COMMENTARY ON:

SOLVING SYMBOLIC EQUATIONS WITH PRESS

Richard Fatemen
Alan Bundy
** Richard O'Keefe
*** Leon Sterling

DAI RESEARCH PAPER NO. 357

Paper submitted to SIGSAM Bulletin 1988

Dr Richard Fateman, Dept of Electrical Engineering & Computer Science
University of California, Berkeley, CA 94720

Dr Richard O'Keefe, Quintas Computer Systems Inc
1310 Villa Street, Mountain View, CA 94041-1126

Dr Leon Sterling, Dept of Computer Engineering & Science
Crawford Hall, Case Western Reserve, Cleveland, Ohio 44106

Commentary on: Solving Symbolic Equations with PRESS

Richard Fateman, Alan Bundy, Richard O'Keefe and Leon Sterling

Abstract

This commentary arose from a review by Richard Fateman of a paper on PRESS, [Sterling *et al* 82], and the subsequent responses from the authors of that paper.

1 Review by Fateman

The paper "Solving Symbolic Equations with PRESS", by Sterling, Bundy, Byrd, O'Keefe and Silver [Sterling *et al* 82], describes a program PRESS (PROLOG Equation Solving System) which solves some symbolic, transcendental, non-differential equations. The problem of solving such equations is divided into two levels, a *meta-level* which is intended as a research vehicle for exploring search strategies in mathematical reasoning and a set of *equation-solving modules* used initially for the MECHO [Bundy *et al* 79] system (a project which aims to solve high-school mechanics problems stated in English.)

The authors suggest that "the techniques used may have something to offer the field of symbolic and algebraic manipulation". We review this paper from this perspective. Previous reports of work on PRESS have appeared generally in AI publications, (IJCAI6, IJCAI17 and AI 16(2)), but it appears that this work has not previously been reviewed from any mathematical perspective.

1.1 What problems does PRESS solve?

The equations that PRESS has been trained to solve are taken from various British A-level examinations. That is, the equations are limited to single-variable problems which are amenable to tricks taught to high-school students. They appear to generally involve applying identities to simplify the problem before it can be solved. Most problems rely on coincidences (such as terms dropping out), rather than on any strong methods, most of which would not be known by high-school students.

I would guess that the objective of the A-level exam questions is to see if the student knows a sufficient collection of identities, and has the ability to search for the clever twist that solves the problem.

For this task, largely heuristic methods are perhaps appropriate. Of the 80 single-variable equations in their sample, the authors are pleased to point out that it can solve 62.

Nowhere is the class of expressions acceptable to the program indicated. It *appears* to allow integers, ordinary arithmetic, only one variable, x, logarithms, trigonometric functions, square-root, and hyperbolic functions. It appears to exclude complex numbers, although they can obviously be constructed. Division appears in no example, but negative exponents allow us to construct division.

1.2 The equation-solving modules

Let us talk about the lower level methods first.

Six "major methods" are implemented.

Some of the methods are obvious, and would be included in any program, heuristic or not, to solve an equation with a single occurrence of an unknown. Some are simplified cases that could be solved by algorithms, but are not. Others methods do not always work, since they produce extraneous solutions or ignore some solutions. These methods are not described exactly in the paper.

Three of the methods, *Isolation*, *Attraction*, and *Collection* attempt to move a (single occurrence) of an unknown to one side of an equation. If the unknown occurs in more than one place, heuristics are used to try to merge these occurrences.

Polysolve attempts to solve polynomial equations. The ones it solves are those likely to occur in "cooked" problems for high-school students: quadratics, or polynomials with small-integer zeros, or minor transformations on them. Any equation of degree 3 or more, or requiring factoring, or involving extra symbolic parameters, or (I suspect) division or gcd for simplification, will not work.

Homogenization is an attempt to find an appropriate substitution to explicitly recognize the relationship of apparently unrelated terms (like $\exp(x)$ and $exp(3x)$).

Function Swapping is essentially a heuristic attempt to apply known inverses to both sides of an equation.

None of these methods break new ground, by comparison to existing algebraic manipulation systems. PRESS must rely on some other ideas if it is to be more than an implementation of weak heuristics.

