BENEFICIAL HIGH–STAKES MATH TESTS: AN EXAMPLE

FRANK QUINN

Abstract. A worked–out example is given to show how mathematical and educational insights can be incorporated in the structure of high–stakes K–12 math tests in a way that promotes more effective learning and better teaching practices. The example concerns symbolic skills deficits seen in students from calculator–oriented K–12 programs.

Contents

1. Introduction

High–stakes tests influence teaching and learning. When learning is poor they provide discipline and motivation for improvement. When learning is good the influence tends to be bad because the focus shifts from learning to test performance. Recent widespread introduction of high–stakes tests is, in effect, a judgement call: the general level of learning is so poor that the discipline enforced by tests will outweigh bad effects in the few previously–good cases. Roughly speaking we accept a cap on the top to get a floor under the bottom[1].

Date: November 2008.

[1]See The K–12 math test conundrum for a brief discussion

The thesis here is that high stakes tests—when very carefully done—can influence teaching and learning in positive ways. Counterproductive influences are the result of poor tests, not of high–stakes tests per se.

Beneficial high–stakes tests require a completely different approach to testing. Every aspect, from how the tests are given, to problem design, down to the level of computer code, must be driven by sophisticated educational and mathematical wisdom. But this wisdom must be correct in an almost mathematical sense, and in particular not determined by conventional wisdom or ideological convictions.

1.1. **Outline.** To illustrate this thesis we give an example worked out in detail.

- First we identify reasons that students taught with calculators have symbolic-reasoning deficits: for instance calculator routines conceal mathematical structure. But then we discover that by–hand arithmetic actually has the same problems to a lesser degree.
- We suggest a change in the way problems are worked—with or without calculators—that would address this.
- We describe a modification to test design that would make the new approach directly effective for test–taking. This provides motivation for teachers and students, though the motivation provided by a yearly test would be a bit remote.
- The envisioned test–generating system could also provide course tests and plentiful practice materials, adjusted to be appropriate for different levels and locales. This would give constant reenforcement and feedback and quickly spread improvements.
- We then discuss high–level issues in implementing such a test system. One is that it would spread mistakes just as quickly as improvements.
- For this and other reasons content for such a system must developed in an open and non–commercial way that can respond quickly to feedback and can draw on the wisdom of the entire community. Large–scale test *administration* might still be a commercial activity.
- This particular suggestion requires a change in the functionality of tests as well as in content. We describe how to use modern electronic formats and programming tools to achieve this.
- We hope to have sample tests with these features available at the Joint Mathematics Meeting in Washington DC January 4–8, 2009.
- Implementing this change would ideally lead to significant changes in the way K–12 math is taught, including systematic use of parentheses from the very beginning of arithmetic.
- The discussion of instructional changes also illustrates use of web reference materials for teacher support.

1.2. **Other Issues.** The discussion here is based on analysis of a single issue and many others will have to be similarly understood to get a complete picture. For instance here we see how to organize elementary material to *set the stage* for abstract and symbolic reasoning but have not tried to work out how such reasoning should be taught and tested. A few more examples:

- The analysis here focuses on single–step problems. How should multi–step problems be handled?

- Students taught with graphing calculators often have geometric–reasoning deficits. What is behind this and how can it be avoided?
- Calculators have inadvertently caused serious problems and we are only now beginning to recognize them and sort them out. Can we figure out how to graduate to modern computer–algebra systems without having to suffer through turbocharged versions of the same problems?

Fortunately these do not have to be tackled all at once. If our analyses are based on thorough and accurate understanding of mathematical structure then we can expect the solutions will fit together and reenforce.

1.3. **Web Resources.** Web resources can help explain and support instructional change. This is illustrated with links to the web site of the American Mathematical Society Working Group on Preparation for Technical Careers, abbreviated AMSTC.

2. Analysis of the Problem

A specific word problem is used to illustrate how calculators, and to a lesser extent traditional approaches, fail to support development of higher–level reasoning. The natural mathematical view suggests a remedy but also makes clear some of the difficulties that will be encountered.

