Computer Algebra or Computer Mathematics?

James H. Davenport[a]

July 11, 2012

Department of Computer Science,
University of Bath, Bath BA2 7AY, England
J.H.Davenport@bath.ac.uk

[a]The author was partially supported by the European OpenMath Thematic Network.

Scope

- "polynomial-type" systems: Axiom, Macsyma/Maxima, Maple Mathematica and Reduce.
- We quote Maple, but do not intend to condemn it.
- I do not ignore systems such as GAP, KANT, PARI and Magma: rather the thesis of this talk is (largely) irrelevant to them.

Strengths

- These algebra systems perform massive computations (Delaunay's 20-year computation of the orbit of the moon can now be done in under a second)
- They incorporate extremely sophisticated algorithms: integration, ordinary differential equations and Gröbner bases
- G.H. Hardy on integration: "There is no known process [algorithm], and there is reason to believe that no such can be given".

  Solved (Risch) since indefinite integration can be viewed as an algebraic processs, essentially anti-differentiation.

But these systems do not please the users

- Some of this is because users are never pleased;
- Some of this is because users have unrealistic expectations.
- but some of it is because there is a mismatch between what these systems do, and what users think they do. In particular, the users think that the systems are doing mathematics, whereas typically the systems are only doing that part of the mathematics that is algebra.

A little history

- The scope of the computer algebra systems of the 1960s was unashamedly limited to algebra: they did the tedious algebra, but it was the task of the user, generally an expert in the relevant mathematics, to see that the algebra made sense.
- As computers spread, and particularly with the advent of personal computers, these systems became available to a much wider audience, and the assumption that they were merely "algebra engines" and all the mathematical knowledge belonged to the user, had to be challenged.
- Mathematica is now explicitly sold as a "computer mathematics system". Do we know what "computer mathematics" is?
- Does "computer mathematics" make sense as one subject, or is "computer $\mathbf{R}$" different from "computer $\mathbf{C}$"?

Algebraic Numbers and their Fields

- The common way of constructing $\mathbf{Q}(\sqrt{2})$ algebraically is as $K := Q[\alpha]/(\alpha^2 - 2)$: the quotient of a polynomial ring by a principal ideal.
- In K, just as in $\mathbf{R}$, the equation $x^2 = 2$ has two roots, α and $-\alpha$. However, nothing says whether α corresponds to $\sqrt{2} = 1.4142$ or $-\sqrt{2} = -1.4142$.
- K as we have defined it is not an *ordered field*.
- How does K embed into $\mathbf{R}$?

For more general algebraic numbers, this information is no longer implicit in the definition of the roots of a polynomial. For example, the polynomial $f(x) := x^4 - 10x^2 + 1$ has four real roots, and we might have to code one possible embedding of $\mathbf{Q}$ extended by a root of this polynomial as

$$\mathbf{Q}[\alpha]/(\alpha^4 - 10\alpha^2 + 1) \wedge \alpha \in [3, 4].$$

The other possible embeddings have $\alpha \in [0, 1]$, $\alpha \in [-1, 0]$ and $\alpha \in [-4, -3]$. Once we have this interval information, a bisection process can refine the interval sufficiently that any two different elements of $\mathbf{Q}[\alpha]/(\alpha^4 - 10\alpha^2 + 1)$ can be compared.

There are several other approaches to distinguishing the real roots of a polynomial, e.g. by Thom's Lemma, but we will not compare these in detail. In these approaches, the abstract algebraic extension, as an unordered field, is modeled as $\mathbf{Q}[\alpha]/(f(\alpha))$, with algebra and equality testing done in this algebraic domain, and the choice of "which root" is only important when the ordering properties of the field are invoked. Nevertheless, a purely algebraic approach has to be blended with some numeric information in order to model the user's mental image of "*this* root of $f(x) := x^4 - 10x^2 + 1$".

The Numeric Approach

An alternative approach is to model the field as a subset of $\mathbf{R}$, with α being represented by a "sufficiently accurate" numerical approximation. Possible models include the use of continued fractions and B-adic approximations for various bases B. This seems to model the ordered field structure correctly, and in one sense it does. However, in computer algebra, we also take for granted the notion of equality, and that is what this approach does not do. If we try to compare $\sqrt{2}^2$ with 2, we will find that, no matter how much precision we call for, the answer is always "uncertain".

The solution to this problem seems to be that an element of $\mathbf{Q}(\alpha) \subset \mathbf{R}$ must be modeled with both its numerical properties and its algebraic properties. One way of doing this is to combine a numerical approximation with a minimal, or at least defining, polynomial, so an algebraic α is represented as $\langle \overline{\alpha}, f(\alpha) \rangle$, where $\overline{\alpha}$ is the approximation-producing equivalent of α. In the example in the previous paragraph, $\sqrt{2}$ would be represented by $\langle \overline{\sqrt{2}}, x^2 - 2 \rangle$, $\sqrt{2}^2$ by $\langle \overline{\sqrt{2}^2}, x^4 - 8x^2 + 16 \rangle$, and $\sqrt{2}^2 - 2$ by $\langle \overline{\sqrt{2}^2 - 2}, x^4 + 8x^3 + 16x^2 \rangle$. This last polynomial admits $x = 0$ as a root, and numerical evaluation of $\overline{\sqrt{2}^2 - 2}$, combined with Mahler's lower bound on non-zero roots of a polynomial will show that this *has* to be zero.

However we do it, we have to blend the numeric approach with some algebraic information in order to model the user's mental image of "this number $3.1462\ldots$ *is* also a root of $f(x) := x^4 - 10x^2 + 1$".

