Intelligent Help at Every Step: Critically Important, Not Difficult, Yet Surprisingly Rare

John C. Miller
Mathematics Department, The City College of CUNY
110 Riverside Dr., #14C, New York, NY, 10024
xyalgebra@mindspring.com

A. Math Learning Modes

At any given time a math student is usually in one of two readily identifiable modes.

- In **Presentation Mode** the student views or reads "presentations" by the instructor, software or textbook stating and explaining new principles, axioms, definitions, theorems, techniques, skills or ways of applying previous material.
- In **Problem Solving Mode** the student attempts to solve problems presented by the instructor, software or textbook.

A conspicuous difference between the two modes is that Presentation Mode permits the student to remain comparatively passive since the initiative lies elsewhere. Problem Solving Mode, by contrast, requires continuous active student involvement.

B. The Problem With Instructional Math Software: Workbook Emulation

Most students learn very little from presentations of new material. As a thought experiment, consider the likely result of routinely quizzing students on the new material of the day immediately after it is presented, before posing any practice problems. With few exceptions **the real learning essentially begins when students begin solving problems for themselves.**

Commercially available instructional mathematics programs often provide excellent presentations of new material. They exploit the multi-media features of modern computers to demonstrate mathematical ideas, principles and techniques in ways that are often superior to, and impractical for, a blackboard or textbook presentation. Unfortunately, when students commence solving problems for themselves, short answer and multiple-choice problem formats are nearly universal. If a student enters or selects a final answer that the program deems incorrect, the help available usually consists, at best, of a pre-stored solution to that problem done by the method preferred by the program designer. For non-trivial problems an incorrect short answer is usually insufficient to infer either the student's method of solution or the student's errors. Intelligent help is impossible in principle for most short answer problems. "Workbook Emulation" aptly describes the short answer and stored solution format prevalent in current mathematics software's problem solving mode.

C. Intelligent Help at Every Step

A properly programmed computer can display a much higher level of intelligence than a workbook. Intelligent help at every step should be the goal. The computer should:

- allow solutions to be entered one step at a time,
- detect syntax errors immediately, in mid-step,
- support simple and intuitive editing,
- evaluate each step as correct or incorrect using criteria that accept any correct and reasonable solution method,
- suggest or hint at a reasonable next step on demand,
- revise suggested solution strategy if a student adopts a non-standard strategy,
- help the student finish any correct partial solution to any practice problem.

Go to www.xyalgebra.org/Description/Intelligent_Help/intelligent_help.html (case is significant) for an on-screen example of intelligent help in solving an equation. Go to www.xyalgebra.org/Download/xyDemo.exe (case is significant) for a downloadable example of intelligent help in solving a verbal problem.

D. Expression Trees: The Key to Intelligent Help

Expression trees are key to a computer's ability to provide intelligent help at every step. Every syntactically correct algebraic expression has an expression tree. See Figure 1 below, from which the concept should be obvious. Note that the structure of the tree specifies the order of operations without requiring parentheses.

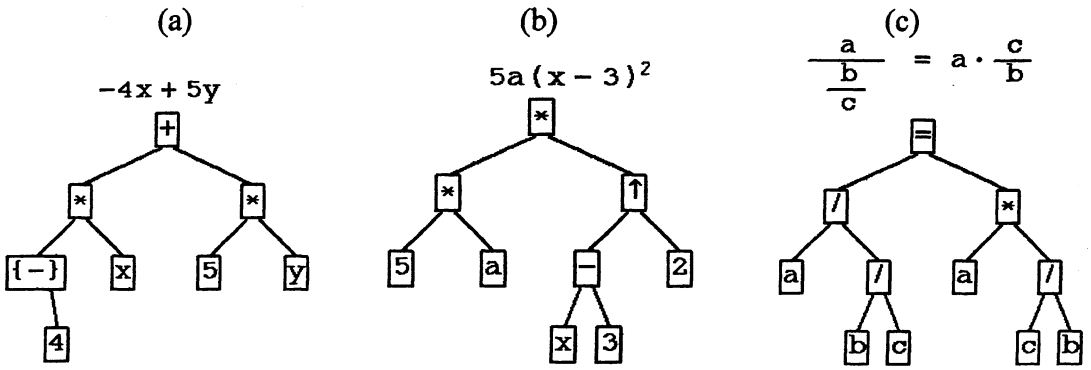

Figure 1. Examples of Expression Trees

Trees have many applications in computer science. Of course, trees as depicted here are graphical representations of "trees" as stored in a computer. The details of the computer representation need not concern us. Algorithms for generating the tree of an algebraic expression and the algebraic expression of a tree are well known. Both algorithms use the rules for order of operations in algebraic expressions. We will demonstrate the first of these algorithms below.

