DOCUMENT RESUME

ED 104 667 SE 018 793

AUTHOR Carlson, Marthena; And Others
TITLE A Computer-Assisted Instructional System for Elementary Mathematics.
INSTITUTION Pittsburgh Univ., Pa. Learning Research and Development Center.
SPONS AGENCY National Inst. of Education (DHEW), Washington, D.C.; National Science Foundation, Washington, D.C.
REPORT NO LRDC-1974/23
PUB DATE 74
NOTE 85p.; Occasional small type in examples used may not reproduce sharply

EDRS PRICE MF-$0.76 HC-$4.43 PLUS POSTAGE
DESCRIPTORS *Algorithms; *Class Management; *Computer Assisted Instruction; Elementary Education; *Elementary School Mathematics; Individualized Instruction; Instruction; *Research; Testing
IDENTIFIERS Computer Assisted Testing; *Computer Managed Instruction

ABSTRACT

This pamphlet contains five reports on different aspects of the Oakleaf Small Computer Project. The major objective of the project was to individualize elementary mathematics instruction by using a fully-integrated system of computer-assisted instruction, evaluation, and class management. The curriculum used was based on IPI Mathematics. The reports given here provide detailed accounts of the operation of the system; student access to the machine, generation and scoring of tests, record-keeping functions, and many other topics are discussed. A study of the effects of practice at a computer terminal on the acquisition and retention of computational algorithms is also reported. (SD)

LEARNING RESEARCH AND DEVELOPMENT CENTER

1974/23

A COMPUTER-ASSISTED INSTRUCTIONAL SYSTEM FOR ELEMENTARY MATHEMATICS

MARTHENA CARLSON, ROBERT FITZHUGH, ROBERT GLASER, TSE-CHI HSU, ERIC JACOBSON, KURT PINGEL, GILBERT PUENTE, RICHARD ROMAN, AND JEROME ROSNER

University of Pittsburgh

A COMPUTER-ASSISTED INSTRUCTIONAL SYSTEM FOR ELEMENTARY MATHEMATICS

Marthena Carlson, Robert Fitzhugh, Robert Glaser,
Tse-Chi Hsu, Eric Jacobson, Kurt Pingel,
Gilbert Puente, Richard Roman, and Jerome Rosner

Learning Research and Development Center

University of Pittsburgh

1974

The research reported herein was supported by a grant from the National Science Foundation (NSF-GJ-540X) and by the Learning Research and Development Center, University of Pittsburgh, supported in part by funds from the National Institute of Education, United States Department of Health, Education, and Welfare. The opinions expressed do not necessarily reflect the position or policy of the sponsoring agencies and no official endorsement should be inferred.

Abstract

The Oakleaf Small Computer Project is designed to show that, in schools where individualized instruction is already taking place, a small computer system can help those schools become more adaptive and effective. Within this project, work has proceeded on developing a suitable computer time-sharing system, computer-aided instruction and testing, and a computer data management system. One desirable outcome of this project would be a coherent instructional system that would bring all those components together. This set of papers describes mathematics components of the project and shows how this integrated system might evolve. The papers were presented as a symposium at the American Psychological Association's 81st Annual Convention in Montreal, Canada, 1973.

TABLE OF CONTENTS

	Page
List of Tables	vi
List of Figures	vii
INTRODUCTION Robert Glaser	1
A COMPUTER-ASSISTED INSTRUCTIONAL SYSTEM Jerome Rosner, Tse-Chi Hsu, and Gilbert Puente	5
COMPUTER-ASSISTED TESTING Tsi-Chi Hsu, Marthena Carlson, and Kurt Pingel	16
LEARNING OF COMPUTATIONAL ALGORITHMS THROUGH COMPUTER PRACTICE Eric Jacobson	34
A COMPUTER-BASED INDUCTIVE APPROACH TO TEACHING ELEMENTARY MATHEMATICS Richard Roman	52
COMPUTER-ASSISTED INSTRUCTIONAL SYSTEM: COMPUTER REQUIREMENTS Robert Fitzhugh	68
References	77

LIST OF TABLES

Table		Page
1.	Prespecified Number of Items for an Exercise Page	25
2.	Average Number of Items and Average Time Spent in Minutes in Taking a Pre- or Posttest by the CAT Group	30
3.	Average Number of Items and Average Time Spent in Minutes in Taking a CET for E Multiplication Unit by the CAT Group	31
4.	Number of Students in Each Group and Level	38
5.	Average Accuracy and Latency Across Session Blocks for Maintenance Group	40
6.	Average Accuracy and Latency Across Session Blocks for Remedial Group	41
7.	95 Percent Confidence Intervals about Accuracy Means for Session Blocks and Totals	43
8.	Correct Answers and Objectives Passed for D Level Retention Tests	45
9.	Correct Answers and Objectives Passed for E Level Retention Tests	46
10.	Outcomes on Objectives	64

LIST OF FIGURES

Figure		Page
1.	Instructional and testing procedures at the Oakleaf School	8
2.	An example of QUERY output	11
3.	Example of a CMI prescription	14
4.	Branching strategy for pre- and posttesting Unit E multiplication	18
5.	Graph illustrating the comparison between the number of items tested and the decisions made in 1971 CAT program and the revised CAT program	23
6.	The computer-assisted testing model	24
7.	Graphic representation of experimental treatments	39
8.	Sample CRT displays of a student interacting with the IPI-FUNCTIONS program	55
9.	Instructional puzzles for adding fractions and comparing the sum to one	60
10.	Computer van--internal view	70
11.	LRDC Experimental Time-Sharing System Hardware Configuration	72

INTRODUCTION

Robert Glaser

When a new technology appears on the scene with an apparent potential for improving education, the new technology can be handled in two ways. One way is to immediately list the technology's virtues, state how it can improve things, mount some quick displays of its use, turn it into something marketable, and sell it to the schools where it will be used to improve the quality of education. The technological object itself becomes the focus of concern. How can it be used is the paramount question. This way of dealing with a new technology seems to have been the mode employed in recent years with educational television, free-standing teaching machines, programmed instruction materials, and computers in the schools. When introduced in this way, the technology generally becomes the tail that wags the dog; cultists arise who advocate use of the new technology, and the new technology becomes a discipline of its own, relatively uninfluenced by relevant or needed knowledge. The technological entity becomes the central thing, rather than becoming an object to be integrated into a knowledge system where its use can be systematically exploited and studied. When introduced in this way, there is a great crescendo of excitement, writing, funding plans, followed by a gradual petering out of enthusiasm, a mounting disappointment, and wonderment about what went wrong. Final questions to be asked include: Is the new technology really useful? Are we ready for it yet?

A second way in which new technological developments can be handled is to work with them in the context of disciplines that can nurture them and to which they might, in turn, contribute. This generally means

that aspirations for immediate breakthroughs take a back seat while the power, utility, and force for change of the new object are allowed to emerge through study of failures, little successes, and the demonstration of utility in relatively small-scale intensive application. The accumulation of evidence of potential stimulates some avenues of use and discourages others. What happens in this latter mode of technological development is that scientists, engineers, curriculum developers, and practitioners begin to bring their knowledge, their problems, and their questions in contact with the potential of the new technology. Experimental give-and-take occurs, and the results obtained are informally or formally assessed as is appropriate to the situation.

In the course of this experimentation, different parties benefit at different times. Sometimes, education in the schools seems to be neglected because some computer system problem seems important to solve, or some theory of learning or instructional technique needs to be assessed, or some way of reorganizing the classroom and the traditional school day needs to be simulated to see how feasible this would be if a computer were introduced. At the same time that these things go on, pilot attempts in cooperation with practical school people and curriculum developers are being conducted.

This symposium reports on some possible uses of a computer resource in the elementary school. The way this work has proceeded can be generally characterized by the second mode I have just described. The project has two fundamental goals:

(1) To examine the ways in which computer technology could be used to assist in making teaching and the environment for learning more personalized and individualized and less subject to regimentation and lock stepping. The terms "personalized" and "individualized" these days have

the sound of empty catchwords, but in a deep sense, they represent the long-term aspirations of educators to teach according to needs of individual students, a goal which has been continuously and widely espoused but extraordinarily difficult to achieve in the context of our conventional school structures. The achievement of individualized environments for learning is a serious matter requiring more detailed developmental work and less general nonoperational talk.