How to Model a Number Field

While we mention minimal polynomials above, this is in practice a very inefficient means of computing, and one should certainly compute in a tower of extensions. We should note this as a typical example of mathematical cleanliness ("without loss of generality, we may assume that $\alpha_1, \ldots, \alpha_k$ all lie in a given field $\mathbf{Q}(\beta)$") versus computational efficiency. There are several reasons for the practical use of towers.

1. Primitive elements tend to introduce a lack of sparsity. For example, the field $\mathbf{Q}(\sqrt{2}, \sqrt{3}, \sqrt{5}, \sqrt{7})$ has a primitive element (viz. $\beta := \sqrt{2} + \sqrt{3} + \sqrt{5} + \sqrt{7}$) whose minimal polynomial is $\beta^{16} - 136\beta^{14} + \cdots - 5596840\beta^2 + 46225$.

2. As one can see from the example above, they also tend to induce coefficient growth.

3. Primitive elements tend to place one in the "most general setting": one could be subtracting $\sqrt{2}$ from itself, but because one had mentioned $\sqrt{3}$, one was dealing in a more complex world, where the generator was $\alpha : \alpha^4 - 10\alpha^2 + 1 = 0$, and to check that $\sqrt{2} - \sqrt{2} = 0$, one has to satisfy oneself that one has the root $\beta = 0$ of $\beta^{16} - 32 * \beta^{14} + \cdots + 4096\beta^8$ rather than of $\beta^4 - 8\beta^2$.

4. From the point of view of programming a computer algebra system, the field is not generally given in advance: the user can

introduce a new algebraic element, i.e. grow the field of definition, at any time. This requires an elaborate data structure to convert elements on-the-fly from the old presentation to the new one, even though that conversion may not be necessary.

Local or global towers

Line	Code	Tower
1	`a:=sqrt(2)`	$\sqrt{2}$
2	`b:=a+sqrt(3)`	$\sqrt{2}, \sqrt{3}$
3	`c:=a+sqrt(5)`	`???`

If `???` is $\sqrt{2}, \sqrt{3}, \sqrt{5}$, then we have adopted the "global tower" approach, using the last tower even though $\sqrt{3}$ is irrelevant to `c`. We are then working in a larger tower than is necessary. For example, if line 2 was a typographical error, and line 3 were `b:=a+sqrt(5)`, from then on we would be working in a field of twice the degree necessary — an expensive price to pay for a simple typing error.

Local Towers

If `???` is $\sqrt{2}, \sqrt{5}$, then we have adopted the "local tower" approach, using the tower of the input (merger of the towers of the inputs, in general) to build on. This leads to much smaller towers, and avoids the "typing error penalty" of the other approach. However, the operation of merging towers can prove interesting.

Line	Code	Tower
1	`a:=sqrt(2)`	$\sqrt{2}$
2	`b:=a+sqrt(3)`	$\sqrt{2}, \sqrt{3}$
3	`c:=a+sqrt(6)`	$\sqrt{2}, \sqrt{6}$
4	`d:=b+c`	`merge(`$\sqrt{2}, \sqrt{3}$`,`$\sqrt{2}, \sqrt{6}$`)`

It is clear that, *algebraically*, the merged tower is $\sqrt{2}, \sqrt{3}$ (which is isomorphic to $\sqrt{2}, \sqrt{6}$), but we have no idea whether $\sqrt{6}$, in this tower, is $\sqrt{2}\sqrt{3}$ or $-\sqrt{2}\sqrt{3}$.

In line with our thesis, that one has to know the embedding into $\mathbf{R}$ as well as the algebraic information, the code fragment should be written as below, where `d` becomes $2\sqrt{2}+\sqrt{3}+\sqrt{2}\sqrt{3}$.

Line	Code	Tower
1	`a:=[sqrt(2),[1,2]]`	$\sqrt{2} \in [1,2]$
2	`b:=a+[sqrt(3),[1,2]]`	$\sqrt{2} \in [1,2], \sqrt{3} \in [1,2]$
3	`c:=a+[sqrt(6),[2,3]]`	$\sqrt{2} \in [1,2], \sqrt{6} \in [2,3]$
4	`d:=b+c`	$\sqrt{2} \in [1,2], \sqrt{3} \in [1,2]$

It should be noted that it is also possible to have a "lazy" tower merging process, in which one can build an unreduced tower.

Line	Code	Tower
1	`a:=[sqrt(2),[1,2]]`	$\sqrt{2} \in [1,2]$
2	`b:=a+[sqrt(3),[1,2]]`	$\sqrt{2} \in [1,2], \sqrt{3} \in [1,2]$
3	`c:=a+[sqrt(6),[2,3]]`	$\sqrt{2} \in [1,2], \sqrt{6} \in [2,3]$
4	`d:=b+c`	$\sqrt{2} \in [1,2], \sqrt{3} \in [1,2], \sqrt{6} \in [2,3]$
5	`if c-a=a*(b-a) ...`	$\sqrt{2} \in [1,2], \sqrt{3} \in [1,2], \sqrt{6} \in [2,3]$

In this case, `d` becomes $2\sqrt{2}+\sqrt{3}+\sqrt{6}$, and it is only at line 5 that we discover that $\sqrt{6} = \sqrt{2}\sqrt{3}$.

