Should Students Learn Integration Rules?

BUCHBERGER, B.

To appear in SIGSAM Bulletin

RISC-Linz Series no. 89-07.1 November 13, 1989

Sponsored by:

Should Students Learn Integration Rules?

Bruno Buchberger

November 13, 1989

RISC-Linz
Research Institute for Symbolic Computation
Johannes Kepler University, A 4040 Linz, Austria
Phone: Austria (7236) 3231 ext. 41 or 20, Bitnet: K313370 @ AEARN

Abstract

In this note we formulate a didactic principle that can govern the use of symbolic computation software systems in math courses. The principle states that, in the treatment of each subarea of mathematics, one must distinguish between a "white-box" and a "black box" phase. In the "white box" phase, algorithms must be studied thoroughly, i.e. the underlying theory must be treated completely and algorithmic examples must be studied in all details. In the black box phase, problem instances from the area can be solved by using symbolic computation software systems. This principle can be applied recursively.

1 The Question

Should students learn integration rules?

Of course, I did not choose this question as the title of this little note for its own sake. Rather, I want to use this question as a paradigm for similar questions that are constantly asked as a reaction to the availability of symbolic math software systems and their potential impact on math education.

It is well known that these software systems (for example, MACSYMA, SCRATCHPAD, REDUCE, MATHEMATICA, ...) provide mathematical problem solving power that outperforms students who have passed, say, the usual introductory calculus and linear algebra courses. For example, integration, the eye-catcher of our note, is much more deeply, more systematically, and, of course, much more efficiently handled in symbolic math systems than was ever possible in calculations by humans. There is no inherent limit to the scope, in breadth and depth, that symbolic math systems will be able to achieve in the future. For example, by recent advances, even theorem proving - the essential nucleus of mathematical activity - has been trivialized in certain areas of mathematics, e.g. geometrical theorem proving.

In this context, I would like to call an area of mathematics *"trivialized"* as soon as there is a (feasible, efficient, tractable) algorithm that can solve any instance of a problem from this area.

In this sense the area of *arithmetic on the natural numbers is trivialized* because there exist algorithms for addition and multiplication on the natural numbers in the "symbolic representation" as decimals. (Note that arithmetic, in early stages of mathematics, was not "trivial" at all and humans who were able to perform arithmetical operations even in a limited number range were deemed to be "intelligent" far beyond average.)

The area of *integrating functions* described by elementary transcendental expressions (i.e. expressions involving variables, arithmetical operators, log, and exp) *is trivialized*: Risch's algorithm decides, in finitely many steps, whether the integral of a given elementary transcendental function is again elementary transcendental and, in the positive case, produces the expression that describes the integral. Furthermore, this algorithm, in combination with heuristic methods, is efficient and, on modern hardware, outperforms humans by orders of magnitude.

Similarly, *geometrical theorem proving is trivialized*: Wu's algorithm or the Gröbner basis algorithm decide in finitely many steps whether or not a given geometrical theorem that can be expressed by equational hypotheses and an equational conclusion is true or not (and, again, these algorithms work amazingly fast on quite non-trivial examples).

Given the present and future power and potential of symbolic math systems it is therefore near at hand to pose the following general question

> *Should math students learn area X of mathematics when this area has been trivialized?*

I think this question has a very simple answer, see Section 3. Nevertheless, much time is still spent in discussions on this question because most people think that they must associate themselves with one of the two possible extreme answers outlined in Section 2. Therefore, I decided to write this little note in order to save some time in future discussions and to make it possible to proceed to the next, more practical, question of how to use symbolic math systems in math education.

In fact, answers to the above question have and will have enormous impact on the structure of math education both at the high school and the university level. The availability of symbolic math software systems with their ability to solve most of the problems treated in the usual undergraduate math courses cannot be just ignored. Either one must deliberately decide, and give arguments, that symbolic math software systems are of no value for math education. Or one must make serious efforts for integrating the use of these systems into the math curricula both for math majors and for others. One cannot just stay "neutral".