(2) To design a computer system (software and hardware configurations) that can facilitate and encourage the innovative use of computer technology in elementary schools (and in schools in general). Toward this end, the project has designed and is investigating the capability of a medium-sized computer that is inexpensive and manageable, but still efficient and large enough to serve as a local facility for a school or for several schools in a system on a shared basis. A local computer (with perhaps some infrequent network assistance) appears to have reasonable possibilities for relatively immediate adoption by schools in the near future. The costs of small computers are decreasing and eventually might be within the range of a capital investment for a small school system. A local computer eliminates the expense and awkwardness of long-line communication, eliminates the necessity for an investment in a computer personnel organization, and incorporates the virtues of a system dedicated to a special purpose. With the development of sophisticated and ingenious software, a relatively small computer might well serve the needs of a school for computer technology. In the long run, when computer utilities are as common as electric power utilities, plugging into a large system might be the most efficient way to go, but at the moment, it appears that innovation will be encouraged by attention to small system design and investigating its capabilities for service to a school.

To accomplish these two purposes, several lines of investigation were simultaneously started. The intention was to investigate different aspects of computer use in some detail, and to concentrate these efforts in a way so that they all would converge upon a particular group of students and teachers. This means keeping within the limits of available resources by working on a number of computer services in a small and intensive way, rather than carrying out the extensive development of one overall curriculum or one kind of service that would cover a whole school. In general, the tactic was to investigate in a few classrooms a range of different possible uses of computer technology. As these various uses are assessed, an overall system for services and research can be planned.

A COMPUTER-ASSISTED INSTRUCTIONAL SYSTEM

Jerome Rosner, Tse-Chi Hsu, and Gilbert Puente

This paper has two purposes: (1) to describe the school environment in which the Oakleaf Small Computer Project operates and how the school-project interface was effected; and (2) to describe a student performance data management system that is intended to enhance the adaptive qualities of an educational environment.

Oakleaf is a small (300 children, kindergarten through grade six) elementary public school located in the Baldwin-Whitehall district of suburban Pittsburgh. It has served as an LRDC developmental school since 1964. As such, most of the instructional programs used in the school are individualized. The intermediate mathematics program currently in use--IPI Mathematics--has been described in detail elsewhere (see Lindvall & Bolvin, 1967). The newly designed primary curriculum--Individualized Mathematics (1971)--is an updated adaptation of the lower levels of IPI math. Both programs are based upon the following general instructional model (from Glaser, 1970):

1. "The goals of learning are specified in terms of observable student behavior and the conditions under which this behavior is to be manifested." In other words, the student's instructional goals are defined as behavioral objectives with accompanying, clearly stated criteria.

2. "When the learner begins a particular course of instruction, his initial capabilities--those relevant to the forthcoming instruction--are assessed." Simply stated, pretests are used to make certain that the student is asked to learn (a) something he does not already know, and

(b) something he should be able to learn with reasonable effort, given his present knowledge.

3. "Educational alternatives suited to the student's initial capabilities are presented to him. The student selects or is assigned one of these alternatives." This acknowledges the importance of flexibility and, to some extent, learner autonomy--that more than one route is available toward achieving competency in a general domain of learning.

4. "The student's performance is monitored and continuously assessed as he learns." That is, tests are provided to measure the accomplishment of small increment instructional goals as well as for major milestones.

5. "Instruction proceeds as a function of the relationship between measures of student performance, available instructional alternatives, and criteria of competence." Simply, progress is noted as the student achieves instructional goals and demonstrates these competencies by meeting the precisely stated criteria of the curriculum tests.

6. "As instruction proceeds, data are generated for monitoring and improving the instructional system." This, of course, acknowledges that an instructional system must always be studied and modified to meet existing conditions--that no instructional system continues to be satisfactory if it is static and nonresponsive to the dynamics of the environment in which it is used.

Translated into operational terms, the mathematics curriculum used at Oakleaf comprises a hierarchy of instructional goals defined as behavioral objectives. The objectives are organized into units, each unit representing a specific subdomain of skills--such as addition, subtraction, multiplication, and division--at a specific level of difficulty. Criterion-

referenced tests have been constructed for each objective, thus providing both teacher and student with a clear statement of the desired behavior and the criteria for demonstrating mastery. Both the teacher and student are kept aware of the student's progress in a way that can be reliably recorded and monitored by using an organized strategy of: (1) placement testing (to determine the student's appropriate level and unit--"where he is"); (2) pretesting (to determine what the student already knows within a particular unit); (3) providing relatively self-instructional teaching materials to teach the student what he has not yet learned; (4) testing of short-term goals (individual skills) with curriculum-embedded tests (CETs); and (5) posttesting of units. The flow diagram shown in Figure 1 illustrates the testing-teaching procedures of individualized instruction at Oakleaf School.

The computer serves three functions at Oakleaf: testing (CAT--computer-assisted testing), instruction (CAI--computer-assisted instruction), and management (CMI--computer-managed instruction). The three are obviously interrelated; each is capable of providing meaningful input for others and, in turn, each is responsive to the input directed to it in a way that foretells the eventual possibility of the system operating independent of external supervision. For example, as CAT procedures are able to provide reliable data regarding a student's competencies and needs, it is feasible that CAI will provide the necessary experiences to teach those skills, and that CMI will serve as a monitor of the outcomes of both functions, transmitting information in both directions as well as to those outside the system who will use the data to refine the workings of the system itself. Only aspects of the CMI component will be discussed in this paper; the other two will be addressed by other papers in this symposium.

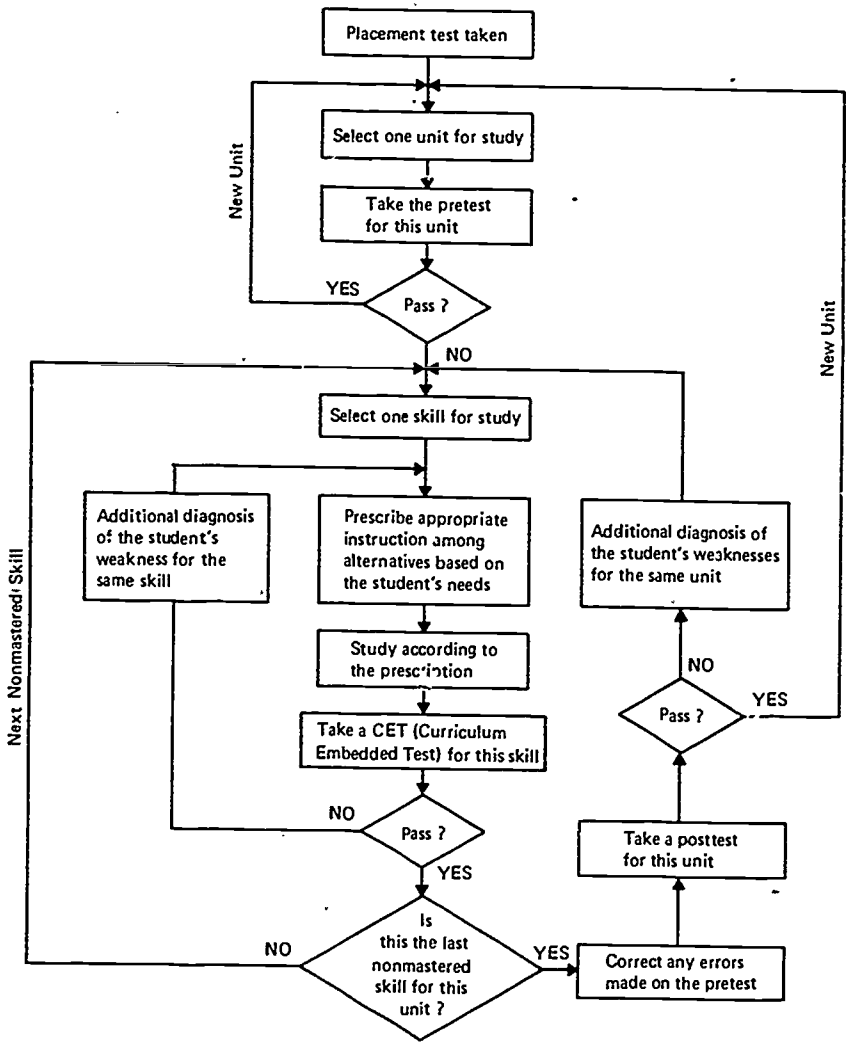

Figure 1. Instructional and testing procedures at the Oakleaf School.