- In the algebraic approach, testing whether a number is zero is tantamount to testing whether it is a zero divisor, and, appliying the extended Euclidean algorithm to inverting $\gamma - \alpha\beta$ subject to $\gamma^2 - 6$, $\beta^2 - 3$ and $\alpha^2 - 2$, we soon find that $\gamma\beta\alpha - 6$ is a zero-divisor, with cofactor $\gamma\beta\alpha + 6$. Which of these is "right" depends on the numeric information for α, β and γ.
- In the numerical approach, since numerical computation with the approximations to `c-a` and `a*(b-a)` does not decide the equality after a suitable point, we switch to a symbolic test of equality via Mahler's inequality. At this point, we can change the minimal polynomial for $\sqrt{6}$, from $z^2 - 6$ to $z - \sqrt{2}\sqrt{3}$. Note that, in this case, had we been asked the same question, but with $\sqrt{6} \in [-3, -2]$, we would not have bothered to construct the (reducible) defining polynomial (which is of course the same), since numerical tests would have shown us that $\sqrt{6} - \sqrt{2}\sqrt{3} \in [-3, -7]$, and is therefore nonzero.

The principal advantage of the "lazy tower" method is that one avoids all factorisation of polynomials over algebraic number fields. Instead, a factorisation is only detected when it is thrust in our face by the user's computations. Although theoretically in polynomial time, factorisation of polynomials over algebraic number fields is very expensive. The drawback is that, if, say, one starts with $\mathbf{Q}(\sqrt{2}, \sqrt{8})$, then, until the redundancy is detected, one is working in extensions of twice the necessary degree. We should note that there is no need to "split" and pursue multiple computations, since the numeric information tells us which branch to pursue.

Moral. The moral of this section is two-fold. The first is that it is necessary, to capture the users' requirements, to model algebraic numbers both as elements of an abstract algebraic extension, and with an embedding into $\mathbf{R}$ (or $\mathbf{C}$). Secondarily, we have learnt that the simplest situation for abstract mathematics (a primitive element) is neither efficient nor suitable for practical computation.

Elementary Transcendental Functions

These functions are normally considered to be exp and its inverse ln, the six trigonometric functions and their inverses, and the six hyperbolic functions and their inverses. For the purposes of this paper, we will class exp, the six trigonometric functions and the six hyperbolic functions together as the *forward functions*, and the remainder as the *inverse functions*. The forward functions are relatively straight-forward: they are many–one, continuous functions $\mathbf{C} \to \mathbf{C}$ (and restrict to similar functions $\mathbf{R} \to \mathbf{R}$). These functions, or at least most of them, are built into computer algebra systems, as well as the numeric languages for which computer algebra systems often generate code, and are used in a variety of ways.

Calculus We expect to integrate, differentiate, and solve differential equations with expressions involving these functions. In particular, we expect $\int \frac{1}{x}\,\mathrm{d}x$ to return $\ln x$ (or possibly $\ln|x|$).

"Simplification" We expect to see expressions involving the elementary functions "simplified" — whatever that might mean, and simplification is, in general, in the eye of the beholder. For example, we might expect $\exp(a)\exp(b) \to \exp(a+b)$, and many people would expect its converse $\ln(a) + \ln(b) \to \ln(ab)$ to happen, even though the branch cuts mean that this is not true over $\mathbf{C}$.

Symbolic Evaluation We expect to see $\ln 1 \to 0$ and $\sin\frac{\pi}{2} \to 1$. Whether we expect to see $\tan\frac{\pi}{2} \to \infty$ or $+\infty$ is a moot point.

Numeric Evaluation We expect to be able to evaluate these functions at floating-point values (in $\mathbf{R}$ or $\mathbf{C}$) to yield

floating-point results.

Furthermore, the user wishes to perform any or all of these operation on an expression built from elementary functions, and know that he is getting consistent results.

What Need a System Know

Indefinite Calculus By this phrase, we mean differentiation, indefinite integration and indefinite solution of ordinary differential equations. Here all that is required is a knowledge of the differential-algebraic properties of the functions, e.g. $(\exp f(x))' = f'(x)\,(\exp f(x))$ or $(\ln f(x))' = \frac{f'(x)}{f(x)}$. From this point of view, $\int \exp x = \exp x$, and whether $\exp(x)$ is e^x or $2e^x$ is irrelevant, and where the branch cuts of any actual function $\ln : \mathbf{C} \to \mathbf{C}$ lie even more so. Indeed, in differential algebra, there is no concept of "evaluating" x at all: x is merely the base symbol whose derivative is 1.

Definite Calculus By this phrase, we mean definite integration and definite solution of ordinary differential equations. Sometimes these are solved numerically, in which case we are really back to numeric evaluation, but more often they are solved by indefinite methods followed by instantiation. At this

stage the actual meaning of the functions, as $\mathbf{R} \to \mathbf{R}$ (or possibly $\mathbf{C} \to \mathbf{C}$) matters: after all $\int_0^1 e^x \, \mathrm{d}x$ is suddenly very different from $\int_0^1 2e^x \, \mathrm{d}x$. Furthermore branch cuts in the integrand and the (indefinite) integral become very important. In certain cases the Lazard/Rioboo/Trager formulation of the integral can help, but in general there is no substitute for a complete analysis of these branch cuts.

A simple example is given by the following integral in the complex plane: $\int_P^{\overline{P}} \frac{1}{z}\,\mathrm{d}z$, where P is the point on the unit circle $\frac{-1+i}{\sqrt{2}}$. An indefinite integration followed by substitution gives $\ln \overline{P} - \ln P$, which is indeed what Maple 8 returns. A naïve evaluation of this would give $\frac{-3}{4}\pi i - \frac{3}{4}\pi i = -\frac{3}{2}\pi i$. However, standard complex variable theory tells us that integrating along the straight line between P and $\overline{P}$ is the same as integrating along the arc, when we are integrating something of absolue value 1 along an arc of length $\frac{\pi}{2}$, so the answer is clearly $\frac{\pi}{2}i$. So we have two answers: $-\frac{3}{2}\pi i$ and $\frac{\pi}{2}i$. The second is correct, the first ignored the fact that the standard branch cut for logarithm, the negative real axis, intersects the path of integration, so that our integral, as expressed by the standard form of ln, is not continuous. The resolution is to choose a different branch cut, and therefore, if $\ln P = \frac{3}{4}\pi i$, $\ln \overline{P}$ has to be $\frac{5}{4}\pi i$, and the correct solution is restored.