1.3 Meta-level solving

The meta-level is basically another level of heuristics. I found it curious that some of these heuristics are quite specific (e.g. one that looks for an arithmetic progression of cosines), and others are quite vague. The notion is that local or syntactic clues should direct the search for a method to invert an equation. This is hardly novel; even so, it is not clear that it is a good idea.

If the only alternative were to state the axioms of arithmetic and the "rules for algebraic manipulation" (whatever they may be – and it is not clear what they are!), and search for a solution, this local clue approach would be a better choice. This backwards search would presumably be doomed by combinatorial explosion. This is, however, a straw man.

I doubt that *any* of the methods used by the algebraic manipulation systems MACSYMA, REDUCE, MuMATH, SCRATCHPAD, SMP, or MAPLE include directly accessing axioms of arithmetic. Mostly, they use data-driven algorithms. By contrast, PRESS seem to use the weakest heuristics that can provide the right answer sometimes.

What kinds of local clues have the authors elevated to meta-level concepts? It appears they are ideas such as

Is there more than one occurrence of the unknown?

- How close, measured syntactically, are terms in which the unknown occurs?
- Is the problem a polynomial equation?

I am unimpressed by this meta-level. It does not seem like a useful subdivision of responsibility. For example, if one had a complete polynomial equation solver, a procedure to recognize a polynomial could be attached or put into a separate "level" It should not rely on local clues or heuristics. For example, $(x^2-1)/(x+1) = 0$ is a polynomial equation, as is $x+\sin^2(x)+\cos^2(x) = 1$ Other solving procedures cannot easily recognize the class of problems for which they succeed, without, in effect, applying the algorithm and reaching (in finite time, one hopes), a dead end. A meta-level would not be useful in such a case.

1.4 Should we be impressed by PRESS?

Most mathematicians do not need help to solve this artificial class of high-school algebra problems. In any case, it appears that PRESS does not find all solutions, nor does it distinguish between actual solutions and extraneous ones.

It would help if the paper actually stated the form of the solutions which the computer obtains for the sample problems. There is some evidence that the program produces forms that are not particularly useful. The discussion below even suggests that in some cases that the authors do not know (all) the solutions to some of the examples!

Furthermore, it seems quite unlikely that a "solution vetting [checking] procedure" to eliminate extraneous roots would be feasible with the heuristic methods used by PRESS. Some of the problems are computationally undecidable, for example. Others *can* be solved, but only by much more powerful canonical simplification programs than apparently exist in PRESS.

Since some of the criticisms we have leveled at PRESS may seem rather harsh, we supply some specifics:

Equation 2 is $\cos(x) + \cos(3x) + \cos(5 * x) = 0$.

The authors advocate solving this by a trick that works for "cosine terms in arithmetic progression", a meta-level concept.

Notice that $\cos(x) + \cos(5x) = 2\cos(3x)\cos(2x)$, so we can rewrite the problem as $\cos(3x)(1 + 2\cos(2x)) = 0$. "The $\cos(3x)$ can then be factored out, and the other factor produces the other root after a simple application of Isolation."

The program presumably produces $x = \arccos(0)/3$ or perhaps $\pi/6$, and $x = \arccos(-1/2)/2$, or $\pi/3$.

A reasonable requirement on an equation solver is that it be able to indicate how many solutions there are, and provide a representation for them. It seems doubtful that PRESS responds that there are an infinite number of positive and negative solutions at various multiples of π. If the program were to return the answer as expressions in arccos, it would appear that there were but 2 answers. There is no indication how PRESS deals with sets of answers.

Incidentally, the program seems to deal in degrees, rather than radian measure, for angles.

Equation 6 is $\log(x - 1) + \log(x + 1) = 3$.

The solution is alleged to be $x = \pm\sqrt{e^3 + 1}$. This was obtained by claiming that $\log(x - 1) + \log(x+1) = \log(x^2 - 1)$, which is not always true. For example, when $x = -\sqrt{e^3 + 1}$, the left-hand side is $3 + 2\pi i$. I speculate that the equation was written with absolute-values inside the logs but PRESS could not handle this complication.

Equation 7 is $\exp(3x) - 4\exp(x) + 3\exp(-x) = 0$.