In this note, I concentrate on the question for math and computer science majors and do not pursue the question any further for math "users" as, for example, engineers or physicists. This is because I think that the question for math users is much easier than for math majors. Namely, depending on how rare or frequent the contact with mathematical applications may be, for math users it might be really sufficient in the future that they learn to formulate their mathematical problems and to use a good symbolic math software system for obtaining solutions. I treat math *and* computer science majors in one frame because, personally, I believe that there is no essential difference between mathematics and computer science. I have explained this belief at various other occasions.

2 Two Extreme Answers

2.1 No Integrals Any More

The first extreme answer to our question is:

> No, students need not learn itegration rules any more or, more generally, as soon as an area of mathematics is trivialized, math (and other) students should not be tortured with it any more.

People advocating this answer typically would add:

> Having numerical pocket calculators at hand, we were able to stop torturing our children with arithmetical calculations. Analogously, we can now stop torturing students with symbolic calculations and make their minds free for "creative" activity.

The argument against this position is near at hand and is, in fact, held emphatically:

> Integration rules (or, more generally, a "trivialized" area of mathematics) is not only taught because the rules (algorithms) can be applied for solving more complex problems but, mainly, because, by teaching the area, students gain mathematical insight and learn important general mathematical problem solving techniques. Each area of mathematics, in particular, also the "trivialized" areas have their own specific insights and techniques and it would be disastrous to eliminate them from the curricula.

2.2 Ignore Integration Software

The other extreme answer is opposite to the first answer:

> For math education, ignore the existence of integration software and, in general, of symbolic math software systems.

People who support this position would add:

> Symbolic math software systems may have a significant value for applications but they should be banned from math education because they spoil students similarly as pocket calculators have spoiled kids. In fact, the impact of such systems on math education, by the breadth of their scope, will be worse than the impact pocket calculators had on school kids.

The argument against this extreme position is immediate, again:

> Why should we bother students with learning to do trivialized mathematics when the time could be used for moving on to the study of more advanced areas? In fact, symbolic math software systems could open the chance to

drastically expand the number and in-depth-treatment of subjects covered in a typical math curriculum because significant time could be saved during undergraduate education.

3 My Answer

3.1 The White-Box/Black-Box Principle

I repeat the question:

> *Should math students learn area X of mathematics when this area has been trivialized?*

My answer is:

> I think it is totally inappropriate to answer such a question by a strict "yes" or "no". Rather, the answer depends on the stage of teaching area X.
>
> - In the *stage where area X is new* to the students, the use of a symbolic software system realizing the algorithms of area X as black boxes would be a disaster (see however the remark at the end of Subsection 3.4). Students have to study the area thoroughly, i.e. they should study problems, basic concepts, theorems, proofs, algorithms based on the theorems, examples, hand calculations.
> - In the *stage where area X has been thoroughly studied*, when hand calculations for simple examples become routine and hand calculations for complex examples become intractable, students should be allowed and encouraged to use the respective algorithms available in the symbolic software systems.

Let me call this didactical principle the

> *White-Box/Black-Box Principle for Using Symbolic Computation Software in Math Education.*

3.2 Details on the First Stage

In the first stage, where area X is new to the students, the mathematical theory (definitions, theorems, proofs) must be developed on which the (algorithmic) solution to the problems studied in area X is based. This is the stage where mathematical insight and new mathematical techniques are acquired. It would be disastrous for the future of mathematics if the insights and techniques that can be taught and learned in this stage would be ignored because the area is "trivialized" and, therefore, in principle, all problems in this area can be handled by the available systems.