Simplification/Symbolic Evaluation Purely algebraic simplification, as in Maple's `simplify(...,symbolic)`, can be achieved from the differential-algebraic definitions and appropriate values of constants. For example, consider $\ln f + \ln g = \ln fg$. Differentiating this equation gives $\frac{f'}{f} + \frac{g'}{g}$ on the left-hand side, and $\frac{(fg)'}{fg}$ on the right-hand side, and these are clearly equal. Hence we can deduce that $\ln f + \ln g = c + \ln fg$ for some constant c. Unfortunately, the meaning of the word "constant" here is "something that differentiates to zero" rather than "something that takes on the same value as the variables range through $\mathbf{C}$". The value of this "constant" is given by equation (3), and depends on the imaginary parts of $\ln f$ and $\ln g$.

However, this sort of simplification says very little about the actual simplification rules as applied to functions $\mathbf{C} \to \mathbf{C}$ (or $\mathbf{R} \to \mathbf{R}$). There is a close relation between precisely what definitions are adopted for the elementary functions, and precisely what simplifications are valid.

- With the standard definitions of

$$\arctan(z) = \frac{1}{2i}\left(\ln(1+iz) - \ln(1-iz)\right), \tag{1}$$

  and, as functions $\mathbf{C} \to \mathbf{C}$, $\arcsin(z) \neq \arctan \frac{z}{\sqrt{1-z^2}}$, since they differ on the branch cuts $(-\infty, -1) \cup (1, \infty)$.

- Conversely, for Derive's definition of arctan, $\arcsin(z) = \underbrace{\arctan}_{\text{Derive}} \frac{z}{\sqrt{1-z^2}}$, and $\underbrace{\arctan}_{\text{Derive}}(z) \neq \frac{1}{2i}\left(\ln(1+iz) - \ln(1-iz)\right)$ since they differ on the branch cuts. What is true is that

$\underbrace{\arctan}_{\text{Derive}}(z) = \overline{\frac{1}{2i}\left(\ln(1+i\overline{z}) - \ln(1-i\overline{z})\right)}$.

- Of course, arctan and $\underbrace{\arctan}_{\text{Derive}}$ agree on $[-1,1]$, i.e. as (partial) functions $\mathbf{R} \to \mathbf{R}$.

Numeric Evaluation Here again, the numeric value is critically dependent on the branch cuts chosen. A further problem, intrinsic to floating-point, is that numeric evaluation near branch cuts is inevitably unstable: a minor change in real or imaginary part can push the problem the wrong side of the cut, and give completely the "wrong" answer. The risk is reduced for elementary functions, since the branch cuts for these lie along the axes (and therefore the "signed zeros" approach works for these branch cuts), but not eliminated.

Hence computer algebra systems need to know:

1. differential-algebraic information;
2. abstract simplification information (including symbolic evaluation);
3. branch cut information;
4. numerical evaluation information.

In theory, all four should be consistent, but a look at the last two lines of the next table shows that it is perfectly possible for them not to be.

Item (1) is easily encoded inside the calculus packages or equivalent, and similarly item (4) is easily encoded in the numerical evaluation package.

Different values of $\operatorname{arccot}(-1)$

Source	Detail	$\operatorname{arccot}(-1)$	Comments
AS	1st printing	$3\pi/4$	inconsistent
AS	9th printing	$-\pi/4$	
GR	5th edition	?	inconsistent
CRC	30th edition	$3\pi/4$	inconsistent
Maple	V release 5	$3\pi/4$	
Axiom	2.1	$3\pi/4$	
Mathematica		$-\pi/4$	
Maxima	5.5	$-\pi/4$	
Reduce	3.4.1	$-\pi/4$	in floating point
Matlab	5.3.0	$-\pi/4$	in floating point
Matlab	5.3.0	$3\pi/4$	symbolic toolbox

The note “inconsistent” means that, although the source quotes, or clearly lets be inferred, a value for $\operatorname{arccot}(-1)$, there are enough inconsistencies in the definition of arccot that one could infer a different value. For GR, $3\pi/4$ and $-\pi/4$ are equally inferrable.

Items (2–3) are more complicated. It is not even clear what would be meant by a formal encoding of "branch cut information" — however, it should include both the location of the branch cut and the values that the function takes on the branch cut. For ln, AS resorts to a mixture of words (for the former) and the inequality $-\pi < \Im \ln z \leq \pi$ (for the latter). For elementary functions, two solutions have recently been proposed.