The solutions are alleged to be $x = \log(\sqrt{3})$ and $x = 0$. The additional solutions, $\log(-1)$ and $\log(-\sqrt{3})$ are apparently also produced by PRESS, but it is suggested that a future implementation would reject these. In fact, if $\log(-1) = i\pi$, there is nothing wrong with either of the rejected answers. There are an infinite number of other solutions for other values of $\log(-1)$.

I am sure that there are numerous other traps into which PRESS will fall.

1.5 How does PRESS compare to existing algebraic manipulation programs?

It would appear that PRESS and equation-solving programs in MACSYMA, REDUCE, Mu-MATH, MAPLE, etc., can solve many of the same equations. Having tried some of the examples in the paper with MACSYMA, it seems that the `solve` program itself does not have the capability to handle these problems "cold". However, a preliminary transformation by another simplification

program (e.g. `logcontract`, `trigexpand`, `exponentialize`, `ratsubst`), brings the problem into a solvable form. Thus a front-end which would heuristically call one or more of these algorithmic simplifiers and then call `solve` might be used to construct a poly-algorithm which would out-score PRESS on a set of symbolic equation problems. Whether this would be mathematically interesting would be a separate question. Some of the simplifiers, if applied to random large expressions, could be expensive. For school exam problems this would not be of much concern since they tend to be small and, in some perverse sense, "simple" if you see the trick.

Can PRESS combine the merits of searching/matching with mathematical methods? Probably by discarding most of PRESS and adding some additional heuristics to any of the well-known systems: but generally at the risk of introducing extraneous roots (difficult to check), or possibly missing roots.

There is considerable literature on solving equations by computer, none of it referenced. (See, for example, the rather old paper by Martin and Fateman on MACSYMA [Martin & Fateman 71] or more recent material in the bibliographies in [Buchberger *et al* 83].)

1.6 Conclusion

This paper describes a well-publicized "artificial intelligence" project, PRESS, from a distinguished research institution.

The program, viewed as a procedure for solving mathematical problems, has substantial design flaws proceeding from its basic premise that heuristics and search are primary (or even the only) methods for solving equations.

PRESS appears to not represent an advance in the state of the art in symbolic mathematics. In major respects it seems to be less powerful than programs over 15 years old. Rather, it is an attempt to solve, using weak methods, a set of canned problems unrepresentative of mathematical problems outside the realm of simple school exams.

Although the paper is easy to follow, I would have found it useful if the authors had given concrete results of the output from PRESS (and perhaps other systems) on their examples, and compared their methods with others in the literature (computer and mathematics). However, correcting such problems would only strengthen the superficial plausibility, and not correct the real weakness, which is that the PRESS program starts from a flawed premise.

2 Bundy's Reply to Fateman

Fateman's review raises a number of important points, many of which I agree with, and others I would like to reply to.

2.1 Grounds for Assessing PRESS

Overall, I feel he judges our work on the wrong grounds. Let me say why by explaining the background to the project. Our principle interest was in automatic theorem proving and, in particular, in guiding the search for a proof through a space of rule firings. We thought the use of numeric scores in A* type search strategies rather crude and tried to look for alternative techniques. The choice of equation solving for a domain was because:

(a) here was a domain in which there was a massive amount of search when viewed as a resolution-type problem,

(b) however humans seemed to do pretty well, and

(c) there was no decision algorithm that they might be using (since the domain is undecidable – see Richardson's results [Richardson 68]).

So we had an existence proof for some guidance techniques. PRESS embodied an interesting new guidance technique, and it is on this that it principally has been and should be judged. It has never been presented, by me, as a useful (MACSYMA-like) equation solving package, and whenever anyone has requested to use it on those grounds I have put them right. (So why did we say "the techniques used may have something to offer the field of symbolic and algebraic manipulation"? Well I regret that particular form of words because I now see it is ambiguous. There is "something to offer" which I will discuss below, but it is not the precise equation solving methods of PRESS except, maybe, in a few special cases.)

2.2 Reply to Criticisms

In summary, I take Fateman's criticism to be:

1. (c) above may be true for the whole class of symbolic equations, but probably not for the class PRESS can actually deal with (if only we would say precisely what that was).
2. Decision algorithms exist for that class which are stronger than PRESS's methods.
3. PRESS's 'solutions' are sometimes incorrect.