Also, the students must learn how to exploit and transform the mathematical theorems developed in area X for algorithmic solutions (or, at least, heuristic recipes) for problems in area X. And, of course, algorithms are best understood when developed and trained in the context of concrete (simple) examples that span the most typical problem instances. In addition, the systematic design of (efficient) algorithms on the basis of mathematical theorems is a skill that is equally important as proving theorems. (More profoundly, it is clear that, essentially, proving and designing algorithms is the same. In the context of modern declarative programming languages, the identity of proving and designing algorithms becomes a living reality.)

When doing hand calculations for solving problems from area X, it is clear that subproblems from earlier areas X', ... should be treated by applying the algorithms for area X', ... developed earlier and, hopefully, availabe in the symbolic software system at hand.

Example 1 (Integration) Heuristic solutions to the problem of integration (of elementary transcendental functions) give rise to a wealth of mathematical concepts, insights and techniques that are, of course, of central importance even if one knows that ultimately the problem can be handled by a software system. This is even more true if one seeks completely algorithmic solutions to the problem. The mathematics necessary for Risch's integration algorithm (e.g. Liouville theorems) goes far beyond what is normally taught in undergraduate analysis courses and, therefore, can form an extremely worthwhile block of mathematical concepts, knowledge and techniques.

As long as these concepts, theorems and techniques are new to the students they are a worthwhile subject of a curriculum. And, as long as the algorithm based on these concepts, theorems and techniques is not routine for the student, hand calculations in examples is a necessary and worthwhile activity during education.

However, at the stage where integration theory and algorithmics is taught, algorithmic prerequisites from earlier areas as, for example, partial fraction decomposition, squarefree factorization ... should not be the subject of hand calculation but a symbolic software system should be used for this.

3.3 Details on the Second Stage

In the second stage, where area X has been thoroughly studied, when hand calculations become routine (or intractable), area X is "trivialized" and can be the basis for higher areas X^*, ... It does not make sense any more to spend time for assignments covering examples from area X. Rather, in the future, problems from area X appearing as subproblems in algorithms for problems from higher areas X^*, ... can be left to a symbolic software system that can treat problems from area X by a black box algorithm (hopefully, one of the algorithms that have been completely studied earlier when discussing area X).

Thus, for each area X of mathematics, we propose to proceed in two steps. First study the area in depth, i.e. lay the basis for the algorithms of the area and study the algorithms as "white" boxes. Then (implement the algorithms in a symbolic software system and) use the algorithms as black boxes. This two-step procedure could drastically expand the scope of topics, insights and techniques in math curricula. Much more problem solving power could be gained, much more interesting problems could be attacked and, at the same time, more profound mathematics can be presented.

Example 2 (Integration) As soon as integration is "trivialized", an integration algorithm can be used as a black box in higher areas as, for example, the area of differential equations. The study of the area of differential equations, essentially is the study of possibilities to reduce the integration problem for differential equations to other, hopefully simpler, problems as, for example, the integration problem for (elementary transcendental functions), the problem of solving implicit functions, the problem of solving algebraic equations etc. (Essentially, the "results", i.e. theorems, in an area of mathematics can always be viewed as "problem reduction techniques".) When studying differential equations it is inappropriate and impeding to spend time for function integration subproblems. Conversely, the availability of integration software can drastically improve the potential to study interesting differential equation problems.

3.4 Recursive Application of the Principle

Note that the White-Box/Black-Box Principle is "recursive" and can govern math education in all levels and areas of mathematics including the elementary first steps in mathematics in high school as well as the most advanced areas in university mathematics that are not yet seriously attacked in an algorithmic spirit.