- CJ introduces the *unwinding number* $\mathcal{K}$, defined by

$$\mathcal{K}(z) = \frac{z - \ln \exp z}{2\pi i} = \left\lceil \frac{\Im z - \pi}{2\pi} \right\rceil \in \mathbf{Z}. \tag{2}$$

The branch cut information associated with each simplification can then be encoded in terms of additional unwinding number terms, as in:

$$\ln(z_1 z_2) = \ln z_1 + \ln z_2 - 2\pi i \mathcal{K}(\ln z_1 + \ln z_2); \tag{3}$$

$$\arcsin z = \arctan \frac{z}{\sqrt{1-z^2}} + \pi\mathcal{K}(-\ln(1+z)) - \pi\mathcal{K}(-\ln(1-z)). \tag{4}$$

One drawback of this is that the notation is unfamiliar to most users, and is not in calculus texts, so a system which used it internally would probably have to convert it into a more user-friendly form on output — quite a challenge in practice. One advantage that such a system might have is that it restricts comparatively easily to $\mathbf{R}$. For example, in equation(3), if $\ln z_1$ and $\ln z_2$ are real, then $\mathcal{K}(\ln z_1 + \ln z_2)$ is automatically zero. It is also possible (at least for a human) to tell from the form of the $\mathcal{K}$ terms whether they are ruling out the simple identity for a whole region (as in equation (3)), or just specific branch cuts (as in equation (4)).

- CDJW proposes to encode all elementary functions in terms of exp and ln. If we treat the branch cut for ln as a built-in primitive, then all other elementary branch cuts can be expressed in terms of this one. One major drawback to this scheme by itself is that the simplifier would then have to do a lot of reasoning to derive, and ensure the appropriate correction terms in, simplification rules. While it is possible to derive equations such as (4) quasi-automatically, the process is complicated and can only be guaranteed to work in a restricted set of cases.

  However, we could still take this approach as a base, in that the primitive would indeed be the branch cut for ln, and any added rules, whether encoded via $\mathcal{K}$ or in other ways, would be verifiable (semi-manually?) to be consistent with this primitive.

Moral. The moral of this section is that a full interpretation of a function such as arctan involves several different kinds of

information, differential-algebraic, simplification, branch cuts and numeric, which have to be present consistently in a system for it to be able to model a user's full range of expectations about that function. It should also be noted that, while "abstract" differential-algebraic simplification can be achieved, in terms of deciding whether two expressions are "equal up to constants", the problem for actual functions $\mathbf{R} \to \mathbf{R}$ or $\mathbf{C} \to \mathbf{C}$ is much harder and in general undecidable.

R or **C**?

Aslaksen asks: "Can your computer do complex analysis?", to which the answer generally is "not very well". An equally relevant question would be "Can your computer do real analysis, or does it insist on (trying to do) complex analysis?". In this section, we explore the difference.

The issue

We have already seen that $\ln(a) + \ln(b) \to \ln(ab)$ is true over $\mathbf{R}$ (by which we mean that not only $a, b \in \mathbf{R}$ but also $\ln(a), \ln(b) \in \mathbf{R}$) but not over $\mathbf{C}$. In general, most computer algebra systems:

- are unclear[a] (certainly to the user, and all too often in the minds of the authors) whether they are working in $\mathbf{R}$ or $\mathbf{C}$;
- provide few or no means for controlling this behaviour. The Maple `assume` facility, while very powerful, does not really control this — it can control, *inter alia*, whether particular variables are to be interpreted in $\mathbf{R}$ or $\mathbf{C}$, as in `assume(a,real)`, but this is insufficient, since $\ln(a)$ might still not be real (indeed $a = b = -1$ is a counter-example to

[a] An honest assessment is given in the Maxima 5.5 documentation: "Variable: **DOMAIN**. Default: [REAL] — if set to COMPLEX, SQRT(X2̂) will remain SQRT(X2̂) instead of returning ABS(X). The notion of a "domain" of simplification is still in its infancy, and controls little more than this at the moment."

$\ln(a) + \ln(b) \to \ln(ab)$). One would need to add that a and b were positive. In a more complicated case, it might be very difficult to work out the assumptions on the input variables that would guarantee that all the intermediate functions were real, and there is still no guarantee that Maple would make those deductions as well.

A common argument is that, since **C** is closed under the elementary functions, it is the "natural" domain, and that to work in **R** would leave systems open to variants on the

$$1 = \sqrt{1} = \sqrt{(-1)\cdot(1)} = \sqrt{-1}\cdot\sqrt{-1} = \sqrt{-1}^2 = -1$$

paradox. This is a reasonable argument for making **C** the default, which we do not challenge, but seems like a poor argument for making the semantics of **R** inaccessible, or at least very hard to access.

But $\mathbf{R} \subset \mathbf{C}$

It is often assumed that, since $\mathbf{R} \subset \mathbf{C}$, this doesn't really matter, since everything that is true in $\mathbf{C}$ will still be true in $\mathbf{R}$. For identities, this is indeed true: since $\exp(a)\exp(b) \to \exp(a+b)$ is true over $\mathbf{C}$, it is true over $\mathbf{R}$. However, this piece of glib reasoning misses the following points.

1. It may give unexpected results to the user: having seen Maple, say, simplify $\ln 2 + \ln 3 \to \ln 6$ (which Maple can do since it knows that the lns involved are positive, so the branch cuts don't interfere), the user is frustrated by the fact that $\ln(a) + \ln(b) \not\to \ln(ab)$.
2. Although $\mathbf{R} \subset \mathbf{C}$, the same is not true of their topological completions: $\mathbf{R} \cup \{-\infty, +\infty\} \not\subset \mathbf{C} \cup \{\infty\}$. For example, over $\mathbf{R}$ one can define tan to be a bijection between $[-\frac{\pi}{2}, \frac{\pi}{2}]$ and $[-\infty, +\infty]$, but over $\mathbf{C}$, $\tan(-\frac{\pi}{2}) = \tan(\frac{\pi}{2}) = \infty$.