Computer-Managed Instruction

A major goal of CMI is to provide teachers with up-to-date, on-line information for making instructional decisions. To accomplish this, a common file was designed to accumulate all pertinent daily performance data for each student, such as the skills he has worked on and is currently working on, the nature of the instruction (CAI, paper/pencil), the tests he has taken, and test outcomes.

Given this data base, the management system provides for three types of communications between the users and the system (Hsu, Fox, & Lerner, 1973). The first pertains to adding new records to the data base, modifying records, or deleting records. Three computer programs ("GATHER," "LIST," and "UPDATE") are designed for this purpose. At this stage in the development of the system, all data are collected interactively from students, teachers, or classroom aides. As mentioned above, when the system has been thoroughly tested and debugged, modifications will be made to the CAI and CAT programs currently on-line so as to transmit the necessary data directly and automatically at the end of an activity.

A second type of communication is the routine printing of reports that have been shown to be of general interest. Several such reports have been selected for initial development. These include:

a. REPOR1--this report consists of a list of students who have mastered at least one skill in a specified time period (from one to ten days) and a list of students who have not mastered at least one skill.

b. REPOR2--this provides a list of students who have failed the same CET or posttest more than once for current units.

c. STREP1 and STREP2--these reports tabulate the frequency of students in each unit of the two math curricula, Individualized Math (IM) and Individually Prescribed Instruction (IPI) math.

d. GCSUM--this is a summary of units completed for the period specified and is useful for preparing report cards.

e. IDDATA--the purpose of this program is to permit a listing of the roster for a specific ID. The roster consists of latest achievement test scores, IQ scores, and units completed to date for each student.

f. DAYSUM--this program calculates the mean and standard deviation of the number of days spent in each skill of a particular unit. The information is reported for each grade and is a grand total of all grades.

The third type of communication is Inquiry Response ("QUERY"). This deals with procedures for making requests from the system for a specific piece or block of information that is not available through the prepared report generating programs described above. For example, an inquiry might be made to the system asking the names of students who took the computer pretest for a particular multiplication unit on September 20. An example of QUERY output is shown in Figure 2.

These report programs (and others still in the planning stage) all share one major feature: They provide updated information regarding each student's performance, thus enabling the teacher (and other interested individuals) to identify quickly (1) those students who are "bogged down"

DO YOU WANT INSTRUCTIONS ? ••• YES

A. WHEN "HEADING" APPEARS FROM TABLE 1, TYPE ATTRIBUTES (SEPARATED BY COMMAS) THAT YOU WANT PRINTED. TYPE THEM IN THE ORDER YOU WANT THEM PRINTED ACROSS THE PAGE. DO NOT SPACE.

B. WHEN "TITLE" APPEARS, TYPE A REPORT TITLE OR JUST HIT CARRIAGE RETURN.

C. WHEN "1." APPEARS, TYPE AN ATTRIBUTE, A COMMA, AND ONE OR MORE VALUES (SEPARATED BY COMMAS). LIMIT: 10 VALUES.

D. THE COMPUTER WILL CONTINUE TO PRESENT YOU WITH 2,3,.... FOR EACH OF THESE ENTRIES, FOLLOW THE PROCEDURE IN C ABOVE. THESE ARE THE DETERMINANTS OF YOUR RETRIEVAL. WHEN YOU HAVE MADE ALL SPECIFICATIONS YOU DESIRE, TYPE "GO" IN RESPONSE TO ONE OF THE NUMBERS PRESENTED.

HEADING: NAME, GRADE, CLASS, DATE, SKILL

TITLE: PUPILS WHO HAVE FAILED COMPUTER CET IN SKILL 8 OF E DIVIDE

1. UNIT, EDIV
2. SKILL, 8
3. ACTIV, CCET
4. DECIS, NMAS
5. GO

NAME	GRADE	CLASS	DATE	SKILL	#CORR	#TEST	DECIS
BELL, SALLY	5	2	48	8	7	10	NMAS
BELL, SALLY	5	2	50	8	4	10	NMAS
BELL, SALLY	5	2	52	8	7	10	NMAS
BROOKS, TOM	4	1	53	8	6	10	NMAS
COX, JIM	5	2	62	8	7	10	NMAS
GREEN, TIM	4	1	73	8	7	10	NMAS
SMITH, RUTH	4	2	94	8	7	10	NMAS
SMITH, RUTH	4	2	97	8	6	10	NMAS
WOODS, JOHN	5	2	99	8	7	10	NMAS
WOODS, JOHN	5	2	100	8	6	9	NMAS

Figure 2. An example of QUERY output.

somewhere; (2) a weak component in the curriculum where students' progress tends to slow down, causing a "pile-up"; (3) those students whose classroom performance over time is sufficiently erratic to warrant close investigation; and (4) those students who are sufficiently precocious in their performance to suggest the initiation of a supplementary enrichment program. Hence, although its major function is to facilitate implementation of an individualized instruction model, CMI makes it possible to effectively monitor the progress of the entire group and to search for performance contrasts within and across individual students over time. The need for this in a responsive environment is evident; without such management resources the instructional system becomes less responsive and less individualized.

Yet another facet of CMI is providing suggested alternative prescriptions to the teacher based on the student's test performance--especially the nature of his errors. Still in the development stage, the ultimate goal of this facet is to identify effective instructional materials or activities for every math skill included in the curriculum. Standard teaching sequences and computer-assisted instruction programs are obvious alternatives. In addition, where the need is indicated, teachers will prepare their own instructional materials. Eventually, all of this information will be stored in the computer where it will be available to present suggested prescriptions to a student immediately following testing.

Conclusion

During the first two years that the computer was used at Oakleaf, the terminals were housed in a small conference room, isolated from the rest of the school and supervised by a paraprofessional aide who performed all procedures that preceded and followed a CAI or CAT session, that is, LOGIN, calling in the desired program, LOGOF, etc. The

students merely answered questions, in a sense using the cathode-ray tube (CRT) as an answer sheet in a workbook. The teachers did not "use" the facilities, they merely allowed their students to go to the terminals when they were summoned by the developers. As a result, the teachers gained little benefit from the computer and, indeed, viewed the whole enterprise with some distrust and perhaps even hostility.

The situation was altered last year. The terminals (10 CRTs, 4 TTYs [teletypes]) were relocated in learning centers--central spaces that are adjacent to and surrounded by classrooms--thus, making the computer programs much more public and available. More important, a procedure was designed and installed that enabled the teacher to decide which of the students would go to the terminals for testing and/or instruction.

