Well...It isn't Quite That Simple

Robert M. Corless and David J. Jeffrey

Department of Applied Mathematics
The University of Western Ontario
London, Ontario, CANADA, N6A 5B7
rcorless@uwo.ca
djeffrey@uwo.ca
SIGSAM Bulletin Vol 26(3) Issue 101, 1992

Present computer algebra systems base their interactive sessions on a very simple model of mathematical discourse. The user's input to the system is a line containing a mathematical expression (an operation, a formula, a set of equations, etc) and the system's response to the user is an output line which contains a mathematical expression similar to the input. There are many situations, however, in which this is too simple a model of mathematics. Algebra systems should be allowed to reply 'Well ... it isn't quite that simple'.

Consider, as a starting problem, the evaluation of a definite integral. The table of integrals by Gradshteyn and Ryshik contains the entry [1, 3.310]

$$\int_0^\infty e^{-px}\,dx = \frac{1}{p}\,, \qquad [\Re(p) > 0]. \tag{1}$$

The requirement $\Re(p) > 0$ is an essential part of this result, and yet there is not, at the moment, any standard mechanism whereby a CAS can reproduce (1), complete with proviso. A more complicated example is [1, 3.222.2]

$$\int_0^\infty \frac{x^{\mu-1}}{x+a}dx = \begin{cases} \pi\,\mathrm{cosec}(\mu\pi)a^{\mu-1} & \text{for } a > 0, \\ -\pi\cot(\mu\pi)(-a)^{\mu-1} & \text{for } a < 0, \end{cases} \qquad [0 < \Re(\mu) < 1].$$

Here we see different forms being offered for the integral depending upon the value of a, and again there is a proviso. So, although most entries in Gradshteyn and Ryshik consist of formula = formula, sometimes it is not quite that simple.

To explore how CAS at present handle problems such as (1), we tried it on all systems within easy reach. We also tried a second problem, which was to solve

$$kx = k\,. \tag{2}$$

We used these systems as a naive user would, selecting the obvious commands and not using any of the assume facilities. The replies we received are listed in table 1. There is a blank entry for Theorist's answer to problem 1 because neither of us uses Theorist often, and we are not confident that the answer we obtained from it is a fair representation of its abilities. Mathematica's answer to problem 2 was obtained using its '`Solve`' command, but we are aware that if we had used its '`Reduce`' command we would have obtained an answer equivalent to Theorist's. We are also aware that we can force Maple and Derive to assume $p > 0$ and thereby get the integral evaluated. These are side issues, however,

CAS	*Problem 1*	*Problem 2*
Mathematica	$1/p$	$x = 1$
Maple	$\int_0^\infty \exp(-px)dx$	$x = 1$
Derive	?	$x = 1$
ALJABR	Is p +, -, or zero?	x = 1
Theorist	—	$x = 1$ if $k \neq 0$ x undetermined if $k = 0$

Table 1: Responses of popular CAS to 2 standard problems.

---

and in spite of them, we find all of the responses in the table unsatisfactory, for reasons we now discuss.

Mathematica's answer for Problem 1 is unsatisfactory because it is incorrect on specialization. If, having obtained it, we later set $p = -2$, then the result is simply wrong. To be fair, the answer is correct if p happens to be positive, and the user may be lucky. The bottom line, nevertheless, is that Mathematica has (to use a popular term nowadays [2,3]) lied to the user: the integral of $\exp(-pt)$ from $t = 0$ to $t = \infty$ is not always $1/p$. Maple's answer is unsatisfactory because it avoids being wrong by avoiding being helpful. In fact, the user may be unaware that Maple can do better, and simply think "Stupid Maple!" before moving on. Maple really relies on the user being sophisticated enough to realize what is needed and to take the appropriate action. For more general problems this can be arbitrarily difficult. Derive's answer is essentially the same as Maple's (though perhaps expressed slightly more clearly), and is unsatisfactory for the same reason.

ALJABR's response, namely to ask the user a question about p, is unsatisfactory for a different reason. If we attempt a more complicated problem, suppose for example the integral is generalized to $\int_0^\infty \exp(f(p)t)dt$ as it is in equation(3) below, the question that ALJABR would ask, namely whether $f(p)$ is positive, negative, or zero, can be arbitrarily difficult to answer. In addition, not all computations are performed interactively, batch computations are still used, and clearly such questions are inappropriate in that context.