2.1. **A Sample Test Question.** The example is:

Problem Three children collect acorns for an art project. Dick finds 7 acorns; Jane finds 13; and Warren amasses 40 acorns. The teacher puts all the acorns in a bowl and then divides them evenly among the three children. How many acorns does each child have for the project?

2.2. **Mathematical approach.** A mathematician would write $(7 + 13 + 40)/3$.

The structure of the situation is clearly reflected in the structure of the expression and it is a small step to write $(\,\#\,+\,\#\,+\,\#\,)/3$, where # is used as a placeholder for the numbers of acorns collected by a child. This abstraction is easily accessible and will have a subliminal influence even if it is not made explicit.

The generalization to an arbitrary number of children is conceptually easily but the placeholder notation is unsatisfactory because it does not display the linkage between the place and the child. To do this we use $\frac{1}{n}(A_1 + A_2 + \cdots + A_n)$. The underlying structure is clearly the same as for three children so the only difficulty is with the notation, and this should seem reasonable because it solves a problem (dangerous imprecision of the # formulation).

A more subtle feature of the notation $\frac{1}{n}(A_1 + A_2 + \cdots + A_n)$ is that the sum $(A_1 + A_2 + \cdots + A_n)$ is seen as a unit that can be manipulated even though the operation has not been carried out. The best version, $\frac{1}{n}\Sigma_{i=1}^{n} A_i$, builds on this and can be made accessible if students are explicitly taught how to parse it and read it out loud.

Finally the expression makes sense for any type A_* that can be added and then divided by an integer. Students who think of polynomials as a fancy sort of numbers (as do mathematicians) will use exactly the same expression to find the average of a collection of polynomials.

2.3. **Calculator approach.** The student presses keys 7, +, 13, +, to get 20, then 40, +, to get 60, then ÷, 3, =, to get 20.

This is an *algorithm* rather than an expression. Students easily see how to generalize it to handle more cases but it does not explicitly display the mathematical structure and cannot be generalized or manipulated as an expression. Further there is no notation for these algorithms so they must be remembered rather than recorded. This makes it difficult to point out structural similarities in work done at different times, or, better, have students recognize similarities because the expressions have the same structure.

Calculator–trained students have to see polynomials as new and different things. The structural similarity between integers and polynomials has been hidden: numbers are algorithmically manipulated by calculator while symbols require rules that seem strange and tedious because they have not already been internalized, for instance through by–hand arithmetic. These students will have difficulty seeing any similarity or connection at all between the solution of the acorn problem and the average of a set of polynomials.

2.4. **Traditional approach.** Traditional students will write $7 + 13 + 40$, but then encapsulate this as a unit by carrying out the operation rather than with parentheses. They then divide the sum, 60, by 3.

We now see the traditional approach as half–way between the mathematical and calculator versions.

On the plus side the sum is seen as a unit as in the mathematical version. It can be generalized to $A_1 + A_2 + \cdots + A_n$ and then to $\Sigma_{i=1}^{n} A_i$. By–hand arithmetic provides a lot of hands–on experience with mathematical structure (of addition and multiplication) so it has been internalized and this makes the transition to symbols and polynomials relatively easy. After this the sum makes sense for polynomials and other symbolic expressions.

On the negative side the full solution is still an algorithm rather than an expression: division by 3 takes place after encapsulating the sum by evaluation rather than with parentheses. The unevaluated sum is not presented as an object that can be manipulated. Students learn to approach problems by alternating organization (setting up the sum or the division) and operations (adding, dividing). This works in school math because problems are designed to be worked this way but is counterproductive in the long run because it disrupts mathematical structure and invites errors.

3. Diagnosis and a Remedy

We saw in the previous section that calculator–oriented math education thoroughly mixes the organizational and computational components of problem–solving, and that this undercuts learning in a number of ways:

- It does not provide unevaluated arithmetic expressions that display mathematical structure, provide templates for generalization and abstraction.See AMSTC/Products of Sums for another example.
- These unevaluated expressions also aid in diagnosis of mistakes, see AMSTC/Diagnostic Aids.
- It hides the functional similarity of numbers and symbolic expressions such as polynomials.