3. The "natural" values of some intrinsically multivalued functions may be different. For example, over $\mathbf{C}$ it is natural to choose $\sqrt[3]{-1}$ to be $\frac{1-i\sqrt{3}}{2} = \exp(\frac{1}{3}\ln(-1))$, but over $\mathbf{R}$ it is natural to choose it to be -1. Now, Maple does provide the function `surd`: $\mathbf{R} \times \mathbf{N} \to \mathbf{R} \cup$"square root of negative" to solve this problem, but nevertheless the issue crops up about once a month in the Maple news group.

4. Any statement that requires an existence argument, e.g. a proof that $f \neq g$ via Maple's `testeq` [17], which relies on finding a z such that $f(z) \neq g(z)$, may not be valid over $\mathbf{R}$, since the necessary z may not be in $\mathbf{R}$. For example, $\arcsin(z) \neq \arctan \frac{z}{\sqrt{1-z^2}}$ as functions $\mathbf{C} \to \mathbf{C}$, since, for example, $\arcsin(2) \approx \frac{\pi}{2} - 1.317i$, whereas $\arctan \frac{2}{\sqrt{-3}} \approx -\frac{\pi}{2} - 1.317i$. However, as (partial) functions $\mathbf{R} \to \mathbf{R}$, they do agree, since all the counter-examples, although real values of z, involve complex values of $\arcsin z$.

The Users' Reality

It is the author's experience, both first-hand and from reading the Maple user group and news group, that the majority of users (if they are concerned at all about the field of definition) are concerned with **R** rather than **C**. The conclusion from Maple users could be considered biased, since those who are happy with **C**, the semantics of Maple, may well not comment; however the conclusion from the author's own experience is probably free from such biases. In particular, when working with the GENTRAN FORTRAN generation package in Reduce, it was obvious (and conversations with the author confirmed this) that the complex part of this was much less heavily exercised than the real.

The reason for this is obvious. The vast majority of physical equations involve real variables. Indeed many physical quantities (masses, resistances, concentrations etc.) are constrained to be non-negative, and the Maple user group is often advising people on

how to ensure this.

Those users who do venture into complex variables, e.g. for the conformal mapping solution of 2-dimensional fluid dynamics problems are often unaware of the pitfalls of complex simplification, and wish to use the rules inherited from the reals. Algebra systems often stop them, but do not explain why they are doing so.

The following excerpt from a posting by Fateman in the Maple news group in 1993 expresses the view that the majority of users of the "polynomial-type" computer algebra systems today are hoping that they are using "computer mathematics" systems.

> I believe that for many readers of this newsgroup, those who are interested in the problems of general algorithmic manipulation of mathematical objects in CAS, the point is really this:
>
> You cannot tell the sign of RootOf(x^ 2-1) (which could be either +1 or -1). As a consequence, an enormous range of computations that you could do with the specific number 1 or the specific number -1 are excluded.
>
> When an algebraist claims "any root will do" he/she has focussed on a set of operations that represent only a portion of what a CAS can do. What is sin(rootof(x^ 2-1))? The algebraist might say, oh, I didn't mean you could do THAT!
>
> While I think I have some appreciation for the importance

of algebra as the basis for the CAS of today, most people "out there" are probably NOT asking algebraic questions. Most people who wish to find the roots of a univariate polynomial are generally NOT seeking algebraic answers. They are seeking a set of complex floating-point numbers. If they are offered algebraic factoring programs instead of root-finding programs, they will be wasting time, and will be unhappy with the results.

Of course, some people ARE interested in the algebraic questions, and it is nice to be able to answer them, too. However, it may be a good idea to address those issues in a different fashion. Some people advocate building a separate CAS in the service of algebraists, number theorists, etc. Although I think it is good to have algebraic facilities available in a general system – to use them when needed, I think it is a mistake to view algebraic answers as "the most correct" or when they are useless for the purposes intended for the answers.

Conclusions

We can make various deductions about the inadequacy of current computer algebra systems in meeting users' requirements for computer mathematics.

- Even in the case of algebraic numbers, $\alpha \mid \alpha^2 = 2$ does not fully capture the user's intuitive $\sqrt{2}$. One needs some way of combining the algebraic information with numerical information $\alpha \approx 1.4142\ldots$. We point out a couple of solutions which have been discussed, but neither have made their way into main-stream computer algebra systems yet. Specialised software, e.g. to do Cylindrical Algebraic Decomposition has incorporated these ideas for years, typically in the $\alpha \mid \alpha^2 = 2 \wedge \alpha \in [1, 2]$ or "$\alpha \mid \alpha^2 = 2$ and α is the second root (from $-\infty$) of this equation" formulations.
- Elementary functions have many uses in mathematics, and

many ways to manipulate them in algebra systems. Systems are not necessarily consistent (see Appendix) in this, and the information required to define both the algebraic and analytic behaviours of these functions, in a consistent way, is not well-defined.

- Systems are much better at not applying a simplification (for good reason) than they are in explaining *why* it was not applied.
- Computer algebra systems have not come to terms with the **R**/**C** dichotomy.

Questions for Mathematicians

1. Is it possible to produce a theory of the elementary functions *as they are in computation*: single-valued functions $\mathbf{C} \setminus \{\text{singularities}\} \to \mathbf{C}$? The following standard responses are not necessarily adequate, for the reasons given.

   - Many analysts would urge one to "consider the Riemann surface". Unfortunately, in the case of $\arctan x + \arctan y \stackrel{?}{=} \arctan\left(\frac{x+y}{1-xy}\right)$, $\arctan x$ is defined on one surface, $\arctan y$ on a second, and $\arctan\left(\frac{x+y}{1-xy}\right)$ on a third. It is not clear how "considering the Riemann surface" solves practical questions such as this identity.
   - In a similar vein, one is urged to consider multivalued definitions (denoted Ln etc.) for the inverses of exp etc. Simple examples with Ln and Arctan look fairly convincing. However, there are difficulties in practice, e.g.