Under this new system, the teacher provides the student with a prescription (a flow diagram specific to the task he is to engage in at the terminal). Figure 3 illustrates a typical prescription. It tells the student what program output will appear on the CRT, what signal will cue his response, what response is required from him to access the desired program, and so forth. Aided by this flow diagram, the student is able to initiate the desired program, participate in its activities (being aware of what he is to do in response to program requests), terminate it at an appropriate time, and return to his classroom with information regarding his performance recorded on the flow diagram sheet. In every grade but the first (and here, too, many are able to perform capably), the students were able to perform independently after a brief indoctrination, except in instances where unforeseen technical difficulties were encountered. The computer, in effect, has become an integral component of the school environment--an instructional resource that can be exploited by the teacher

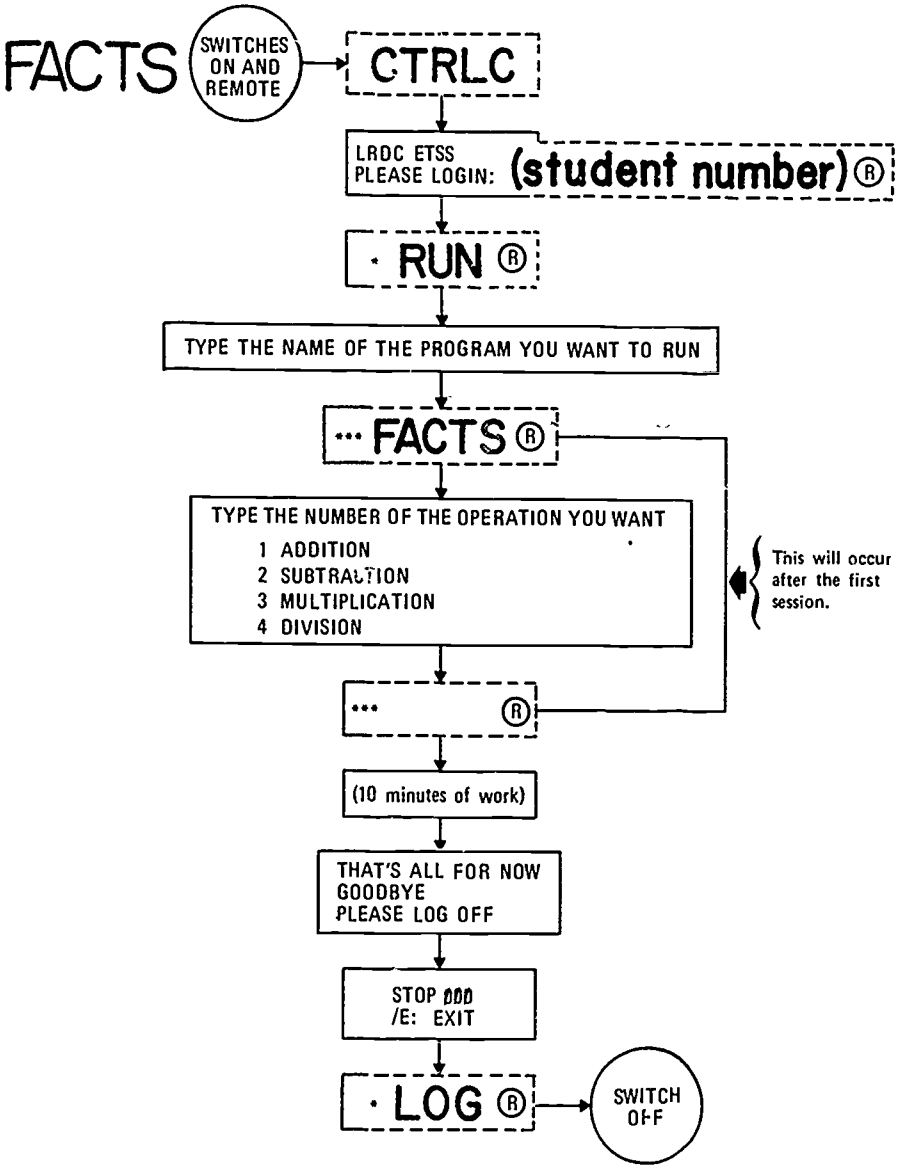

Figure 3. Example of a CMI prescription.

as (s)he guides the daily learning experiences of each student, and by the students as they gain awareness that they are the controllers, rather than the servants, of technology.

The ultimate worth of computer applications in an elementary school has yet to be determined, but our experience to date indicates that it has the potential to be an extremely useful component in the construction of an adaptive educational environment--an environment that is optimally responsive to all of its students.

COMPUTER-ASSISTED TESTING

Tse-Chi Hsu, Marthena Carlson, and Kurt Pingel

In an individualized instruction system, criterion-referenced tests are used extensively to diagnose the characteristics and to determine the mastery status of the learner. There are some potential weaknesses of these tests administered through the traditional paper and pencil mode: (1) The number of items and their level of difficulty are fixed at the time of construction; (2) criteria for mastery or nonmastery is fixed; (3) because there are tests for every objective, a child's time involved in testing is considerable; (4) tests may be too short to be reliable if only limited numbers of items are used for the purpose of reducing testing time; (5) management of testing and scoring is laborious; (6) feedback of test results to the child is not immediate; and (7) using a previously taken test is occasionally required, thus raising questions about practice effect.

In a computer-assisted instructional system, it is logical to use the computer for testing as well as for instruction. Rather than simply replacing existing paper and pencil tests with computer-administered tests, it is desirable to improve measurement procedures and overcome some of the deficiencies of paper and pencil testing. Some of the ways a computer can be used to improve measurement procedures are: (1) test more skills in a shorter time; (2) perform a more detailed diagnosis without being limited by fixed difficulty levels and number of items; (3) increase the validity of the decisions of the students' mastery status with reference to a specific skill; (4) provide precise diagnostic information and immediate feedback; and (5) generate additional nominally equivalent tests for each

skill. In other words, the purpose of using the computer for testing is to promote adaptive testing by taking into account individual differences. Reaching the desired level of efficiency necessary for adaptive testing is easier in the computer mode than it is in the paper and pencil mode. The advantages and potential of adaptive testing are discussed extensively by Weiss and Betz (1973).

This paper presents our experiences in using computer-assisted testing within an individualized instruction system. Discussion is focused on the design of the computer-assisted testing model and psychometric issues associated with the implementation of the model.

The Computer-Assisted Testing (CAT) Model

The computer-assisted testing (CAT) model (Ferguson & Hsu, 1971) is designed for a unit of instructional content that consists of several behavioral objectives (skills). The model is intended to perform pretesting, posttesting, curriculum-embedded testing (CET), and to generate exercise pages (practice problems) for all of the skills included in the unit. Basically, the model consists of four components: (1) item generation based on the requirements of the skills; (2) branching strategies based on the nature of the content; (3) item administration and scoring according to required formats, and (4) decision-making about the mastery status of the student. A multiplication unit, shown in Figure 4, will be used to illustrate the structure of the model and the development of computer testing programs based on this model.

Item Generation. This component generates the numbers specified by the item forms developed for an objective. An item form is a general specification of a behavioral domain. The actual test items which represent that behavioral domain are categorized by a list of generation rules.