Next, we consider Problem 2. All the CAS but Theorist produce the unsatisfactory answer $x = 1$. Again, this answer is incorrect on specialization of the problem to $k = 0$, but the way in which it is incorrect bears further comment. Compare the solution of $kx = k$ with the solution of $ky = 1$. If we say $y = 1/k$, we have again neglected the exception $k = 0$, but with the difference that if the user later substitutes $k = 0$ into the solution for y, the expression is singular, a visible warning to even the less alert user that something is wrong. The solution $x = 1$, in contrast, fails invisibly when $k = 0$. It is the invisible nature of the failure that makes the usual solution to problem 2 particularly unsatisfactory. By the way, ALJABR does not stop to ask whether k is zero or non-zero when solving this problem, because, like most systems, it does not test for pivots that would be zero on specialization.

Theorist's answer to problem 2 was exemplary, but we now suggest, it might seem perversely, that it too is unsatisfactory. The reason is completely different: in larger sets of equations, containing more parameters, it can be combinatorially expensive to explore all the different special cases, *most of which will be of no interest to the user.* Consider a bi-

furcation problem with six free parameters, for example: without some physical knowledge of the reasonable domains for the parameters, this problem could be hopeless to solve.

The automatic exploration of conditions or alternative results requires considerable computational resources, and for the sake of speed there is an attraction to picking one 'obvious' answer. This way of proceeding is also attractive because it fits easily into the existing model of interactive exchange. The difficulty is to balance efficiency against correctness. Some users will say 'I don't care *how* quickly a program can get me the wrong answer' while others will say 'I don't care *how* correct the answer will be, if I have to wait a thousand years to see it'.

As a focus for discussion, we can take a specific problem and list some of the possible templates that a CAS could use when presenting its solution. Given the integral

$$\int_0^\infty \exp(a^3 - a)x \; dx, \tag{3}$$

a CAS could return any of the following responses.

(A) An unevaluated integral: $\displaystyle\int_0^\infty \exp(a^3 - a)x \; dx.$

It could be returned unevaluated, but with an indication of why.

(B) $\displaystyle\int_0^\infty \exp(a^3 - a)x \; dx,$ [ADVICE: Need the sign of $a^3 - a$].

An unqualified result could be returned.

(C) $-1/(a^3 - a)$

One value could be returned together with a proviso.

(D) $-1/(a^3 - a),$ [PROVISO: $a^3 - a < 0$].

If only one value is returned, it may not be obvious which value that should be. A qualified result that is equally valid is

(E) $\infty,$ [PROVISO: $a^3 - a \geq 0$].

We could return all possible values, but without analysing the conditions beyond the form in which they naturally arise.

(F)
$$\begin{cases} \infty, & \text{if } \; a^3 - a \geq 0, \\ -1/(a^3 - a), & \text{if } \; a^3 - a < 0. \end{cases}$$

The condition on $a^3 - a$ could be subject to further analysis, to obtain conditions on a.

(G)
$$\begin{cases} -1/(a^3 - a), & \text{if } \; a < -1, \\ \infty, & \text{if } \; -1 \leq a \leq 0, \\ -1/(a^3 - a), & \text{if } \; 0 < a < 1, \\ \infty, & \text{if } \; 1 \leq a. \end{cases}$$

Of course we could interrupt the computation and ask interactively

(H) Is $a^3 - a$ positive, negative or zero?

This does not exhaust the possible templates, which could stretch at least to (Z).

There are three issues to be dealt with. In the first place, we must decide whether or not the user should be informed of the provisos on the computed results, i.e. we must choose between templace (C) and the others. Then, if it is accepted that we should work with provisos, we must decide whether or not all the special cases should be solved for in advance of the user choosing to explore them, ie, choose between template (F) and (D). In the above example, the integral has only 2 different values, and the condition separating them is easy to see, but in other problems there may be many different cases to be computed. Consider solving a set of linear equations containing several parameters: each special case would become a large computation in itself. Lastly, we must decide whether or not the conditions themselves should be simplified, as in template (G) above.