- Mixing cognitively different tasks degrades both and increases error rates, see AMSTC/Separation of Tasks.

We then saw that the traditional approach with by–hand arithmetic actually has some of the same deficiencies and therefore also undercuts learning albeit to a lesser degree. In other words calculators did not cause the current abstract and symbolic reasoning deficits, but their use enabled expansion of bad practices that worsened deficits that had been invisible.

One conclusion is that "go back to the old ways" is not a satisfactory solution: we need a new approach that exploits this new understanding. For instance complete comfort with parentheses seems to be vital but traditional elementary math education has a parenthesis phobia, see AMSTC/Parentheses.

3.1. **Remedy.** The proposal is for teachers and tests to encourage students to separate the organizational and computational components of problems by explicitly using unprocessed intermediate expressions.

To clarify this we give an example.

3.2. **Better Sample Question.** We revise the word problem of §2.1 to illustrate how the remedy might be implemented in a test, and give a variation for class use.

3.2.1. *Test Version.* Three children collect acorns for an art project. Dick finds 7 acorns; Jane finds 13; and Warren amasses 40 acorns. The teacher puts all the acorns in a bowl and then divides them evenly among the three children. Give *an arithmetic expression* that evaluates to give the number of acorns each child has for the project.

Problems like this must be explained and used with care:

- "Raw–output" expressions are not well defined, and there will be a great many correct expressions, including the numerical outcome (20), that are logically correct.
- For (machine) scoring purposes the expression is considered correct if it evaluates to give the correct outcome. In other words the student only has to set it up and the test will take care of the arithmetic. In §4.2 we suggest taking this literally: when an expression is entered the result of evaluation is automatically displayed.
- The point is that the student's best strategy is to do only the organizational component of the problem and enter the resulting expression without doing any processing. This minimizes time and exposure to errors.
- The student's best strategy therefore implements the proposed remedy: organization and processing are separated and attention is focused on the intermediate expression where structure etc. is displayed.

The test formulation is not immediately suitable for use outside a computer test environment because the unevaluated expression is not well–defined and cannot easily be checked for correctness. A class version could be:

3.2.2. *Classroom Version.* Three children collect acorns ...

(1) Set up an arithmetic expression that gives the number of acorns each child has for the project, but don't do any arithmetic.
(2) Evaluate this expression.

An answer would be considered correct if the response to (1) is not obviously bogus and has the right structure (in this case something like $(\# + \# + \#)/3$), and the number in (2) is correct. Otherwise it is considered incorrect:

- If the number is incorrect then the expression can be considered more carefully. A correct expression indicates that there was a mistake in evaluation and more evaluation drill may be called for. An incorrect expression suggests an error in setup.
- Determining that the form of an expression is wrong relies on human pattern–recognition skills. The instructions to students is that this part of the answer is to be used for diagnosis when something goes wrong; see AMSTC/Diagnostic Aids. If the expression is unsuitable for diagnosis then the answer is unsatisfactory even if the number is right.

This last point goes against the idea that a correct number justifies everything, but requiring an organizational and diagnostic step really is important. In particular I believe that if students are consistently required to get the right raw form then using calculators to do the evaluation should not cause problems.

4. Test Design and Implementation

§2 gives an analysis of a single issue of a whole constellation. Rather than consider the issue in isolation we build on the analysis in Task–oriented Math Education.

Test features already identified as useful are summarized in §4.1 and we add to this those of §3.2.1, Better Sample Question. Specific format and programming proposals are made in §4.3.

4.1. **Learning Tasks.**

The analysis in Task–oriented Math Education suggests that test features should include:

- Computer–based (presented and worked in an electronic format);
- Software–generated (not assembled from a database of problems);
- Multiple–try (many equivalent instances rather than static);
- Instances can be used for practice, and provide diagnostic aids, reference links, etc. after scoring (designed as a learning environment);

We refer to tests with these features as "Learning Tasks" to emphasize that learning, not assessment, is the primary objective.