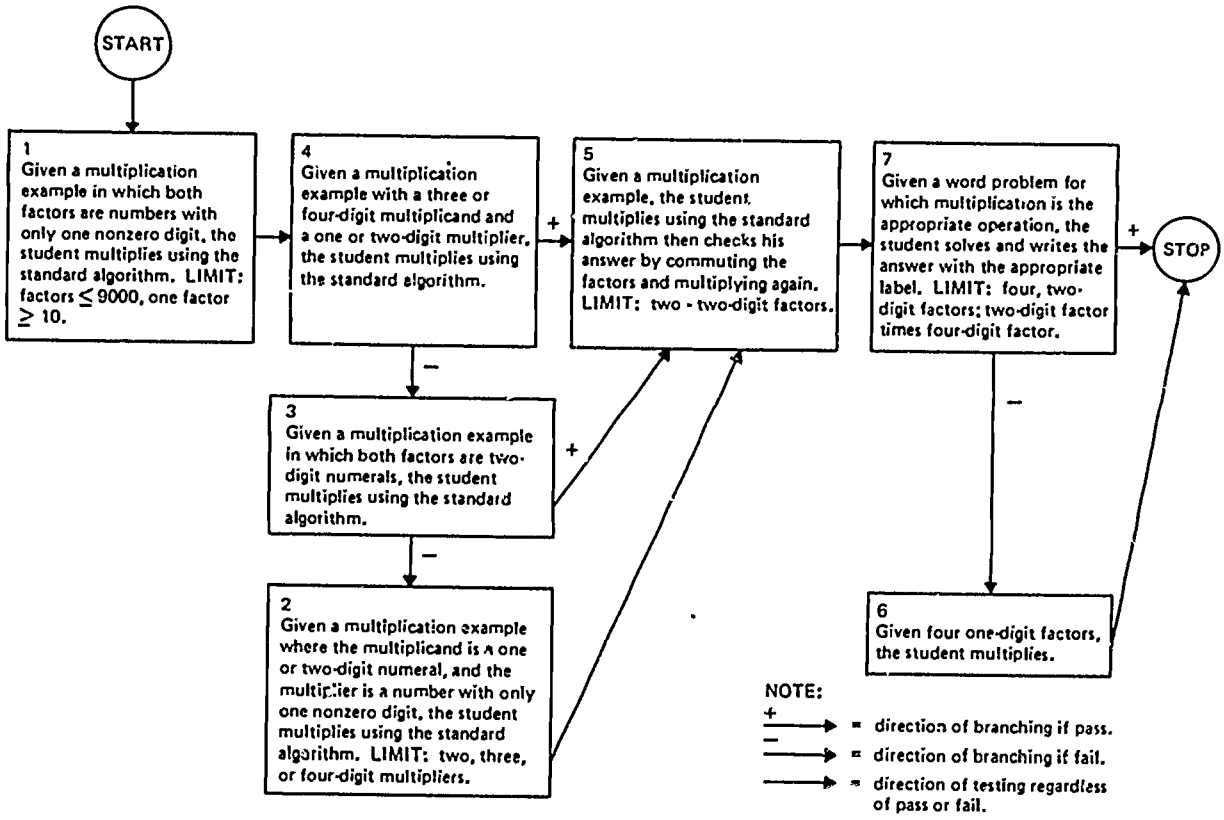

Figure 4. Branching strategy for pre- and posttesting Unit E multiplication.

These rules are used to generate nominally parallel items (Hively, Patterson, & Page, 1968). Normally, three or four item forms are developed for a behavioral objective. However, the number of item forms developed for an objective depends on the domain of the objective.

The item forms for the CAT model were established by content analysis and by examination of the existing test materials to determine the types of errors students tend to make within each objective. Properly constructed item forms should be helpful in prescribing appropriate instruction. For illustration purposes, the description of three possible item forms for Objective 3 in Figure 4 are listed next.

Objective 3: Given a multiplication example in which both factors are two-digit numerals, the student multiplies using the standard algorithm.

Form I:	Vertical. All digits in the multiplier and multiplicand will be 1-5.
Form II:	Vertical. The digits in the multiplicand will be 1-5 when the digits in the multiplier are 6-9 or vice versa.
Form III:	Vertical. All digits in the multiplicand and multiplier will be 6-9.

The unique requirements for each form are specified in the program. When a test for this objective is requested, numbers will be generated randomly to meet the specifications of the item form sequentially from Form I to Form III, one at a time. This procedure assures the efficient generation of a computer test because actual items do not need to be stored. The use of adequate item forms assures sampling of the domain, and random generation of the digits within forms guarantees

nominally parallel tests, not identical item for item, but similar because the replacement digits have a controlled range.

Branching Strategy. This component controls the sequence of testing. In order to provide branching that takes into account the performance of the student, branching sequences must be established for each unit and programmed into the model to manage the flow of testing. The following considerations were used to determine the branching sequences.

(a) A valid hierarchy was assumed. Therefore, when a child passes a higher objective, the lower objectives within the same branch are assumed mastered. For example, in Figure 4, Skills 2 and 3 are assumed mastered if Skill 4 is passed.

(b) All terminal behaviors in the unit are tested before certification of mastery is made. In Figure 4, Skills 1, 4, 5, and 7 are terminal objectives.

(c) In a larger unit, say more than 10 skills, testing should begin near the middle of the hierarchy so that branching can be either way, and testing time can be reduced.

Item Administration and Scoring. After the required numbers are generated in component 1, this component presents the item according to prespecified formats and computes the key for the item. When the student enters his response, it is matched with the key. The student is given a message if the response is correct. If the response does not match the key, the key is printed. The number of correct responses is recorded internally for decision-making purposes.

Decision Making. The last component of this model is the decision-making strategy. In CAT, the strategy for determining the mastery or non-mastery of a skill is based on Wald's Sequential Probability Ratio Test

(Wald, 1947). A detailed discussion of the application in computer testing is provided by Ferguson (1971). The following is a brief summary of the concept.

This test is computed from four variables:

(a) p_0 -- an arbitrary criterion for mastery;

(b) p_1 -- an arbitrary criterion for nonmastery;

(c) α -- Type I error (tolerance for misclassifying a mastery student as a nonmastery student);

(d) β -- Type II error (tolerance for misclassifying a nonmastery student as a mastery student).

Consider the determination of the mastery status for an individual as the testing of the following hypotheses:

$$H_0: p = p_0$$

$$H_1: p = p_1$$

The questions governing the sequential process in Bernoulli Trials are:

(a) If $\left(\frac{p_0}{p_1}\right)^s \left(\frac{1-p_0}{1-p_1}\right)^f \leq \frac{\alpha}{1-\beta}$ then accept H_1.

(b) If $\left(\frac{p_0}{p_1}\right)^s \left(\frac{1-p_0}{1-p_1}\right)^f \geq \frac{1-\alpha}{\beta}$ then accept H_0.

(c) If $\frac{\alpha}{1-\beta} \leq \left(\frac{p_0}{p_1}\right)^s \left(\frac{1-p_0}{1-p_1}\right)^f \leq \frac{1-\alpha}{\beta}$ then take an additional observation,

where: s is the number of successes,
f is the number of failures.

Taking logarithms, the last expression becomes:

$$\log \frac{\alpha}{1-\beta} \leq s \log \left(\frac{p_0}{p_1}\right) + f \log \left(\frac{1-p_0}{1-p_1}\right) \leq \log \frac{1-\alpha}{\beta}.$$

The equations giving the boundaries of this keep-sampling region are:

$$\log \frac{\alpha}{1-\beta} = s \log \left(\frac{p_0}{p_1}\right) + f \log \left(\frac{1-p_0}{1-p_1}\right) \text{ and}$$

$$s \log \left(\frac{p_0}{p_1}\right) + f \log \left(\frac{1-p_0}{1-p_1}\right) = \log \frac{1-\alpha}{\beta}.$$

A graphical representation of these boundaries is given in Figure 5. For example, using the decision boundaries for the revised CAT program (solid lines), a student who answers seven of eight items correctly and then answers the ninth one correctly, is classified as a mastery student. If the student did not answer the ninth item correctly, he is administered another item. The lower limits of the decision boundary illustrate that no student may master a skill with less than five items tested or achieve nonmastery with less than two items tested.

Actual programming of this model into a computer testing program is illustrated in Figure 6. This flowchart shows the relationship between these four components. After the student turns on the terminal and requests a program for a specific unit, the program asks for the student's name and the type of test--either a pretest, CET, posttest, or exercise page. For CETs and exercise pages, the student is asked to enter the number of the objective he wants. For pretesting, the student begins with the skill prespecified in the branching chart as shown in Figure 4. If this test is a continuation of a previous test, the student begins with the skill interrupted previously.

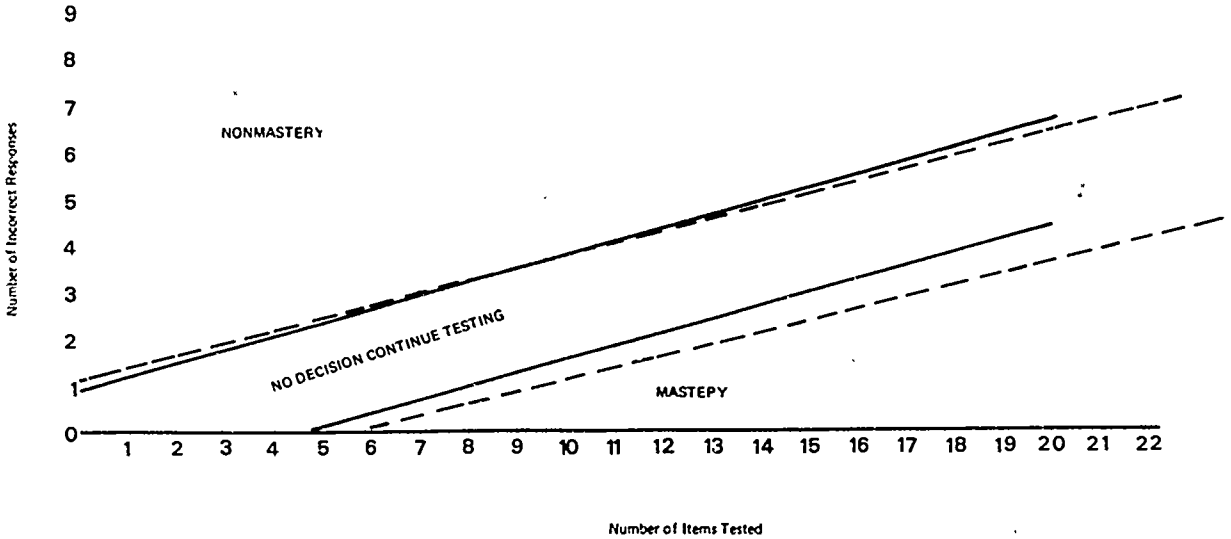

Figure 5. Graph illustrating the comparison between the number of items tested and the decisions made in the 1971 CAT program (dotted lines) and the revised CAT program (solid lines).