Before we make a proposal concerning these ideas, we need to emphasise two important points. The first is that we are not aiming to select the one template that will be used by CAS. Over a period of time, for different problems, a user will want to use several of the above possibilities, so several of them must be available. The second is the fact that the issue is not exclusively a computer algebra one; it can also be a mathematical one. We have shown elsewhere [4] that the traditional textbook way of solving a mathematical problem may not be the best way for a computer algebra system, but a modification of the mathematical theory, and more importantly of the user's expectations, can help matters greatly.

An approach which conveniently combines the features of templates (C)—(E) is as follows. The CAS returns results after the fashion of template (C), but if a proviso is required, the statement is stored in a variable, let us call it `lastproviso`. With each new computation, `lastproviso` is rewritten. There was sufficient discussion of this at the 1992 Maple retreat for us to hope that such a feature will be included in a future release of Maple. We anticipate syntax along the lines of the following simulated Maple session.

```
# The user issues a command
solve(k*x = k,x);
                              x = 1
# The system has returned the same solution that it does now,
# but the user is of a cautious disposition and checks for provisos.
lastproviso;
                              k≠0
# Had there been no proviso, this line would have beeen blank, but as it is,
# the user sees that the computed result is correct only provided k ≠ 0.
```

Of course, for more complicated problems ths provision will be replaced by a sequence of qualifiers, all of which must be true for the result to be correct.

This approach allows the user who is confident that the results are correct to ignore the last proviso (i.e. not ask for it) and minimize the computational penalty, while still carrying the insurance that if the results are unexpected or an apparent failure occurs, the provisos can be checked for information about what went wrong. The results are *guaranteed* to be correct, in the sense that the answer is in the form "If such-and-such is

true, then the answer is so-and-so". It is possible that the conditions may be such that they are never true at the same time, but that cannot be a criticism of the approach, because it may be the mathematically correct statement. Without such provisions, the CAS and the user are charging blindly ahead, and the '`simplify`' command should really be re-named '`oversimplify`'.

The above proposal is only one mode of operation. The important issue here is *user control*. In some fashion, the user's expectations about the parameter values must be taken into account or prodded into existence. Facilities should be available for the user to establish a knowledge base of what is known about parameters and variables (assume commands, etc), but also facilities should be available for the system to indicate when that knowledge base needs to be expanded. Only the user knows how this is done best. Only the user knows enough about the problem context to decide what regions of the parameter space are relevant. So the issue for the CA community is how to provide the flexibility to get the desired behaviour. A 'mode switch' would go a long way towards this:

```
provisionprinting := complete;
< Thereafter provisions are automatically printed
and simplified with each result>
or
provisionprinting := partial;
< Thereafter provisions are printed only on request but
simplified as the computation proceeds >
or
provisionprinting := background;
< Thereafter provisions are printed on request, but not simplified>
```

The '`assume`' facility of the CAS should play a rôle in the simplification of the provisions but not necessarily in their simple printing.

There are not conclusions here, only ideas put forth for consideration. We have said that we do not like any of the existing methods CAS use to deal with qualifications and provisions on the computed answers, and have suggested a different approach. We remark that if we expect our computer algebra systems to help us do *analysis*, these qualifications and provisions are essential – indeed, in many problems they are the items of main interest.

Acknowledgements This column arose out of a discussion led by Michael Monagan at the 1992 Maple Retreat, and many of those present contributed useful insights. Thanks go to the organizers of the retreat for the chance to be there, and NSERC, ITRC, and WMSI for the financial support that made it possible.

References

1. Gradshteyn, I.S. and Ryshik, I.M. 1979 *Table of integrals series and products*, Academic Press.

2. Corless, R.M. 1993 Six, lies and calculators. *Am. Math. Monthly*, **100**, 344–350.

3. Nievergelt, Y. 1992 Numerical linear algebra on the HP28 or How to lie with supercalculators. *Am. Math. Monthly*, **98**, 539-544.

4. Corless, R.M., Jeffrey, D.J. and Nerenberg, M.A.H. 1992 The row echelon decomposition of a matrix. [Note added 1999: This was eventually published as Corless and Jeffrey 1997 The Turing factorization of a rectangular matrix. *SIGSAM Bulletin*, **31**(3), 20–28.]