It is difficult to counter 1 because the nature of PRESS makes it hard to specify the class of equations that it deals with. PRESS is probably a terminating algorithm. Certainly each method is individually, but there is no guarantee that the application of methods will terminate. Thus PRESS is probably a decision algorithm for whatever class it deals with. (Note, by the way that PRESS is not restricted to single equations in one unknown as Fateman claims in section 1.1).

On 2, Fateman states "None of these methods break new ground, by comparision to existing algebraic manipulation systems", but he gives no evidence for this claim. I accept his claim in the case of Isolation and Polysolve, but would like some pointers into the literature in the case of the other methods, which don't seem anything like the conventional methods I know of. In fact, I understood from Bernard Silver that Fateman and he found an example that PRESS could handle and that MACSYMA (for instance) could not. However, it would not surprise me if Fateman were right. It was never our intention to provide a state-of-the-art equation solver, but merely to find some interesting examples of the kind of reasoning which might take place during the guidance of problem solving. The discovery of new methods during this exercise would be entirely fortuitous.

On 3, I am prepared to admit there may be errors in PRESS, but the two examples Fateman quotes in section 1.4 are not errors. We took a decision to adopt the usual high-school, equation-solving procedure of deriving consequences from the original equation until a formula of the right syntactic form to be a solution is derived. This formula is guaranteed to include all solutions, but may also include non-solutions if any derivation step was non-reversible. These must then be weeded out by checking, but I'm afraid we never got round to implementing checking because it was not central to our investigation of search control. In the log example there are 2 non-reversible steps:

$$\log_e(x-1) + \log_e(x+1) => \log_e(x+1)(x-1)$$

and

$$x^2 = e^3 + 1 => x = \pm\sqrt{e^3+1}$$

Note that in the directions indicated the inferences are ok, e.g. if $x+1$ and $x-1$ are positive then $(x+1)(x-1)$ is.

We also took a decision to limit ourselves to real valued solutions (although this was a more arbitrary decision). That is why $\log_e(-1)$ etc should have been rejected by the missing checking step in Fateman's second example.

In general, the meta-level inference architecture lends itself well to spotting errors, because of its separation of factual and control information. Decision algorithms typically interwine these making checking difficult. This does not mean we spotted them all.

2.3 Contribution of PRESS

So what has PRESS got to offer conventional systems? I see two potential applications:

1. Meta-level inference may be a useful technique for controlling the application of conventional methods and/or for explaining their application to users.

2. Some of the PRESS methods may extend conventional methods or may provide more efficient alternatives in some circumstances.

As Fateman notes in section 1.4, MACSYMA's solve routine is limited in its abilities to solve equations 'cold'. This was my experience too. It seems to consist of Isolation plus some powerful, but special-purpose polynomial methods. However, its abilities can be increased by preliminary transformation using a simplification method. This will involve some planning. My idea under 1 above, is that each MACSYMA method carry a meta-level description of its preconditions and effects that could be used by a PRESS-like system to plan a solution strategy. This strategy could also be the basis of an explanation to the user, e.g. "You wanted to solve $(x^2-1)/(x+1)=0$ but this contains 2 occurrences of x and is not a polynomial so I could not do it directly. However, it is a rational function, which I transformed to a polynomial $x-1=0$ using Ratsimp and then solved to give $x=1$." (Note: PRESS does not give explanations in English, but a formal version of the above.)

My idea under 2 is that some of the PRESS methods might help fill in the gaps in existing MACSYMA methods and cope with unusual situations like $x+\sin^2(x)+\cos^2(x)=0$. This is neither a polynomial nor a trigonometric equation (although it can be reduced to a polynomial of course). Conventional methods are likely to be stumped (if not on this one exactly, I am sure there are others). However, Collection can be applied to this to remove 2 occurrences of x and produce $x=0$.

As for efficiency, I have no concrete examples, but I know that decision algorithms can be very inefficient and that heuristic methods can be an efficient alternative in particular circumstances. For instance, I understand that some of the original SIN integration techniques were left in MACSYMA despite the Risch-Norman algorithm. I also have had personal communications from Joel Moses and John Campbell about the potential value of this in equation solving.

3 O'Keefe's Reply to Fateman

Fateman says "nowhere is the class of expressions acceptable to the program indicated." This is ambiguous. It could mean

- What is the syntax which the program can read?

  The answer in some sense is "any PROLOG term".