$\mathrm{Arcsin}(x) - \mathrm{Arcsin}(x) =$

$\{2n\pi \mid n \in \mathbf{Z}\} \cup \{2\arcsin(x) - \pi + 2n\pi \mid n \in \mathbf{Z}\} \cup \{\pi - 2\arcsin(x) + 2n\pi \mid n \in \mathbf{Z}\},$

which depends on x, so the algebra of multivalued functions is non-trivial. Several useful identities also become more difficult, e.g. $\arcsin z \stackrel{?}{=} \arctan \frac{z}{\sqrt{1-z^2}}$ (true except on the branch cuts) becomes the more complicated

$$\mathrm{Arcsin}(z) \cup \mathrm{Arcsin}(-z) = \mathrm{Arctan}\left(\frac{z}{\mathrm{Sqrt}(1-z^2)}\right).$$

2. Is it possible to produce a "calculus of branches"? This might be more promising, since it might extend to non-elementary functions, for which there may be no simple explanation of the branch structure in the form $+2n\pi$, as in the case of the Lambert W function.

Questions for Computer Algebra System Designers

1. Should there be (and we have argued for) an explicit choice between the semantics of **R** and **C**? This could either be global, e.g. a massive reworking of Maxima's `DOMAIN` switch, or local, e.g. a command such as `realsimplify`.
2. Is it possible to explain to users why simplifications *don't* happen?
3. Could some interactivity be built in? This is trickier than it looks: Macsyma used to be notorious for asking questions of the form "is `very large expression` positive, negative or zero?".
4. Below we observe that a command like `LOGCONTRACT` does not keep any record of what is being assumed in the process, e.g. $\ln a + \ln b \to \ln(ab)$ assumes that $-\pi < \Im(\ln a + \ln b) \leq \pi$. Maybe such a record should be kept, and either displayed to the user, or maybe even fed to some kind of verifier. Such

verification would need to take account of any `assume` facility [37]. Also, the verifier would need to have the same meanings of the elementary functions as the algebra system: not entirely obvious in view of the table in the appendix.

5. Generalising the above idea, is the manipulation of these functions best left to an algebra system, or is interaction between an algebra system and theorem provers required?

Questions for User Education

1. It is quite clear from the questions asked in, e.g., the Maple news group, that many users are ignorant about, or puzzled by aspects of, the analysis of **C**. We have argued above that some of these problems could be remedied by explicitly offering **R** instead. However, this is not a complete solution, and one has to ask how one explains the issues to the user, who is currently faced with a choice between:

   - A refusal by the system to "do the right thing", e.g. $\ln a + \ln b \not\to \ln(ab)$, with no explanation attached;
   - A command like `LOGCONTRACT`, which will do what the user believes to be the right thing, but with no explanation of what the user is assuming in the process — the verification community would think of these as the user's proof obligations.

2. How does an algebra system explain to the user "valid except on these branch cuts" or "valid outside this region"? We have already said that an explanation in terms of unwinding numbers is unlikely to make sense to most users in the current state of complex variable education. In one complex variable, one can imagine the system drawing a diagram (though issues of scale would be interesting), but in more than one variable it is hard to see how to do it.

References

[1] Abbott,J.A., Bradford,R.J. & Davenport,J.H., Factorisation of Polynomials: Old Ideas and Recent Results. Proc. "Trends in Computer Algebra" (ed. R. Janssen), Springer Lecture Notes in Computer Science 296 (Springer-Verlag, Berlin-Heidelberg-New York-Tokyo, 1988), pp. 81–91. MR 89f:12001.

[2] Abramowitz,M. & Stegun,I., Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables. US Government Printing Office, 1964. 10th Printing December 1972.

[3] Aslaksen,H., Can your computer do complex analysis?. In: *Computer Algebra Systems: A Practical Guide* (M. Wester ed.), John Wiley, 1999. `http://www.math.nus.edu.sg/aslaksen/helmerpub.shtml`.

[4] Boehm,H. & Cartwright,R., Exact Real Arithmetic: Formulating Real Numbers as Functions. Chapter 3 of D.A. Turner (ed.), *Re-*

search Topics in Functional Programming, Addison-Wesley, 1990, pp. 43–64.

[5] Bradford,R.J., Corless,R.M., Davenport,J.H., Jeffrey,D.J. & Watt,S.M., Reasoning about the Elementary Functions of Complex Analysis. *Annals of Mathematics and Artificial Intelligence* **36** (2002) pp. 303–318.

[6] Bradford,R.J. & Davenport,J.H., Towards Better Simplification of Elementary Functions. Proc. ISSAC 2002 (ed. T. Mora), ACM Press, New York, 2002, pp. 15–22.

[7] Collins,G.E., Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition. Proc. 2nd. GI Conference Automata Theory & Formal Languages (Springer Lecture Notes in Computer Science 33) pp. 134–183. MR **55** (1978) #771.

[8] Corless,R.M., Davenport,J.H., Jeffrey,D.J. & Watt,S.M., "According to Abramowitz and Stegun". *SIGSAM Bulletin* **34** (2000) 2, pp. 58–65.

[9] Corless,R.M. & Jeffrey,D.J., The Unwinding Number. *SIGSAM Bulletin* **30** (1996) 2, pp. 28–35.

[10] Coste,M. & Roy,M.F., Thom's Lemma, the Coding of Real Algebraic Numbers and the Computation of the Topology of Semi-Algebraic Sets. *J. Symbolic Comp.* **5** (1988) pp. 121–129. Also *Algorithms in Real Algebraic Geometry* (ed. D.S. Arnon and B. Buchberger), Academic Press, London, 1988.

