$$x^3 + 3x\sqrt[3]{\left(\frac{\sqrt{93}}{18} + 1/2\right)}\sqrt[3]{\left(\frac{\sqrt{93}}{18} - 1/2\right)} + 1$$

One may wonder that the product of roots is not simplified. The reason is the same as explained above: If considered as independent, it is not clear which branches have to be combined to get the mathematically correct result. (Note that this changed in version 1.4 thus weakening mathematical exactness.) Probably, MuPAD implemented Cardano's formula $x = u + v$ with $u, v = \sqrt[3]{-\frac{q}{2} \pm \sqrt{D}}$ "as it is" and did not remember, that both roots u, v are related through the relation $u\,v = -\frac{p}{3}$.

For the power sum simplification this seems to be not important:

```
[radsimp(_plus(op(s,i)^k $ i=1 .. 3)) $ k=2 .. 9];
```

```
[-2, -3, 2, 5, 1, -7, -6, 6]
```

Axiom, Macsyma, Mathematica and Reduce solve this task in a manner similar to Maple that will not be discussed here.

5 Algebraic numbers not represented as root expressions

It is time to convince the students that there exist natural CAS contexts that (should) produce algebraic numbers not presented as root expressions. For this purpose we discuss (in the course, not in this paper; for a nice explanation see (Pieper, 1988)) how to solve polynomial equations of degree 3 theoretically. The *casus irreducibilis* that leads to 3 real roots deserves special attention, since these roots usually are represented in *trigonometric form*, and Reduce, MuPAD and Derive do so.

```
s:=solve(x^3-3*x+1,x);
```

$$\left\{2\cos\left(\tfrac{2\pi}{9}\right), -\cos\left(\tfrac{2\pi}{9}\right) + \sqrt{3}\sin\left(\tfrac{2\pi}{9}\right), -\cos\left(\tfrac{2\pi}{9}\right) - \sqrt{3}\sin\left(\tfrac{2\pi}{9}\right)\right\}$$

Needless to say, that all CAS under consideration (currently) don't simplify expressions containing such numbers automatically and hence the above task

fails on that example. (Nevertheless Maple and Mathematica are strong enough for a *guided* simplification of such expressions through conversion to exponential form and expansion, followed by algebraic simplification.) This is probably the main reason why Maple, Mathematica, Macsyma and Axiom use Cardano's formula also for negative discriminants.

The students' knowledge about the values of trigonometric functions at the special arguments $\frac{\pi}{3}$, $\frac{\pi}{4}$ and $\frac{\pi}{6}$ shows that special values of trigonometric functions may be of the same nature as root expressions. CAS usually know more such examples, e.g., root expressions for $\cos(\frac{\pi}{5})$ and $\sin(\frac{\pi}{5})$, but have difficulties with $\cos(\frac{\pi}{n})$ for $n > 6$. CAS are powerful enough to let us easily play around with such examples. So we can try to simplify expressions of the form

```
pi:=Pi;    # for Maple
u0:=cos(pi/5)*cos(2*pi/5);
u1:=cos(pi/7)*cos(2*pi/7)*cos(3*pi/7);
u2:=cos(pi/9)*cos(2*pi/9)*cos(4*pi/9);
```

u_0 simplifies with Maple and little effort to $\frac{1}{4}$. For u_1 and u_2 numeric approximations show that these numbers are very close to $\frac{1}{8}$ but it is impossible or at least hard (depending on the CAS) to prove that they are in fact equal to that number.

At this point the knowledge about trigonometric simplification rules, developed earlier in the course, may be applied to transform such expressions into polynomial expressions with a single kernel. This ends up with the equivalent problem (for $n = 7$ or $n = 9$) if $\cos(\frac{\pi}{n})$ is a root of a certain polynomial. If n is odd such a polynomial $p_n(y)$ may be obtained from the expansion of $\cos(n\,x)+1$ as a polynomial in $y = \cos(x)$ since for $x = \frac{\pi}{n}$ we get $\cos(n\,x) + 1 = 0$. Note that for odd primes n this polynomial factors as

$$p_n(y) = (y + 1)\, q_n(y)^2$$

for a certain irreducible polynomial $q_n(y)$. Hence $\cos(\frac{\pi}{n})$ is an algebraic number of degree $\frac{n-1}{2}$ for such n.