4.2. **Functionality.**

The example in §3.2.1 requires the following:

- a form box in which the student enters an arithmetic expression;
- an "Evaluate" button so that when it is activated:
- the result of evaluation appears in a *different* box, and
- when the test is scored the contents of these boxes is frozen (and correct answers, diagnostic aids etc. appear).

The evaluation appears in a different box so the expression can be preserved. Further, if there is an error—missing parenthesis for example—an error message should appear in the evaluation box. The expression can then be diagnosed, edited, and re–evaluated.

4.3. **Formats and Programming.**

We suggest formats that provide the functionality described above, and more.

4.3.1. *Test Format.* A test (or learning task) is an Adobe PDF document. PDF supports web links and forms, and embedded javascript can be used to process or evaluate material entered in form boxes. Javascript gives access to a wide range of mathematical functions so the functionality described in the previous section is easily obtained.

Other benefits are:

- Tests can be self–scoring and fully functional via embedded javascript without depending on a server or test system.
- For–credit tests generally will be linked to a test system for security and recording grades, but other than this will have exactly the same functionality as a free–standing practice test.
- Answers and diagnostic aids, revealed when a test is scored, can also have javascript functionality and can include web links to reference material.
- The content generating system is independent of for–credit administration systems. Content should be provided through a single open–source or public domain system, while security and database software for administration might be commercial products offered by a number of companies.

4.3.2. *Source Code.* LATeX is an effective source code for functional PDF documents. It can be compiled directly to PDF by the PDFTeX program[2], or compiled to PostScript and then Distilled to PDF.

- The hyperref package included in standard LATeX installations provides intra–document and web linking, and basic support for HTML forms.
- The AcroTeX education bundle developed by D. P. Story provides further support for HTML forms, and facilities for embedding document–level javascript in a PDF document.

Story's system actually provides packages for producing self–scoring PDF tests. These are not flexible enough (off-the-shelf) to provide the functionality described here, but almost all the work is done.

4.3.3. *Producing Source Code.* The problem generators used in the Math Emporium at Virginia Tech are written in Mathematica. Current versions produce problem source code designed to be processed by Mathematica and distributed as web pages from a server. Most of them could be easily modified to produce LaTeX source code.

This means the technology and methodology for producing the source code is, in a sense, already established. However this work is done in–house and there is little publicly available material on it. A more problematic point is that developing problem generators is a very high–level activity and requires mathematical and educational sophistication.

4.4. **Advanced Functionality.** Eventually more functionality will be needed than can reasonably be provided by javascript embedded in a PDF document. For instance multi–step problems will require some sort of iterative computational support. This should be done with a separate computational environment. The test would interact with the environment to specify the appropriate level of functionality and receive output, but would not itself provide the functionality. For a draft

[2]This document was produced in this way.

description of such a computational environment see Student Computing in Mathematics: Interface Design and Student Computing in Mathematics: Functionality

5. Conclusions

High–stakes tests may improve minimum competency but current tests influence instruction in ways that depress achievement at only modestly higher levels.

In this article we worked through an example to explore what might be involved in designing tests that would actually improve teaching and learning. Our conclusion is that it is possible in principle, but getting it to work will be very challenging.

This single example required:

- recognizing a subtle problem unnoticed or denied by large parts of the education community;
- mathematically sophisticated analysis of causes that revealed an unexpected flaw in traditional elementary instruction;
- figuring out how a test might provide context and motivation to fix the flaw;
- being willing to have every single aspect of testing, from administration strategies to computer–based formats, driven by instructional needs;
- having sufficient experience and technical expertise to see how to implement the design.

The objective is not to get the test design exactly right—this is unrealistic—but to get close enough that equally careful and sophisticated field–testing would lead to an effective version. A less–sophisticated analysis or a priori constraints on test design would lead to a system that no amount of field–testing could fix.