Numerical evaluation is a simple but useful application of the expression tree of an algebraic expression. Given a numerical value for each variable appearing in an expression, the algorithm for evaluating it (or its tree) is:

- The value of a constant node is the constant.
- The value of a variable node is the value of the variable.
- The value of an operation node is the result of applying that operation to the value(s) of its subnode(s) in left-to-right order.

A simple recursive subroutine based on this algorithm suffices to evaluate an algebraic expression of any complexity.

E. Determining Whether a Step Is Correct

Numerical evaluation suffices to determine whether a step in most algebra problems is "correct," i.e., equivalent to the previous step in the sense appropriate to that type of problem. The details should be obvious upon reflection. For a verbal problem step in which an expression is said to represent a described quantity, evaluation combined with simple substring matching, with synonyms, suffices to determine correctness.

F. Suggesting a Reasonable Next Step

The field axioms and the other basic rules of algebra all correspond to straightforward tree modifications. Figure 1(c) above illustrates the "invert and multiply" rule. It applies whenever a quotient node has a quotient node as its right (denominator) subnode. To apply it, change the first quotient to a product and interchange the subnodes of the second quotient. This applies anywhere in a tree of any level of complexity, as illustrated in Figure 2, below. Human pattern recognition skills make it evident where to invert and multiply in the expression in Figure 2(a). For a computer, the easy way to "see" this is to search the corresponding tree recursively for adjacent quotients, circled in Figure 2(a). Then modify the tree as described above and generate the corresponding expression shown in Figure 2(b). Figure 2 also suggests how tree manipulation permits separating all issues involving parentheses from the process of applying algebraic rules.

(a) Before inverting and multiplying (b) After inverting and multiplying

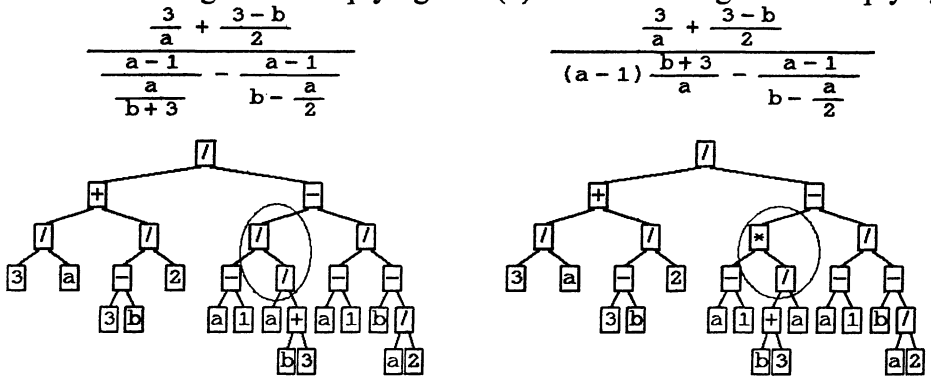

Figure 2. Applying the Invert and Multiply Rule at the Circled Location

The complicated part of suggesting the appropriate next step is then simply to decide which of several applicable rules of algebra a competent instructor would suggest next. This is a non-trivial but ultimately straightforward issue.

The suggested next step routine must also become more ambitious or "smarter" as the student progresses. Operations that would initially be done in several small explicit steps should later be done automatically. For example, after a few practice problems, both associative laws should be applied automatically to avoid unnecessary parentheses.

In verbal problems, the recommended next step can be determined by a "shortest route" strategy. Suggest the shortest route from whatever relevant quantities currently have correct expressions available to any of the several relationships among the quantities in the problem that can possibly give rise to equations. Use evaluation to verify that the resulting equation(s) are independent, that is, have non-zero determinant.

G. Allowing Entry of Arbitrary Steps with Instant Feedback on Syntax Errors.

Support for responsive keyboard entry of arbitrary steps is perhaps the most difficult of the requirements for intelligent help at every step. Algebraic expression entry must be simple and intuitive, closely mimicking the way expressions are written on paper in both appearance and character entry sequence. Students unaccustomed to mathematics on computers must learn to enter steps easily and quickly.