- What is the class of forms which the program will not only read but accept responsibility for solving, and acknowledge failure as its fault, rather than blaming the user for submitting an ill-posed problem?

  This is, I think, what Fateman wanted.

- What is the class of equations which the program can actually solve?

  This is the really interesting question, but I doubt whether it can be answered.

In fact, from *our* point of view, the third reading isn't terribly interesting. We could have programmed up some of the polynomial factorisation methods that are in the literature, beefing up "polysolve". But there is no general method for polynomials of degree > 3, so this would just be strengthening a heuristic, and the question of interest for us was "when it is a good idea to try poly_solve and what does its success guarantee us". Similarly, adding a few more axioms to attraction might have helped us solve a few more equations, but wouldn't have told us anything more about attraction as a method.

Our failure to answer the second reading is a fault. We should have indicated that PRESS could to some extent handle inequalities and simultaneous equations. We should also have said what the output looked like. Fateman slams PRESS hard when he says "A reasonable requirement on an equation solver is that it be able to indicate how many solutions there are, and provide a representation for them. It seems doubtful that PRESS responds that there are an infinite number of positive and negative solutions at various multiples of π." But of course PRESS *does* indicate and represent such solution sets, even solution sets with several index variables.

Fateman's claims about what a heuristic front-end to MACSYMA *might* be able to do remain (a) just claims and (b) support for the PRESS approach.

There are two very important things about PRESS which Fateman doesn't mention at all.

First, he appears to be under the impression that Isolation, Collection, Attraction, and so on are an utterly unrelated set of separate heuristics and that their application is "blind". I regard it as one of the interesting things about PRESS that this is not true. For example, we tried isolation before collection, but it would have made no difference if we had tried collection before isolation, because the two can never be applicable to the same equation. Poly_solve is anomalous, because it might act as isolation, collection, or even act when one might have expected only attraction to work. I find it inconceivable that Bernard Silver's work on LP [Silver 85] could have started from anything in MACSYMA. It is not without significance that Leon Sterling has a version of PRESS in Flat Concurrent PROLOG [Sterling 86], a system with no provision for backtracking at all...

Second, Homogenisation is a genuinely hard task. I suspect that Bernard's methods will not scale well, but I was in the habit of reading SigSam bulletin and anything I could find on Computer Algebra, and I never came across anything which approached Homogenisation. I could well be mistaken about that. Fateman says "None of these methods break new ground". With respect to Homogenisation, it would become him to supply references to back up this rather startling claim. My impression is that people in the computer algebra field found very early on that it was hard and switched to problems and algorithms which don't need change of variable, or where the change of variable is built into the algorithm and requires no search.

4 Fateman's Reply to O'Keefe

Beginning from the top, I think that what I had in mind about classification was that it is generally considered appropriate for the author(s) of a program to have a specification for their program, or at least a stated goal. It would be nice to know the answers to all of the questions, including, in the first category, does PRESS know a set of names by which one can ask about hyperbolic functions and their inverses? (e.g. acosh or arccosh ?) Can it solve equations which involve indeterminates other than the one being solved for?

In the second category, It seems that O'Keefe agrees with my criticism, namely that regardless of what PRESS' authors would have learned by changing the program, it would nevertheless be nice to know that PRESS can (or as it happens, cannot) solve some polynomial equations according to whether they are linear, quadratic, or by substitution can be made linear or quadratic in x^n for

some n. Solutions expressed by indexed sets was a feature of PRESS which was not mentioned in the article under review. If some sample answers had been supplied, it would have been evident. Even so, the cos problem results in two such indexed sets, when one index would suffice.

In the third category, since Bundy believes PRESS always terminates, it appears that the class of expressions solvable is fully determined logically by merely running the program. Thus the answer, when O'Keefe doubts it can be answered, is in some sense trivial.

I believe that the ingredients in PRESS could be added to a system like that in MACSYMA, if it were deemed cost-effective (That is, able to solve more problems without hobbling the common cases with costly unnecessary transformations, and introducing wrong answers sometimes.). As such it would be a relatively minor modification of MACSYMA. I think it would be a relatively major modification of PRESS to include general algorithms for cubic and quartic equations, polynomial factorization, solving algebraic systems, etc.