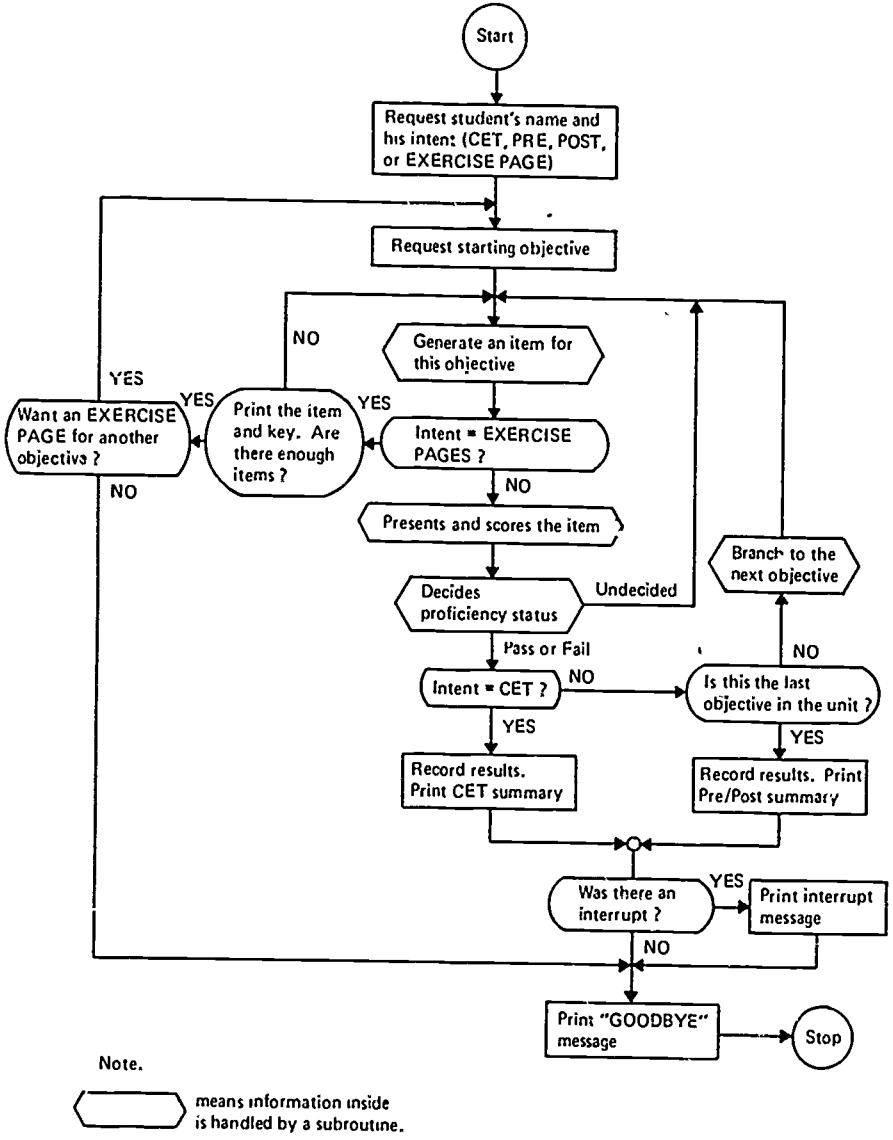

Figure 6. The computer-assisted testing model (adapted from Hsu et al., 1973).

When a test for a specific objective is requested, the program randomly selects numbers from the range specified by the item form. Items are constructed from these numbers according to the required format and presented to the student. The computer waits for a response and then checks the accuracy of the response. If the response does not match the key, the key will be printed before the next item is generated. In the case of exercise pages, the program continues to generate items and keys for later scoring until it reaches the numbers specified in Table 1.

TABLE 1

Prespecified Number of Items for an Exercise Page

Number of Item Forms in a Skill	Number of Items For Each Form	Total Number of Items
1 or 2	5	5 or 10
3	4	12
4	3	12
5 or more	2	10 or more

After each item is scored, the number of correct responses is cumulated into the record for decision-making using Wald's Sequential Probability Ratio Test. If continued testing is indicated, the program will branch back to item generation for the same objective using the next item form in order. If the decision can be made, either pass or fail, the program will print the summary statistics or proceed to the next skill in the case of pre- or posttesting. The summary statistics classify the student's responses according to item forms. This summary is useful to teachers for prescribing appropriate instructional material.

Psychometric Issues

Based on the CAT model described above, experimental programs were written for E level units of IPI Math in addition/subtraction, multiplication, and division. (The classification of IPI Math units is given in the first paper of this symposium. E level units are comparable to elementary arithmetic for third or fourth grade.) Field testing of these programs at Oakleaf School began in Fall 1971. Major activities in the 1971-72 school year were to collect data for improving CAT (Carlson & Hsu, 1973), and to explore the procedures for evaluating the quality of the tests generated and administered by a digital computer (Hsu & Sebatane, 1973). The evaluation effort for the 1972-73 school year focused on the comparison of the CAT group and the P&PT (paper and pencil testing) group in terms of average time spent in taking each test, average number of days spent in the unit, scores on a criterion test, and students' attitudes toward math, testing, and the computer. Overall evaluation effort is still underway so that no conclusive results can be presented here. Only some psychometric issues associated with CAT will be discussed below.

Mastery Criteria. The sequential decision strategy for mastery status is an important characteristic of CAT. This strategy allows the number of test items to vary according to the child's performance. If a child is consistently wrong or right, an earlier decision of either mastery or nonmastery can be made than is presently possible with paper and pencil tests. If the pattern of responses is not a sequence of all right or all wrong, more items are administered. The additional items should permit a more valid decision than would be possible with paper and pencil tests.

In 1971-72 the values used in the decision strategy were arbitrary mastery, $p_0 = .85$, arbitrary nonmastery, $p_1 = .60$, Type I error, $\alpha = .20$, and Type II error, $\beta = .10$. If a decision was not made in 30 items, it was

forced after considering the proportion correct for those 30 items. In comparing the number of mastery decisions made under the CAT criteria with those from the P&PT criterion (85 percent mastery), the conclusion was that students were less likely to achieve mastery under the CAT criteria. In addition, the teachers felt that a maximum of 30 items per skill was too high. Although there were only 10 of 104 tests where more than 20 items were administered, we decided to modify the decision rules so they would be more comparable with the paper and pencil decision, and hopefully, make the teachers feel more comfortable using CAT. Thus, for 1972-73, the arbitrary nonmastery criterion (p_1) was reduced to .55. This change narrowed the region of continued testing, as shown in Figure 5, and faster decisions of either mastery or nonmastery could be made. In addition, the maximum number of items to be tested for each skill was reduced to the same number of items presented on exercise pages shown in Table 1.

The arbitrary values used in our decision strategy are not final in any sense. They should be further evaluated in terms of the nature of content, the ease of transfer to the next higher order skills, and students' previous background and interest levels. A Bayesian sequential approach is currently under investigation. This approach can take into account the difficulty of the skill and the student's previous history (Hsu & Pingel, 1974).

