MATHPERT: An Expert System for Learning Mathematics

MICHAEL J. BEESON
DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE
SAN JOSE STATE UNIVERSITY
SAN JOSE, CA 95192
beeson@mathcs.sjsu.edu

Abstract[1]

MATHPERT is an expert system in mathematics designed explicitly to support the learning of algebra, trigonometry, and first semester calculus. This paper will describe what MATHPERT can do and how it is to be used by students. The internal design of MATHPERT will be described elsewhere. MATHPERT is intended to be used either alone or as a basis for more explicitly tutorial programs (which are not described here).

MATHPERT is an expert system in that it is capable of solving (almost) all problems in the stated subject domain internally and autonomously.[2] In this respect it is similar to existing computer algebra programs such as MACSYMA, MAPLE, or *Mathematica*. However, it differs from them in several respects, which are not only important but vital for education. The most basic of these differences are that MATHPERT is *glass box* and *cognitively faithful*; these terms are explained in the paper. MATHPERT produces not just "answers", but full step-by-step "solutions", with each step intelligible to the user and justified on-screen.

MATHPERT also incorporates a fairly sophisticated *user model*, and uses it to produce step-by-step solutions at a level of detail custom-tailored to the knowledge of the individual user.

MATHPERT is based on the analysis of the stated subject areas into several hundred *operators* which can be applied to mathematical expressions. When MATHPERT is operating in "menu mode", the user chooses which operator to apply next. The computer carries out the actual application of the operator. Operators which are "well-known" to the student, according to the student model, will be applied automatically, allowing the student to focus on the less well-known parts of the problem. At any time MATHPERT can be switched into "automatic mode", in which it will not only apply the operator but choose it, thus generating one (or more) steps of the solution automatically. The user can return to "menu mode" at will.

MATHPERT accepts arbitrary symbolic problems from the user; for example, a student might type in her homework. MATHPERT is designed for use with existing courses; whether or not the class is officially using MATHPERT, an individual student should be able to use MATHPERT beneficially. It is designed to be useful to students across the spectrum from those needing remedial work to the very brightest students.

MATHPERT includes a (two-dimensional) graphing package capable of every kind of two-dimensional graphing that might be useful for students, including facilities for rapidly changing the value of a parameter or the portion of a graph visible on the screen.

MATHPERT is written in Prolog and C, and runs on an IBM PC/AT with an expanded memory board and EGA or CGA graphics.

Glass Box and Cognitively Faithful

An expert system is called *glass box* if you can see how it arrives at its answer. MATHPERT can print out the individual steps of its solution, with their justifications. (We use the word "solution" to mean such a sequence of intelligible steps, whose last line is the "answer".)

An expert system is *cognitively faithful* if its own internal solutions correspond to the solutions a human would produce. MATHPERT solves math problems in the way we teach students to do, rather than using high-powered algorithms. Cognitive fidelity must be designed for from the beginning, as the demand for cognitive fidelity complicates the construction of an expert system considerably.[3]

[1]This work partially supported by NSF.

[2]The range of MATHPERT's capabilities is extensive. For example, it can solve problems in simplification, including all kinds of exponents and radicals, factoring, equation solving including transcendental equations, trig identities, limits, differentiation, and integration.

[3]The term *glass box* is in the literature, e.g. in Anderson [1988], Burton and Brown [1982] (where it is credited to a 1977 memo of Goldstein and Papert). The term *cognitively faithful* has probably been used, too: certainly the *concept* appears in Anderson [1988] and in Wenger [1987] (there under the name "psychological plausibility").

MATHPERT also incorporates an elaborate internal *user model*, or *student model* (but there may well be non-student users). This model contains (among other things), the information concerning which of several hundred pieces of knowledge are, for this user, *well known*, *known*, *learning*, or *unknown*. MATHPERT uses the model to tailor its output to the user. A naive user will receive more detailed explanations than a sophisticated one, and in particular ways tailored to the exact knowledge of that user; a generally sophisticated user with some gaps in her knowledge will still receive detailed explanations when her weak points are involved. This use of the student model to modify output results in MATHPERT's being "cognitively faithful" not just to some idealized student, but to the particular, individual user with whom it is dealing at the moment (provided, of course, that the internal user model is accurate).

The Operator View of Mathematics

MATHPERT depends on an analysis of its subject matter (algebra, trigonometry, and elementary one-variable calculus) into several hundred *operators* which can be applied to mathematical expressions. For example, one operator is called **collect powers**. It applies to an expression x^2x^9 and produces x^{11}. The key to the solution of a mathematical problem, according to this view, consists in a correctly-chosen sequence of operators which are to be applied to the input. The "solution" itself is the line-by-line record of the result of these operator applications, together with their "justifications". The justifications are simply the names (or formulas) describing the operators applied.