I doubt that I understand the relationship, in PRESS, of the various methods. I think that it is fairly simple, in MACSYMA. There is a logical progression of classification tests. When one succeeds, it reduces the problem and either produces an answer, calls the solve program again on any incompletely reduced equation(s), or declares that it can't be done using any of the built-in methods. The fact that PRESS can be written in a PROLOG lacking back-tracking is not surprising, because backtracking didn't seem to be a required strategy for Solve.

While no proof has been attempted to show that the existing processing steps in the MACSYMA solve program terminate, they are all fairly obviously moving toward the goal of producing an equation with a single variable, unencumbered, on one side of the equation.

It would not be difficult to change the solve program so that it stores all incompletely reduced equations on a list, and before "giving up", attempts to crack these equations by applications of any or all of the dozen or so additional tricks not used (e.g. converting trig forms to complex exponentials). The problem is, these transformations can be very expensive. Control mechanism like "try this function for ten CPU seconds only" have never been very appealing. Perhaps when we are encouraged to gamble with computer time (as parallel processors become more common), this will change.

I do not understand the comment concerning LP and MACSYMA. MACSYMA is written in LISP. If it is inconceivable that LISP could be used for LP, then it must be remarkable.

Homogenization, if I understand it fully, was used by the **radcan** program described and illustrated in the Martin and Fateman paper [Martin & Fateman 71], as well as my PhD dissertation [Fateman 72a] and also, my paper [Fateman 72b] discusses this.

The idea of forming a minimal algebraic, exponential, or logarithmic basis or "set of kernels" is essential for simplification or other operations including Risch integration. Because of its fundamental nature, it is not surprising that it was found to be useful in PRESS.

However, examination of earlier work in the area would show that this process has not been neglected, nor have most people avoided it.[1]

Other references including papers by B. F. Caviness [Caviness 67] and W. S. Brown [Brown 69] on simplification may be found in the *Computer Algebra* monograph edited by B. Buchberger.

5 Sterling's Comments about PRESS

First, a rather obvious comment. Several people have contributed to the current state of PRESS. Each of us have had, and do have, slightly different views about PRESS' significance. So there can be seeming contradictions in comments. An outside reviewer needs to consider the different viewpoints of the authors of PRESS in assessing what the program has accomplished.

[1] I recall one implementation of integration in which the user had to provide the set of kernels, but this was an exception. It post-dated the implementation of the Risch algorithm in MACSYMA (c. 1971).

There are three issues discussed in the exchanges which I want to comment on.

(a) What exactly does PRESS do?

(b) Does PRESS have anything to offer the symbolic algebra community, and if so, what?

(c) How easy would it be to embed PRESS in MACSYMA, and vice versa?

5.1 What exactly does PRESS do?

The version of PRESS described in the paper was primarily built to model the behavior of A-level students solving actual equations which appeared in A-level exams. Consequently, no attempt was made to embed equation solving knowledge beyond the scope of A-level students, such as a general solution to a cubic or quartic equation, even though adding such a general solution would be trivial. Furthermore, little effort was devoted to expressing answers in simplest form, a not so trivial extension.

A common question asked when presenting talks about PRESS was 'What is the class of equations that PRESS solves?' Since no-one asks what is the class of equations that an A-level student solves, we had not studied this problem.

Before presenting our work to the symbolic algebra community, we probably should have done more research on this question, or at least done a complete search of the literature.

I do not believe, however, that there is an easy characterization of the class of equations actually solved by PRESS, nor do I believe that the question is especially useful. The algorithm or procedure used to match an expression to the left-hand side of a rewrite rule illustrates the difficulty. The initial version relied on PROLOG's backtracking and was combinatorially explosive with a large expression. I believe, though there is no concrete evidence, that an expression with many addends and multiplicands would have caused stack overflow. The revised version, which dramatically improved the running time, was ad-hoc. The methods themselves clearly terminate, it is the symbolic manipulation such as done by the pattern matcher or expression tidier that causes problems.

However, the behavior of other symbolic algebra systems is not clear-cut. I heard a paper at a EUROSAM conference on hacks to make a computer algebra program, in this case REDUCE, work more efficiently. The hacks made the difference between whether the problem was successfully solved or not. Does anyone characterize what equations MACSYMA actually solves? What is the limit on numbers of variables etc?