[11] Davenport,J.H., Equality in Computer Algebra and beyond. To appear in *J. Symbolic Comp.*

[12] Delaunay, Ch., Théorie du mouvement de la lune. Extract from the Comptes Rendus de l'Académie des Sciences, Vol. LI (1860).

[13] Della Dora,J., DiCrescenzo,C. & Duval,D., About a new Method for Computing in Algebraic Number Fields. Proc. EUROCAL 85, Vol. 2 (Springer Lecture Notes in Computer Science Vol. 204, Springer-Verlag, 1985) pp. 289–290.

[14] Dicrescenzo,C. & Duval,D., Algebraic Extensions and Algebraic Closure in SCRATCHPAD II. Proc. ISSAC 1988 (ed. P. Gianni), Springer Lecture Notes in Computer Science 358, Springer-Verlag, 1989, pp. 440–446.

[15] Duval,D., Algebraic numbers: an example of dynamic evaluation. *J. Symbolic Comp.* **18** (1994) pp. 429–445.

[16] Gates,B.L., A Numeric Code Generation Facility for Reduce. Proc. SYMSAC 86, ACM, New York, 1986, pp. 94–99.

[17] Gonnet,G.H., New Results for Random Determination of Equivalence of Expressions. Proc. SYMSAC 86, ACM, New York, 1986, pp. 127–131.

[18] Gradshteyn,I.S. & Ryzhik,I.M. (ed A. Jeffrey), Table of Integrals, Series and Products. 5th ed., Academic Press, 1994.

[19] Hur,N. & Davenport,J.H., An Exact Real Arithmetic with Equality Determination. Proc. ISSAC 2000 (ed. C. Traverso), ACM, New York, 2000, pp. 169–174. MR 2002c:11171.

[20] IEEE Standard 754 for Binary Floating-Point Arithmetic. IEEE Inc., 1985.

[21] Jeffrey, D.J., Integration to obtain expressions valid on domains of maximum extent. Proc. ISSAC 1993 (ed. M. Bronstein), ACM, New York, 1993, pp. 34–41.

[22] Jeffrey,D.J., Hare,D.E.G. & Corless,R.M., Unwinding the branches of the Lambert W function. *Mathematical Scientist* **21** (1996) pp. 1–7.

[23] Kahan,W., Branch Cuts for Complex Elementary Functions. *The State of Art in Numerical Analysis* (ed. A. Iserles & M.J.D. Powell), Clarendon Press, Oxford, 1987, pp. 165–211.

[24] Lazard,D. & Rioboo,R., Integration of Rational Functions — Rational Computation of the Logarithmic Part. *J. Symbolic Comp.* **9** (1990) pp. 113–115. MR 91h:68091.

[25] Lenstra,A.K., Factoring Polynomials over Algebraic Number Fields. Proc. EUROCAL 83, Springer Lecture Notes in Computer Science 162, Springer-Verlag, 1983, pp. 245–254.

[26] Mahler,K., An Inequality for the Discriminant of a Polynomial. *Michigan Math. J.* **11** (1964) pp. 257–262.

[27] Ménissier-Morain,V., Arithmétique exacte, conception, algorithmique et performances d'une implémentation informatique en précision arbitraire. Thèse, Université Paris 7, Dec. 1994.

[28] Mulders,T., A note on subresultants and the Lazard/Rioboo/Trager formula in rational function integration. *J. Symbolic Comp.* **24** (1997) pp. 45–50. MR 98c:26001.

[29] Rich,A.D. & Stoutemyer,D.R., Capabilities of the MUMATH-79 Computer Algebra System for the INTEL-8080 Microprocessor. Proc. EUROSAM 79 (Springer Lecture Notes in Computer Science 72, Springer-Verlag, Berlin-Heidelberg-New York) pp. 241–248.

[30] Richardson,D., Some Unsolvable Problems Involving Elementary Functions of a Real Variable. *J. Symbolic Logic* **33** (1968) pp. 514–520.

[31] Rioboo,R., Real algebraic closure of an ordered field, implementation in Axiom. Proc. ISSAC 1992 (ed. P.S. Wang), ACM, New York, 1992, pp. 206–215.

[32] Rioboo,R., Towards Faster Real Algebraic Numbers. Proc. ISSAC 2002 (ed. T. Mora), ACM, New York, 2002, pp. 221–228.

[33] Risch,R.H., Algebraic Properties of the Elementary Functions of Analysis. *Amer. J. Math.* **101** (1979) pp. 743–759. MR 81b:12029.

[34] Stoutemyer,D.R., Crimes and Misdemeanors in the Computer Algebra Trade. *Notices AMS* **38** (1991) pp. 779–785.

[35] Stoutemyer,D.R., Should computer algebra programs use ln x or ln $|x|$ as their antiderivation of $1/x$. *DERIVE Newsletter* **26** (Jun 1997) pp. 3–6.

[36] Vuillemin,J.E., Exact real computer arithmetic with continued fractions. *IEEE Trans. Computing* **39** (1990) pp. 1087–1105.

[37] Weibel,T. & Gonnet,G.H., An Assume Facility for CAS with a Sample Implementation for Maple. Proc. DISCO '92 (Springer Lecture Notes in Computer Science 721, ed. J.P. Fitch), Springer, 1993, pp. 95–103.

[38] Wolfram,S., *The Mathematica Book*. Wolfram Media/C.U.P., 1999.

[39] Zwillinger,D. (ed.), *CRC Standard Mathematical Tables and Formulae*. 30th. ed., CRC Press, Boca Raton, 1996.