This yields the usual definition of an algebraic number as the root of a certain polynomial and also a good motivation to prove some properties of such a presentation. In particular we may easily extract the important additional information that such a description doesn't characterize the corresponding algebraic number uniquely. Indeed, the equation $p_n(y) = 0$ has obviously the zeroes $y_k = \cos(\frac{(2k+1)\pi}{n})$ for $k = 1, \ldots, \frac{n-1}{2}$.

6 Computing with algebraic numbers

With the definition given at the end of the last section we arrive at true symbolic objects that may represent several algebraic numbers (with the same defining polynomial). As long as we are not interested in the interaction between roots

of the same polynomial different algebraic numbers represented by the same symbol may be treated uniformly by the CAS simplificator attaching to the symbol the defining polynomial as an algebraic rewriting rule. Since the students learned earlier in the course how to work with symbols and rewriting rules we may do the corresponding calculations "by hand".

This allows to illustrate some usually difficult for beginners questions in the arithmetic of algebraic numbers. Consider the statement that the sum of algebraic numbers is again an algebraic number and the crude point in the proof to construct a corresponding defining polynomial. CAS based explanations may give much more evidence of the driving principles. Consider, e.g., the algebraic numbers $a = \sqrt{2}$ and $b = \sqrt[3]{5}$. For calculations "by hand" we will represent them as symbols a and b with rewriting rules $\{a^2 => 2, b^3 => 5\}$. Applying (with Reduce) these algebraic rewriting rules to the powers of their sum $c = a + b$

```
for k:=0:10 collect (a+b)^k where { a^2 => 2, b^3 => 5 };
```

we see that all such powers may be expressed as linear combinations of the 6 products $a^i b^j$ with $i = 0, 1,\ j = 0, 1, 2$. Hence we may expect the powers $c^i,\ i = 0, \ldots, 6$ to be linearly dependent. Such a dependency relation may easily be found from a generic polynomial $p(x)$ of degree 6 solving a system of linear equations. If one knows how to select the coefficients from a polynomial this yields 4 lines of code.

```
p:=x^6+for k:=0:5 sum mkid(c,k)*x^k;
p1:=(sub(x=a+b,p) where { a^2 => 2, b^3 => 5 } );
sys:=for each u in coeff(p1,a) join coeff(u,b);
sol:=solve(sys);
```

Hence

```
q:=sub(sol,p);
```

is a (possibly not yet irreducible) polynomial with root c. The main advantage of such an approach is that the students can (and must) concentrate on the top level algorithmic steps and are not burdened with tedious calculations.

The same applies to the computation of the defining polynomial of the product and also to the computation of the inverse of an algebraic number. Finally we arrive at the following constructive version of a well known proposition from algebra:

> *If α is an algebraic number of degree d over a field k then the set $R := k[\alpha]$ of k-linear combinations of terms from*
>
> $$T_{red} := \{\alpha^i,\ i = 0, \ldots, d-1\}$$
>
> *is a field.*
>
> *If k is effective then so is R. More precisely: If k has (canonical) normal forms and the minimal polynomial of α is known then R has (canonical) normal forms.*

CAS	Version	Commands

Macsyma	421	`algebraic:true;`
		`tellrat(a^5-a+1);`
		`rat(1/(1-a^2));`

Maple	V.5	`alias(a=RootOf(x^5-x+1));` `evala(1/(1-a^2));`
Mathematica	3.0	`a=Root[x^5-x+1,1];` `[ 1/(1-a^2); ]`