Quality of Computer Testing. The problem in computer testing is not that programming for testing is a difficult task, but that the procedures for evaluating the quality of the test have not been developed. The unique characteristics of the tests are that two students taking the same type of test for the same skill do not get the same items. The decision about the mastery status of the skill needs to be made without reference to other students; traditional test evaluation procedures are not designed for this

type of test. Therefore, one of our efforts is to explore the procedures for evaluating the quality of tests which are generated and administered by the computer.

An empirical study was conducted to compare P&PTs administered to a group of students with equivalent computer-generated tests administered to another group of students. This is essentially a study of matched-item testing, where every student takes the same items (P&PT); and unmatched-item testing, where every student takes different items (CAT) on the variables of item form difficulty, item form discrimination, and generalizability coefficients (Hsu & Sebatane, 1973). Item form difficulty is defined as the proportion of correct responses for items generated by that item form. Item form discrimination is defined as the difference between the proportion of correct responses in the mastery group and the proportion of the correct responses in the nonmastery group for all items generated by that item form. Stratified alpha coefficients were computed for paper and pencil tests (matched-item). Stratified gamma coefficients were calculated for computer-generated tests (unmatched-item).

The results show that the item forms for computer-generated tests are consistently more difficult than that of equivalent paper and pencil tests. However, the way item forms discriminate between mastery and nonmastery groups are about the same for both types of tests. In terms of generalizability, the stratified gamma coefficients of computer-generated tests are slightly higher than the stratified alpha coefficients of equivalent paper and pencil tests. An important conclusion of the study is that the item form as a unit for evaluating the quality of an unmatched-item test seems reasonable. In short, there are many issues along this line in need of further exploration. For example, the index of item form homogeneity is another possible procedure for evaluating the quality of computer-generated tests (Macready & Merwin, 1973).

Testing Hierarchy. The efficiency of CAT is dependent on an efficient branching procedure. In determining the branching sequence, a valid hierarchy for the unit is assumed. If this assumption is violated, an invalid decision about mastery status may occur. Thus, the evaluation of testing hierarchies is another important psychometric issue.

In order to determine if the hierarchy used in CAT was valid, the paper and pencil pretest scores were collected. Because paper and pencil pretests test all skills, a score is available for each objective and, therefore, provides evidence to investigate the assumptions made in establishing the branching procedure used in computer testing. Following a technique of modified scalogram analysis as discussed in Cox and Graham (1966), the scores on each objective were coded pass if the percent correct was 80 percent or above, and fail if less than 80 percent. If the hierarchy is valid, students should not master skills that are terminal skills unless they also master the subordinate skills.

For the branching pattern used in the multiplication unit shown in Figure 4, the questions are: Do students who pass Skill 4 also pass Skills 3 and 2, and do students who pass Skill 7 also pass Skill 6? Only three of 41 (7 percent) who passed Skill 4 did not pass one of the lower skills and two of 41 (5 percent) who passed Skill 7 did not pass Skill 6. Thus, we feel the testing hierarchy for this unit is reasonably valid.

Time Factor. Our interest in time involves two related issues. The first issue is time spent working on the test. The second issue is the amount of time it takes students to work through a unit. To study these issues, students were randomly assigned to one of the two test situations: CAT or P&PT.

In terms of time spent taking the test, our aim is to eliminate unnecessary testing if the decision can be made quickly; otherwise, more

items should be administered. In other words, making an accurate decision is more important than the reduction of time during testing. Nevertheless, we wish to keep the testing time within a reasonable range. This year the attempt to collect data for the paper and pencil group was unsuccessful and we have only the time required for the computer test group as shown in Tables 2 and 3. However, the average number of items required for a pre- or posttest is substantially smaller than that of P&PTs, which have 42 items. We suspect that the time spent taking computer tests is less than that spent taking paper and pencil tests.

TABLE 2

Average Number of Items and Average Time Spent in Minutes in Taking a Pre- or Posttest by the CAT Group

Unit	Pretest		Posttest	
	Number of Items	Time	Number of Items	Time
E Addition/Subtraction	21.2	47.4	22.2	49.3
E Multiplication	26.1	42.4	21.0	32.7
E Division	29.6	42.7	29.0	41.2

Table 3 shows the average number of items and average time spent for typical CETs. The computer CETs, on the average, test more items than the six-item P&PTs. The question, "Is there any advantage in testing more items on computer CETs?" was raised. This can be answered by the data obtained from posttests. This data showed that 81 percent of the students who passed a computer CET passed the first posttest on that skill. Only 44 percent of the students who passed a paper and pencil CET were able to pass the skill on the first posttest.

TABLE 3

Average Number of Items and Average Time Spent in Minutes
in Taking a CET for Skills in E Multiplication Unit by the CAT Group

Skill	Average Number of Items	Average Time
1	36/5 = 7.2	47/5 = 9.4
2	17/2 = 8.5	19/2 = 9.5
3	36/4 = 9	64/4 = 16
4	43/5 = 8.6	55/5 = 11
5	83/11 = 7.5	163/11 = 14.81
6	15/3 = 5	17/3 = 5.6
7	64/7 = 9.1	140/7 = 20

To determine if there is a difference between the two groups in the amount of time it takes students to work through the unit E Multiplication, the number of days to mastery was collected for each student. Using the number of skills a student was required to work as the covariate, an analysis of covariance was performed. The computer F-statistic is significant at 10 percent level ($F = 2.90$, $df = 1, 42$, $p = .0961$). We suspect that there are three possible interpretations. The first is the better validity of the decision criteria for mastery status used in computer tests. The better validity means that CAT students are more likely to pass the posttest and will not be required to restudy any portion of the unit. The student who fails a posttest will be required to restudy and, therefore, will take more time to master the unit. Another interpretation is the time spent taking the test. For those students with a clear-cut mastery or non-mastery, the decision can be made very quickly. These students will not spend unnecessary time testing and can proceed with instruction in an appropriate skill without any delay. A third interpretation is that a more specific diagnosis results in a more accurate prescription, thus reducing

time. These interpretations are all tentative. Further information about the time spent in instruction and accurate measures of time spent taking paper and pencil tests should be obtained to support these interpretations.

Summary

This paper describes the structure of the computer testing model and discusses the psychometric issues associated with the implementation of such a model in a system of individualized instruction. Because the evaluation effort is still underway, no conclusive generalization can yet be made. However, there are several remarks that can be made here.

First, our effort to make CAT one of the alternatives for testing under the present IPI system seems to be working. The students from third grade and above have few problems handling the terminal. The computer can generate desired tests and make valid decisions about mastery status comparable with the paper and pencil tests. Some teachers are uncomfortable having students leave their class to use the computer because of the loss of personal contact. However, the majority of the teachers are willing to prescribe CAT as a testing alternative. They find the test summary and exercise pages generated by the computer useful for prescription.

Second, our preliminary results show that students on the average can master a unit faster when CAT is used. This is probably due to the reduction of testing time and valid decisions about mastery status. More important, this tentative result implies that computer testing can be more adaptive to individual differences.

Finally, the use of the computer in testing has a potential for modifying the traditional concept of measurement. Although the computer alone may not administer an ideal test at this stage, audio and

graphical presentation of items should be possible in the near future. A combination of these applications can make traditional paper and pencil tests obsolete. On the other hand, traditional measurement concepts are focused on matched-item testing. With the development of individualized instruction and criterion-referenced measurement, the need for students to be tested by the same items no longer exists. The administering of unmatched-item testing by a digital computer should have great appeal. If this statement is accurate, we can anticipate the growth of unmatched-item testing and the development of measurement theories associated with this type of testing.

LEARNING OF COMPUTATIONAL ALGORITHMS THROUGH COMPUTER PRACTICE

Eric Jacobson

My particular concern is how students learn computational algorithms and how we can get students to learn these algorithms. Computational algorithms consist of a series of separate operations which, when considered in detail, can be very extensive. It can be presumed that their acquisition is not an instantaneous process but proceeds gradually across time as the student learns more and more detail of each operation. This internal acquisition is reflected at least crudely in the probability that a test problem will be computed correctly.