My experience with symbolic integration, reported in [Sterling 83] shows that the exact utility of computer algebra systems is not easily understood.

5.2 Does PRESS have anything to offer the symbolic algebra community, and if so, what?

My opinion of what PRESS contributes to the computer algebra community is insight on the value of heuristics for building programs. It is not clear that a program embodying a general algorithm is always preferable to a heuristic problem solver. Experience with programs for symbolic integration *in practice* may tell the same story. There is value in a heuristic program which is easy to build.

How easy would it be to embed PRESS in MACSYMA, or vice versa?

The discussion addressing the issue of whether PRESS can be embedded in MACSYMA and vice versa, alluded to the LISP v PROLOG debate. I have no doubt that PRESS could now be written in LISP fairly easily. What is also true, however, is that there was immense benefit by writing

PRESS cleanly and *logically*, which is a direct consequence of good PROLOG programming style. This led directly to the learning program, LP, and the translation of PRESS to parallel logic programming languages, as mentioned by Richard O'Keefe. Whether PRESS would have been rewritten logically if initially written in LISP is unclear to me.

Many of MACSYMA's methods could be trivially added to PRESS, for example more sophisticated polynomial equation solving. What would be harder to add is MACSYMA's robustness, which is an unfair comparison since PRESS was not designed to be robust. One problem is intermediate expression swell, but that is a problem for general computer algebra systems – again note the SIGSAM article.

I don't have enough knowledge to know how to add PRESS methods to MACSYMA – it did not seem obvious to me as a naive user. But an expert in using MACSYMA should decide.

References

[Brown 69] W.S. Brown. Rational exponential expressions and a conjecture concerning π and e. *Am Math. Mon.*, 76:28–34, 1969.

[Buchberger *et al* 83] B. Buchberger, G.E. Collins, R. Loos, and R. Albrecht. *Computer Algebra. Computing Supplement 4*, Springer-Verlag, NY, 1983. 2nd Edition.

[Bundy *et al* 79] A. Bundy, L. Byrd, G. Luger, C. Mellish, R. Milne, and M. Palmer. Solving mechanics problems using meta-level inference. In B.G. Buchanan, editor, *Proceedings of IJCAI-79*, pages 1017–1027, International Joint Conference on Artificial Intelligence, 1979. Reprinted in 'Expert Systems in the microelectronic age' ed. Michie, D., pp. 50-64, Edinburgh University Press, 1979. Also available from Edinburgh as DAI Research Paper No. 112.

[Caviness 67] B.F. Caviness. *On Canonical Forms and Simplification.* Unpublished PhD thesis, Carnegie-Mellon University, 1967. Also in Journal ACM Vol 17, No 2 (1970) pp385-396.

[Fateman 72a] R.J. Fateman. *Essays in Algebraic Simplification.* Unpublished PhD thesis, MIT, April 1972. also available as MAC TR-95.

[Fateman 72b] R.J Fateman. Rationally simplifying non-rational expressions. *SIGSAM Bulletin*, (23):8–9, July 1972.

[Martin & Fateman 71] W.A. Martin and R.J. Fateman. The macsyma system. In S.R. Petrick, editor, *Proc. of the 2nd Symposium on Symbolic and Algebraic Manipulation*, pages 59–75, ACM, 1971.

[Richardson 68] D. Richardson. Some undecidable problems involving elementary functions of a real variable. *Symbolic Logic*, 33(4):514–520, December 1968.

[Silver 85] B. Silver. *Meta-level inference: Representing and Learning Control Information in Artificial Intelligence.* North Holland, 1985. Revised version of the author's PhD thesis, DAI 1984.

[Sterling 83] L. Sterling. Of integration by man and machine. *SIGSAM Bulletin*, (17):21–24, 1983.

[Sterling 86] L. Sterling. Pressing for parallelism: a prolog program made concurrent. *Journal of Logic Programming*, 3(1):75–92, April 1986.

[Sterling *et al* 82] L. Sterling, A. Bundy, L. Byrd, R. O'Keefe, and B. Silver. Solving symbolic equations with press. In J. Calmet, editor, *Computer Algebra, Lecture Notes in Computer Science No. 144.*, pages 109–116, Springer Verlag, 1982. Also available from Edinburgh as Research Paper 171.