Reduce	3.6	`load arnum;`
		`defpoly a^5-a+1;`
		`1/(1-a^2);`

Table 2: User defined algebraic numbers

This proposition proves that the representation of an algebraic number as a symbol equipped with an algebraic rewriting rule derived from the corresponding defining polynomial is well suited for computationally efficient arithmetic operations. Hence CAS designers are well advised both to enable the user to introduce algebraic numbers in such a form and to detect and represent algebraic numbers produced by the system in even this form. Different CAS meet this requirement on different levels. The latter is commonly connected with the introduction of `RootOf` symbols, but, e.g., Macsyma (version 421) doesn't even apply the obvious rewriting rule to expressions containing such symbols. Note, on the other hand, the ubiquity of `RootOf` symbols even for algebraic numbers of degree 3 and 4 in the solution of systems of polynomial equations obtained with the corresponding `solve` function. As explained above a representation of the corresponding number as root expression may not only explode wrt. size but also wrt. computational complexity.

For the introduction of user defined algebraic numbers the CAS provide different mechanisms. In table 2 we collected the instructions necessary to introduce an algebraic number a with defining polynomial $p(x) = x^5 - x + 1$ and to simplify the expression $1/(1-a^2)$ yielding $a^3 + a$ for some of the CAS under consideration. Note that we found no way to tell Mathematica to rationalize the denominator. `Simplify` is to weak and `RootReduce` to strong.

To compute with user defined algebraic numbers in Axiom and MuPAD we need some more advanced knowledge. This is due to the strong type system (Axiom) resp. the object oriented domain concept (MuPAD). For example, in MuPAD we may define the corresponding field extension as a new domain of computation

```
Q:=Dom::AlgebraicExtension(Dom::Rational,a^5-a+1);
```

and then put `a:=Q(a)`, assigning to the variable `a` as value the algebraic number a obtained from the symbol `a` via the domain element constructor $Q(a)$. This

on a first glance difficult procedure is a good illustration of subtle notions in an object oriented approach but is beyond the scope of this paper. Finally

```
1/(a^2-1);
```

yields the desired result since the domain type of a forces the operations from Q to be called.

7 Computing in algebraic extension towers

The above proposition may be applied recursively to adjoin several algebraic numbers $\alpha_1, \alpha_2, \ldots, \alpha_n$ to a ground field k. Such a series of algebraic extensions $k_i = k_{i-1}[\alpha_i]$ with $k_0 = k$ is an *algebraic extension tower*. We can effectively compute in such towers if we can factor polynomials in $k_i[x]$ over all intermediate extensions k_i.

The latter is essential for computations with several algebraic numbers with the same (over the ground field) defining polynomial. If α_1 and α_2 are two algebraic numbers with the common (e.g., over $k = \mathbf{Q}$) defining polynomial $p(x)$, then α_2 is a root of $p(x)$ over $k_1 = k[\alpha_1]$ but

$$p(x) = (x - \alpha_1)\, q(x)$$

for a certain polynomial $q(x) \in k_1[x]$ and α_2 is a root of this second (not necessarily irreducible) factor. For example, if a is an algebraic number with defining polynomial $p(x) = x^5 - x + 1$ then $p(x)$ factors over $\mathbf{Q}[a]$ as

$$p(x) = (x - a)\,(a^4 + x^4 + ax^3 + a^3x + a^2x^2 - 1)$$

Such a factorization may be computed with a special version of the factorization command in Macsyma, Maple, Mathematica and MuPAD. Macsyma factors only in single algebraic extensions, the other three systems support also factorization in multiple extensions (at least in principle). Axiom offers the most advanced factorization tool but it is very difficult to transform the input parameters into the appropriate data types to get it work. The algebraic factorizer implemented in the `arnum` package of Reduce may be invoked only with the `factor` switch and not with the `factorize` command and doesn't cooperate well with the remaining part of the system.

We conclude that advanced computations with algebraic numbers exhaust the full power of today's general purpose Computer Algebra Systems.

References

Penrose, R. (1989). *The Emperor's New Mind.* Oxford University Press.

Pieper, H. (1988). *Die komplexen Zahlen.* Deutscher Verlag der Wissenschaften, Berlin.