Roughly, the recursion runs as follows:

```
trivialize area ($X$):
    if area $X$ is not thoroughly understood
        then study area $X$;
    apply algorithms from area $X$ as black boxes.

study area ($X$):
    clarify (define) concepts from area $X$;
    clarify and specify problems from area $X$;
    study (prove) theorems (techniques) from area $X$
        that reduce problems from area $X$
        to problems from areas $X'$, ...;
    cast the reduction techniques in algorithms
        (or, at least, heuristics);
    train the algorithms as white boxes
        by doing sufficiently many hand calculations
        using algorithms from area $X'$, ... as black boxes
    until the algorithms become routine;
    trivialize areas ($X'$, ...).
    {In this stage area $X$ is "thoroughly understood".}
```

Note also that this recursive procedure can be applied, as any recursion, top down or bottom up. The bottom up usage seems to be the one normally applied in courses. However, it is certainly also possible to apply the procedure top down and present, for example, an integration course without having studied partial fraction decomposition, squarefree factorization etc. in depth. For such an approach, the availability of symbolic math software systems is essential because, in order to compute examples, i.e. in order to train the reduction techniques of integration, we need (but need not understand) lower algorithms as, for example, partial fraction decomposition. Hence, symbolic math systems open entirely new avenues for math curricula (including the ones for the "users" of mathematics as, for example, engineers):

In the attempt to make the distinction between the white-box and the black-box phase clear, I was a little too Puritan. It is clear that in the phase where, in the above "recursive didactic algorithm", concepts and problems of area X should be clarified it may often be very helpful to experiment with a symbolic system using the algorithms from area X *before* they are understood. In fact, seeing typical problem solutions by using a symbolic system may drastically facilitate the understanding, on the student's part, of what the actual problem really is. However, this use of symbolic systems in the white-box phase should not be misunderstood as an ecouragement to skip the detailed and thorough mathematical treatment of area X. (I acknowledge the discussion with Paul Zorn on this important use of symbolic systems in the white-box phase.)

4 Impact for Symbolic Math Software Systems

In order to realize the didactic principle explained in the preceding section, symbolic math software systems need some features that are not widely available in the present systems:

- A possibility to use an algorithm as a "black box" (this is the way algorithms are used in the present systems) and as a "white box", i.e. in a step-by-by step mode, exhibiting the important outer-most structure in which the reduction of the problem to subproblems is exhibited.
- More facilities to support "hand calculations" that go through the algorithms as white boxes and allow to use subalgorithms as black boxes.
- More possibilities for graphical and other auxiliary mechanisms to trace algorithms and visualize the steps in hand calculations.
- Explanatory, interactive tutorials on algorithms.

Much progress has been made in the design and user-friendliness of symbolic software systems. Still, for the didactic purposes described in this note, much additional work has to be done for improving the man-machine interface of these software systems.

More importantly, a systematic study of high school and undergraduate university mathematics has to be undertaken in order to exhibit the realization of the "white-box/black-box principle" in all concrete areas of mathematics.

Finally, systematic experiments at all levels of math education must be carried out in order to try out the new didactic principle and the new features of symbolic software in the classroom context. These experiments will produce the feedback for further improvements of the didactical and methodical principles and the software systems.

A research project encompassing these research goals is currently organized at RISC-LINZ.

5 References

The principle presented in this little note has its origin in a discussion at ICME 5 (International Congress for Mathematical Education), Adelaide, 1984. At this congress, I was asked to organize and chair a working group on the use of symbolic math systems in math education. The discussion in the working group went on for one week and basically saw two nearly disjoint groups of people who advocated one of the two extreme answers described in Section 2 with myself and a few others having the faint but not very pronounced feeling that the truth must lie in the middle. At the end of the week we, finally, saw clearly that the easy principle presented in this note is a common and natural basis on which we could all agree.

Although we cast the outcome of our discussion in a pamphlet, see SIGSAM Bulletin (1985), I detect now that the same discussion we had in 1984 (and, most probably, others had in earlier years) is replayed over and over again at various occasions. In fact, when reading again our 1984 pamphlet, I became aware now that the pamphlet is not specific enough on the White-Box/Black-Box Principle presented in this note. Therefore I thought that I should make the principle here as explicit as possible and that this could help to avoid repetition of the boring parts of the discussion and to promote the discussion towards realizing the principle in curricula and symbolic math software systems.