MATHPERT operates in two "modes": in *menu mode*, the user directs the course of the developing solution by choosing the next operator from a system of menus. Since there are several hundred operators, it would not be practical to require the student to remember and type the names of the operators.[4] The menu system has been designed to show only those operators which might be relevant to the problem at hand, so that usually the student does not have to leaf through all four hundred operators looking for the right one. Moreover, even in menu mode, "well-known" operators will be applied automatically. Thus while doing integration by parts, for example, you need normally not search for the operator $-(-a) = a$, which should be well-known long before you are tackling integration by parts.

In *automatic mode*, MATHPERT will generate its own "ideal solution", step by step. The user can switch at will between automatic and menu mode. Thus you can start in menu mode, get stuck, switch to automatic mode for one or two steps for a hint; then, back on the track, you can continue in menu mode. When you choose "finished", MATHPERT will switch into automatic mode and see if you really are finished, according to its own internal algorithm. If not it will supply the last steps of the problem. If you switch into automatic mode and stay there, even at the beginning of the problem, MATHPERT will generate the complete solution for you. Thus in principle you could just type in your homework and have MATHPERT print out complete, step-by-step solutions to each problem.

Although automatic mode generates a single "ideal solution", menu mode permits "alternate solution paths": any correct sequence of operators will be accepted. At any point you can switch into automatic mode and MATHPERT will successfully complete the problem from that point, if possible.

Implications of the Operator View

The view of mathematics as consisting in the application of operators to expressions leads to the classification of mathematical knowledge into four kinds:

• *Knowledge Of or Knowledge That.* This consists in knowing what the effect of a given operator will be, and in knowing that this operator exists and can be applied when needed. For example, knowledge that $\sin^2 x + \cos^2 x = 1$.

• *Clerical Knowledge.* This consists in knowledge of how to apply a given operator. The boundary line between knowledge of what an operator does and the ability to apply it is hard to observe directly when students work with pencil and paper; if they fail to collect powers in x^2x^9 you can't be sure why. However, MATHPERT takes over the clerical function for the student, leaving him (at least in menu mode) the responsibility to know (or discover!) the effects of operators.

• *Knowledge How.* This consists in the procedural knowledge of how to choose operators in the order necessary to solve a given problem. Very little explicit attention is given to this "control knowledge" in traditional

[4]Besides, students are generally poor typists. In MATHPERT, they will have to type only to enter their problem, not to solve it.

instruction, yet it is very important. MATHPERT permits the student to focus more attention on "knowledge how". In theory the correct sequence of operators could be chosen by some general planning process, based on the knowledge of the effects of operators. In practice students succeed in mathematics only when they learn a procedure for solving a given type of problem.

• *Knowledge That.* This consists in knowing the reasons why the operators are correct. For example, why is $\sin 2\theta = 2\cos\theta\sin\theta$ a correct operator? Why is collecting powers a legitimate operation? MATHPERT makes no contribution to help the student learn this kind of knowledge; the operators are just "given". However, the architecture of MATHPERT makes it easy for tutorial software running on top of MATHPERT to tutor students in "knowledge that". Note that "knowledge that" is almost never taught to students in conventional classes at the levels MATHPERT covers: although it may be mentioned in passing, it is never on the tests. Only the more elementary three kinds of knowledge are tested.[5]

Skills and Concepts

The guiding philosophy of my work on AI-based educational software for the personal computer is to make software that will help students in traditional courses. These courses are primarily skill-based, and I believe for good reason: Concepts grow, in my opinion, out of the use and mastery of skills, and not the other way around. This dictates the design of skill-based software as the starting point for computer-based education.

Let me offer an operational definition of "skill" and "concept", which I believe is helpful in clarifying these issues: How does one test whether a skill has been learned? By asking problems of the same sort used during instruction. How does one test whether a concept has been learned? By asking problems of a different sort from those used during instruction, but to which the concepts are applicable. If the student can solve only problems which follow a very strict template, then a skill has been learned.

Informal discussions with colleagues have made it clear that, to the contrary, many people feel the computer should be used to teach concepts, "liberating" students from the demands of skill-based learning. This is an issue that deserves a long discussion, which space here does not permit. I have only space to emphasize that MATHPERT has been designed on purpose as a tool for the learning of skills, not concepts; although it has been done in such a way to support tutorial software in the future which might teach concepts.

How will MATHPERT *improve learning?*

MATHPERT will relieve people of the clerical burden of mathematics, just as word processing relieves people of the clerical burden of writing. Instead of having to concentrate on the clerical details, people will be free to think about what they want to do to solve the problem.

