The Natural Language Mathematics Testing System

Peter Rice
Mathematics Department
University of Georgia

In the summer of 1985, the Mathematics Department of the University of Georgia launched a project to design a computer system that would create, administer, grade and record all of the regular exams given in College Algebra. Since then, the project has grown to include the testing of Precalculus and has been sold to other schools where it is in use in a variety of courses.

The original idea was to create a system similar to the one designed by Stephen Franklin at the University of California, Irvine. It utilized a database of questions stored on a mainframe computer and administered exams with terminals. There were two types of questions: questions that required a numeric answer and multiple choice questions.

The questions requiring numeric answers were administered in a straightforward manner. Sitting at a terminal, a student would work out the answer to a problem using pencil and paper, then enter the answer. Checking the answers to such problems amounted to comparing the response to the correct answer.

The major innovation of Franklin's system was the "rolling" multiple choice problem. The question was presented along with one of the possible answers. If that answer was rejected it was removed and the next possible answer was shown. The student continued replacing old possibilities with new ones until a satisfactory answer appeared. However, no answer could be viewed or chosen once it was rejected. This feature defeated the common practice of reviewing all of the answers to select the most likely answer and required the student to know absolutely which answers were correct and which were incorrect.

At Georgia, the designers were convinced that a more intelligent testing system was within the scope of modern technology. Specifically, it should be possible to have the student type in an expression and have the computer analyze it to determine whether it was an acceptable answer to the question posed. For example, if the problem was to factor a quadratic expression in x, the student could type in the answer using the normal keys on a computer keyboard. The process of determining whether it was a correct factorization would first check to see if the student's response was functionally identical to the correct answer, then to see that it was factored.

Determining whether two expressions are functionally identical can be a difficult or impossible problem, depending on the class of functions allowed, but the class of functions that is normally encountered in elementary mathematics courses is so small that a practical test of identity is not difficult to construct. To determine whether an expression is factored we compare the two factored expressions, the student response and the correct answer, to determine whether their factors are the same. Using procedures of this sort, we were able to produce grading procedures for common problems.

Using a database of problems causes certain difficulties. The database has to be large enough to ensure that the questions do not get passed around by the students, or the tests would be compromised. If it is large enough to avoid such problems, then it is difficult to construct and takes up a large amount of space. A better method of preparing a large number of problems is to use algorithms that generate specific problems using randomly chosen values for parameters. One algorithm can generate hundreds or thousands of individual problems with unique answers and it could be designed so that a single procedure would be used for grading. Since between one and two hundred algorithms would suffice for a course, the problem database could be reasonably small and easier to construct. The major difficulty with

this approach is that each algorithm has to be individually coded, requiring programming expertise in addition to the ability to make up good mathematics problems.

Another feature of Franklin's system that did not attract us was the use of a mainframe computer. Not only was there a large cost associated with setting up such a system (or even renting space on an existing computer), there were the problems of access and reliability. When a large machine is used heavily, the response can become unacceptably slow, making it almost impossible to arrange timed tests. Also, when a mainframe computer goes down, all of the testing system goes down with it. For these two reasons, we decided to try to create the system on microcomputers, which offered the advantages of low cost and freedom from catastrophic breakdown.

The major problems associated with developing a large system on microcomputers turned out to be the speed and graphics capacity of the machine and the existence of an easy to use but powerful programming language. The display of algebraic expressions called for graphics displays, and the processing of graphics causes most small machines to slow down considerably. Luckily, we were able to develop a system that was acceptably fast using assembler language on a PC clone. But most smaller machines, such as the Apple II series, Commodore, etc. were too slow or offered unacceptably low graphics resolution to be useful.

The other problem was the choice of a programming language. While the developers were competent in several languages, including assembler, we were planning to code the algorithms for the problems, and hoped that this task could be distributed among several people, not all of whom could be assumed to be so competent. At that time, BASIC was not well suited to system development because it was not a procedural language and had a 64K code and data restriction. Turbo Pascal 3.0 was much better because, although it had the 64K size limitation, it allowed the use of overlays and, using locally developed hooks into the operating system, we were able to have a Turbo Pascal program load and execute another program. Also, Turbo Pascal was becoming common and had an easy to use programmer interface. It took about two years to reach the limits of the language, and development slowed down considerably before the introduction of Turbo Pascal 4.0.

At the present time, major portions of the system are being rewritten in C both for speed and portability. An additional benefit of this conversion is the existence of many highly competent professional C compilers that can push the PC to its limits. However, the problem algorithms are still being written in Pascal for ease of support and because C will never be a common language.

A number of other problems came up and had to be solved in the course of the development of the system. Some were connected with the sheer mass of students that had to be tested, some were associated with the general problems of computer support and repair, and some have to do with the mechanization of the processes that we as teachers perform one at a time for our students. Here are three examples of such problems and a brief description of the solution.

1. In testing 1000 students per week on 35 computers, it is necessary to have the computer lab open at virtually all hours. Instructors cannot be expected to be in the room at all of these hours, so it was necessary to hire student assistants to man the labs. As long as the system was designed to be simple to use and had good security measures, this system proved workable.

2. Computers do break down. Whereas a system designed to be run on micros can sustain the breakdown of a testing computer, the file server must be working at all times. By having a spare machine available and keeping good backups of data, it is possible to avoid long delays or disasters. However, it is still helpful to have a ready source of computer repair expertise. We discovered that routine repair tasks on PC clones can be performed by untrained people, so equipment repair became a minor issue.

3. When we test in class, we regularly accommodate students with disabilities. With a computer testing system, such accommodation has to be built in to the system. We designed the system to allow the operator to create a printed version of a test so that students with special problems can be served in a more traditional way.

Because we were introducing technology to replace a traditional teaching function, it was inevitable that the question of suitability arise. Our colleagues immediately asked whether the system can test students as well as traditional methods. The best answer to that question is the personal testimony of teachers who have used the system and can compare it with traditional testing methods. After two years of experience, their testimony is that the system is as effective at testing students as the traditional method and, in some ways, does a better job.

However, complete evaluation of the system is more complex because it does more than just replace a traditional function of the teacher, it fundamentally alters the teaching environment. For example, it is designed to be used by students at their convenience, so all the students in a class do not take tests at the same time. This makes it possible for students who take the test later to gather information on the tests that have been given to their friends. Does this availability of extra information compromise the test? Also, since there is quite a bit of information about tests floating around, do the students concentrate on learning how to pass the tests and not on learning mathematics?

We have explored one statistic that sheds light on these questions. We have some data on the grades of students who took a computer graded precalculus course (all tests, including the final exam, given on the computer) and those taking a traditionally graded precalculus course along with their grades in the subsequent first quarter of calculus. After normalizing the precalculus course for grade average (in non-computer graded courses, class averages differed considerably), both groups had the same expected performance in calculus. That is, the computer graded precalculus student can expect his calculus grade to be .79 lower than his precalculus grade, while the student having the traditionally graded course can expect a drop of .78. This statistic indicates that the computer testing system did not promote rote learning of exam questions to a greater degree than the traditional testing system.

The system is being extended, expanded and continually evaluated, and has now been installed in other schools. Short term development goals include enhancing the system so that it may be used in calculus courses and looking at applications of the system in schools.