Syntax errors should be reported immediately, not at the end of a step. Constructing the expression tree as a student enters an expression simplifies syntax error detection since such errors cause easily detected aberrations in the tree construction process.

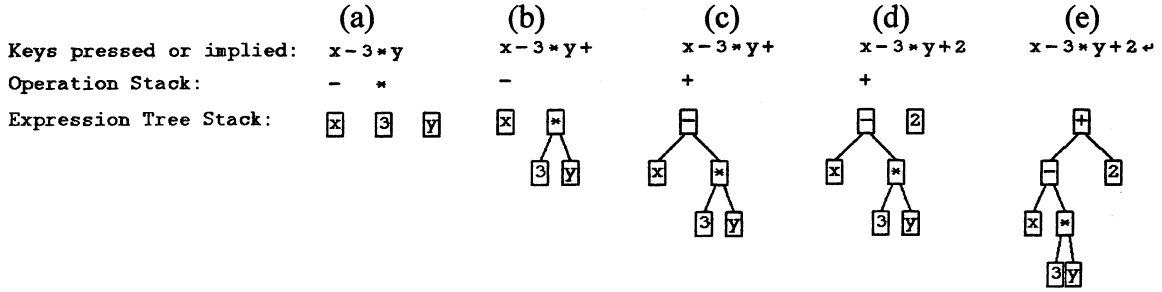

Figure 3. Constructing the Expression Tree of $x-3y+2$ As It Is Typed

In Figure 3 above the student presses "$x-3y+2$" and the Enter key. Temporary storage consists of two last in, first out stacks, an operation stack and an expression tree stack. Place all constants and variables on the expression tree stack as they are typed. Place an operation on the operation stack only if it has higher priority than the top operation already on the stack (if any) according to the rules for order of operations in algebra. Figure 3(a) shows x, 3 and y on the expression tree stack and the subtraction and implied multiplication on the operation stack. Then the student presses "+." Addition is not of higher priority than either of the entries already on the operation stack, so they are

removed. Figure 3(b) shows the use of the multiplication from the operation stack as the root of a product tree with the top two trees from the expression tree stack as operands. Figure 3(c) shows the same for the subtraction, followed by placement of the addition on the operation stack. Figure 3(d) shows the 2 placed on the expression tree stack. Figure 3(e) shows the use of the addition from the operation stack as the root of the overall tree constructed from the remaining two trees on the expression tree stack after the Enter key was pressed.

In more complicated expressions, mark the stack at the start of each explicitly or implicitly parenthesized or bracketed subexpression and clear it back to that point at the end of that subexpression. This assures correct nesting of such subexpressions in the resulting tree and simplifies immediate detection of mismatched or improperly nested grouping symbols, such as an equal sign inside a numerator.

H. The Fault Lies Not in Our Publishers But In Ourselves

The blame for the routine failure of commercial math software to offer intelligent help in problem solving ultimately lies not with the publishers but with the adopters. If mediocre and unintelligent software is widely and uncritically adopted, there is no reason for publishers to produce better materials. Every faculty member at every institution using or considering use of instructional software shares responsibility for demanding that publishers offer intelligent support for students during the problem-solving phase of their learning. Materials lacking such support should be shunned.

Demands for more intelligent materials must be forcefully pressed at every opportunity:

- At publishers' brainstorming sessions.
- During campus visits by publishers' representatives.
- At commercial presentations in all venues.
- At commercial exhibits.
- In both internal and public reviews of materials.

Intelligent computer-based support of mathematical problem solving is neither mysterious nor difficult. The paucity of computer-based materials offering intelligent help on a step-by-step basis amounts to collective and inexcusable gross negligence and dereliction of duty by the entire mathematical community. As institutions and individual faculty rush to offer their courses over the Internet, the situation is predictably worsening. Faculty must vehemently protest the limitations of current materials and vigorously demand improved standards of intelligent and interactive problem solving support.

I. Intelligent Algebra Course Available Free of Charge

The author's web site, **xxx.xyalgebra.org**, offers a downloadable basic algebra course featuring thousands of practice problems with intelligent help at every step. These materials, which resulted from an academic research project, include an academic site license and may by used without cost by individuals or by academic institutions.