This is, however, not the only effect of using MATHPERT. An important feature of MATHPERT is that it does not allow an incorrect step. It *will* allow "red herring" steps, which do not really lead in a useful direction, but are at least mathematically correct. But the fact that you can't really arrive at an incorrect formula with MATHPERT should prevent a lot of headaches with mathematics. When you realize that your "red herring" steps have led you "on a wild goose chase", you can use the "undo" facility to back up to where you went wrong.

Another way in which solving problems with MATHPERT differs from solving them by pencil and paper is this: MATHPERT's menu system permits you to peruse your possible moves at any time when you feel puzzled as to what to do. Simply seeing the possible operators laid out systematically before you will help you choose a useful one. Moreover, repeatedly seeing the operators while working problems is bound to make it easier to remember the range of possible choices.

Much previous work on ICAI (intelligent computer-assisted instruction) has focussed on the "buggy model of learning",[6] in which the computer is supposed to diagnose the student's incorrect ("buggy") operators and direct specific tutorial action to correcting these misconceptions. Generally speaking, the "buggy model of learning" will not be very useful with MATHPERT, since MATHPERT does not permit the student to take an incorrect step. The student will become disabused of buggy notions without the necessity of diagnosing these bugs. To put the matter another way: the "buggy model" describes learning in a traditional environment.

[5]Note also that justifications of operators are of two kinds: some operators, such as $\sin 2\theta = 2\sin\theta\cos\theta$, can be justified in terms of simpler operators. Others, such as the sum formula for sin, require a justification outside the framework of "operators".

[6]See e.g. Burton [1982] for an explanation and detailed example of the buggy model; and Matz [1982] for a discussion of the buggy model applied to high school algebra.

We do not believe it will describe how learning takes place in an environment based on MATHPERT. In this environment, we will "nip bugs in the bud ". (Perhaps we should not mix metaphors: we will nip bugs in the larval stage.)[7]

A better theoretical foundation for learning in the MATHPERT environment is Maria Montessori's principle that educational materials should have "control of error", i.e. the student working alone with the materials should not be able to go very far wrong: the square peg won't fit the round hole.[8] Although Montessori had very young children and physical materials in mind, her principle (which for lack of space we have not fully explained) has much wider applicability.

The User Model in MATHPERT

The operators used by MATHPERT have been carefully chosen so as to correspond to cognitive skills, that is, to identifiable "chunks" of knowledge which can be taught and learned. Thus the "skills lattice" which is used in cognitive science to model learning can be directly correlated to a user model based on the executable operators of MATHPERT . Each operator is at the same time a procedure and a skill.

MATHPERT's internal model of its user consists in recording, for each of some four hundred operators, one of the values *learning, known, well-known*, or *unknown*. This section will describe how these values are used in MATHPERT and how they will be used by future software.

The values *learning* are used by individual operators. For example, we stated above that the operator **collect powers** will produce output x^{11} on the expression x^2x^9. This is not strictly true: if the student is still on record as "learning" this operator, it will produce instead x^{2+9}, after which the operator **arithmetic** will be automatically applied (if *it* is well-known) to produce x^{11} on the next line. A substantial fraction of MATHPERT 's operators are designed in this way, to produce more explicit versions of their output when certain operators are still recorded as "learning".

The values *well-known*, on the other hand, are used not by individual operators, but by the main MATHPERT system. Even in menu mode, well-known operators are applied automatically. This lets the user concentrate on the task at hand, relieving her of the necessity to search through the menus for an operator she learned two years ago and knows very well how to use. However, these operators will still be visibly applied on the screen. This provides the best of both worlds: the student does not have to think about applying $-(-a) = a$, but also can see explicitly that it was applied. Otherwise, the application of two or three well-known operators (invisibly) can result in confusion.

The values *known* and *unknown* are not explicitly used in the present version of MATHPERT, but are maintained to allow for the possibility that tutorial software may want to "grey out" unknown operators so that the student can't see or use them. This will be used particularly in the (not uncommon) case that one operator combines the skills represented by several simpler operators. For example, the operator **common denominator** is broken into five or six simpler operators intended for use while learning common denominators.

Evidently the user model is only useful if it is accurate. By definition, the process of *diagnosis* is *initializing and updating the student model.* A program called DIAGNOSER will initialize the student model interactively at the student's first serious session. DIAGNOSER will generate problems dynamically, based on the student's previous responses; since there are about four hundred operators, dynamic generation rather than a pre-stored test is necessary. At present (9-15-88), DIAGNOSER is in the design stage. Closely related to DIAGNOSER will be a program called EVALUATOR which will analyze the student's performance with MATHPERT and decide on the correct updating of the user model. At present EVALUATOR is in the early design stage, and we shall not comment further on it. DIAGNOSER and EVALUATOR are both components of an expert system MATHTUTOR which we plan to construct as a companion to MATHPERT.

MATHPERT can be used now, without DIAGNOSER and EVALUATOR, if a human tutor (or the user herself) adjusts the student model appropriately to the level of the student in question. It need not be absolutely accurate to be useful. The student model can easily be adjusted through the menu system, without altering the program itself.

[7]Of course, that doesn't apply to bugs that are already full-grown. The architecture of MATHPERT does permit it to accept a "buggy operator" as a new operator and simulate the mistaken behavior of a student, should tutorial software based on MATHPERT wish to do so.

[8]Montessori [1917], pp. 74-75, explaining the concept of control of error in the design of materials: "it is not enough that the stimulus should call forth activity, it must also direct it. The child should not only persist for a long time in an exercise; he must persist without making mistakes."

Graphics in MATHPERT

MATHPERT provides two-dimensional graphics; facilities are provided for graphing ordinary functions, parametric functions, and functions in polar coordinates. You can compare two functions on the same graph or on different graphs; you can choose the size, position, color, and other parameters of your graphs. You can pass from the symbolic manipulation part of the program to the graphics at will, for example first differentiating a function and then graphing it and its derivative on two graphs one above the other.

You can graph a formula containing a symbolic parameter, and change the value of that parameter at a touch of the '+' or '−' keys. You can "zoom" by a factor of two in the x or y direction by touching the arrow keys, or by several factors of two at once. In this way you can, for example, compare x^n with 2^x, alternately increasing n until x^n dominates, then zooming until 2^x dominates again.

The graphics of MATHPERT, however, is not cognitively faithful. There is no attempt to get the computer to sketch graphs as we wish students to do. It simply draws the graph in the usual computerized way.

Comparison of MATHPERT *with Mathematica*

These days one finds considerable enthusiasm for the application of computerized mathematics systems such as the new *Mathematica* (Wolfram [1988]) and its predecessors *Maple* and *MACSYMA* to mathematics education, particularly calculus. These applications fall into three categories, which largely correspond to the three major categories of abilities of *Mathematica*: computer graphics, symbolic manipulation, and numerical computation. The purpose of this section is to show that MATHPERT and *Mathematica* have different things to offer; one should not think of them as similar packages.

Graphics and numerical computation in MATHPERT are not different from graphics and numerical computation elsewhere, although the interfaces to them have been designed for education.

It is in the realm of symbolic computation that MATHPERT differs from *Mathematica* and its predecessors. *Mathematica* is similarly based on the concept of applying operators to expressions, but the operators in *Mathematica* do not represent skills that are taught to students. Cognitive fidelity is (intentionally) missing from *Mathematica* for two distinct reasons: First, the internal algorithms are not those taught to students, but rather the finest, fastest, most powerful algorithms known. *Mathematica* can factor $x^n - y^n$ for any value of n; MATHPERT, like the student, can only handle a few special values, and in many cases (for large n) can factor it only partially. This limitation of ability is a necessary correlate of cognitive fidelity.[9] Second, the operators provided with *Mathematica* are too powerful for use in unmodified form in education. For example, *Factor* is a single operator: it does not distinguish the many methods of factoring, and does not indicate the method or steps it uses.

References

Anderson, John R. [1988], The Expert Module, in: Polson, M. C., and Richardson, J. J. (eds.), *Foundations of Intelligent Tutoring Systems*, pp. 21-54, Erlbaum, Hillsdale, N. J. (1988).

Burton, R. R. [1982], Diagnosing bugs in a simple procedural skill, in Sleeman and Brown [1982], pp. 157-185,

Burton, R. R., and Brown, J. S. [1982], An investigation of computer coaching for informal learning activities, in Sleeman and Brown [1982], pp. 79-98.

Matz, M., [1982], Towards a process model for high school algebra errors, in Sleeman and Brown [1982], pp. 25-50.

Montessori, M. [1917], *Spontaneous Activity in Education*, reprinted by Schocken, New York (1965). (Page numbers in the text refer to the Schocken edition.)

Sleeman, D., and Brown, J. S. [1982], (eds.), *Intelligent Tutoring Systems*, Academic Press, London/ Orlando, Fla. (1982).

Wenger, E. [1987], *Artificial Intelligence and Tutoring Systems*, Kaufmann, Los Altos, Calif. (1987).

Wolfram, S. [1988], *Mathematica: A System for Doing Mathematics by Computer*, Addison-Wesley, Redwood City, Calif. (1988).

[9] Of course, one could always include an "oracle operator"— "appeal to methods beyond the scope of this course", which would result in the use of high-powered algorithms.