Of special interest for education is the terminal stage of this acquisition process. It is a goal of math instruction to produce students who henceforward will always, or nearly always, compute right answers. Thus, if this goal is to be met, the acquisition process must continue not simply to a point where the student is fairly accurate but to where the student is 100 percent accurate and to where it is reasonably certain that he will retain this capability for 100 percent accuracy.

Information required to learn an algorithm can be presented to a student in one of three modes. First, it can be given as a verbal description of the algorithm, so, for example, the student can read or be told how to multiply. Second, the student can watch a demonstration of the algorithm; that is, he can watch a teacher multiply correctly. Third, he can perform a potentially correct version of the algorithm and receive feedback concerning its correctness. Or, in other words, he can practice multiplying.

A priori reasoning strongly supports the notion that verbal and demonstration modes should be used relatively early in acquisition and practice relatively late. A complete novice would have little likelihood of generating a correct algorithm and thereby learning from practice. Only after verbal instructions and demonstrations have imparted some minimum amount of information does it seem likely that practice could be effective. Furthermore, according to long standing educational belief, the practice mode is important during the final phase of acquisition when the student achieves 100 percent accuracy and high retentiveness.

The above reasoning was used to develop and apply computer-presented practice at Oakleaf. Practice was given late in the acquisition period and it was reasoned that this practice would be especially useful in helping students achieve perfect accuracy and in aiding long-term memory of the algorithm. The late-acquisition period was easy to define in the individualized Oakleaf system as the period after the passing of the unit posttest. At this point each student has answered at least 85 percent of the problems correctly for each skill.

Retention as a goal presents an especially difficult problem, since the prime measure of acquisition available is present-time accuracy. If a student is working at 100 percent accuracy, yet still has not practiced enough to retain the algorithm, how can it be determined when enough practice has occurred? One solution to this retention problem is to give all students large amounts of postmastery practice. This insures that all students will get enough practice to retain the algorithm. This strategy has the obvious shortcoming that some students will be practicing much more than they need to. It might be possible to effect a savings in the effort put into practice by postponing practice after initial acquisition and applying it as remediation only for students who have failed to retain this

algorithm. In this way students who reach a high retentive state without practice need not waste time in practice. This strategy of delayed practice would be efficient only if it can serve to reinstate lost information relatively quickly and easily. If students cannot relearn easily from practice alone, then the delayed practice would not be efficient. In order to answer some of these questions some students were given immediate long-term practice while others were given smaller amounts of practice delayed several weeks after initial acquisition.

Method

Instruction. Computer practice programs[1] were written to present problems from two multiplication units in the IPI Mathematics curriculum (see Rosner et al., in this symposium). Unit D introduced the algorithm for multiplying multi-digit factors and led ultimately to the multiplying of a two-digit number times a two-digit number. Unit E continued work on the multiplication of larger integers, and introduced the commutative and associative laws for multiplication. Most students working in D were fourth graders; most working in E were fifth graders.

Each unit is divided into skills which must be mastered independently by the student. There are 13 skills in the D level unit and seven skills in E level. No practice was provided for three skills from unit D and two skills from unit E. These skills dealt with terminology, exponential notation, and the representation of arrays of objects as multiplication equations. Of those skills which were practiced, two skills in D and one in E could be identified as number facts only (single-digit products), six

[1] Computer programs used were designed and written by Kenneth J. Hernandez.

skills in D and two in E as multi-digit algorithm only and two skills in D and two in E neither. Included were such topics as the special properties of the factors 1 and 0 and the associative principle.

The practice program was based on hierarchies of problem forms taken from the units. Student success on one form caused the program to present problems of the next most difficult form up to the most difficult level in the unit, while failure caused the program to move back. This decision procedure meant that a successful student would remain at the top of the hierarchy practicing the most difficult problems from the unit. The algorithm used to solve these difficult problems subsumed as components the algorithms used to solve the easier problems.

Within a problem form, individual problems were produced by generating numbers randomly to meet the specifications of the form. The student copied the problem from the cathode-ray tube terminal onto paper with pencil, computed the answer on paper and then typed his answer back in at the terminal. The program then gave him feedback based on speed and accuracy of solution. If the student answer was wrong, the correct answer was shown and the student was asked to copy it. After the feedback interaction was complete, a new problem was shown. Practice sessions lasted ten minutes.

Procedure. In order to master a unit in the curriculum the student must show independent mastery of all the skills in the unit. Thus, at the point of mastery when a student entered the experiment, he had shown at least 85 percent proficiency for each skill in the unit. The maintenance group received three practice sessions a week for six weeks. The remedial group received equivalent nonpractice, nonmultiplication computer experience for three ten-minute sessions a week for four weeks and then three practice sessions a week for two weeks. The no-practice group received

the equivalent nonpractice, nonmultiplication experience for three ten-minute sessions a week for all six weeks. Twelve weeks after unit mastery and six weeks after completing computer work, all students were given the unit posttest again as a test of retention. The structure of the experiment is shown graphically in Figure 7.

While in the computer phase of the experiment, students left their classroom during the math period and went to the terminal where an aide helped them get onto the program to which they were assigned experimentally (either practice or nonpractice). The standard schedule was computer on Monday, Wednesday, and Friday, but if for any reason a student could not attend a regularly scheduled session, it could be made up on a Tuesday or Thursday. In this way all students received 18 sessions over the six-week period. Retention tests were given in the classroom by the teacher.

Subjects. In the individualized Oakleaf setting, students completed the target units and became eligible for the experiment separately throughout the school year. Each new set of three students entering the experiment was randomly assigned to the maintenance, remedial, and no-practice group. This random assignment was performed separately for D level and E level students. Because the final triad was not completed and because one retention test could not be administered, the number of subjects in each group was not equal. The number of subjects in each group for each unit level is shown in Table 4.

TABLE 4

Number of Students in Each Group and Level

	Maintenance	Remedial	No Practice
D	10	9	10
E	5	4	5

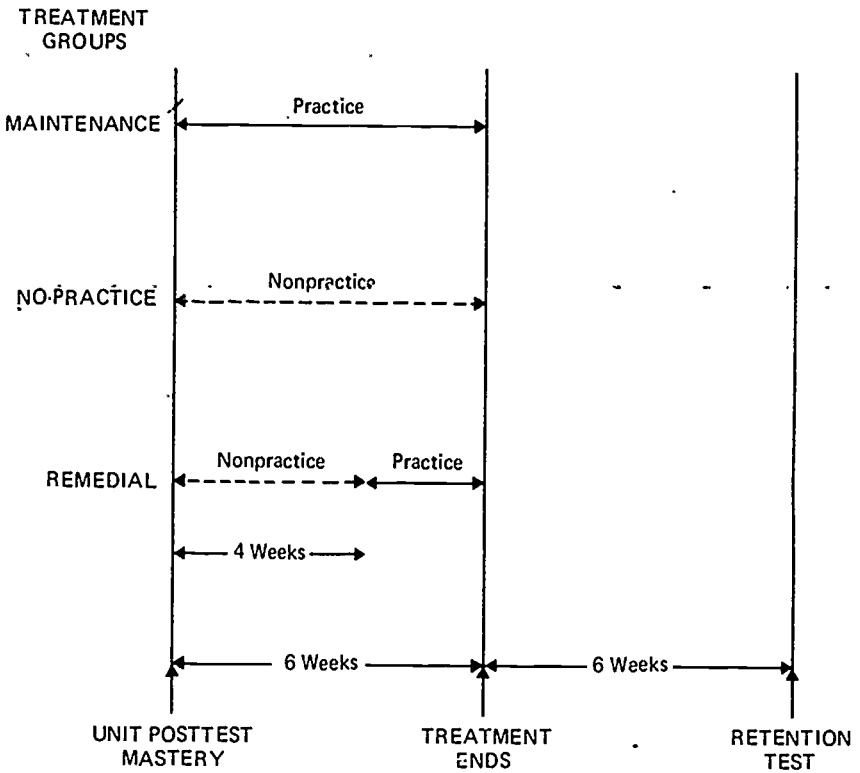

Figure 7. Graphic representation of experimental treatments.

Results

Accuracy in Practice. In order to remove the effects of variability due to program familiarization, the data from the first practice session were removed from the analysis. The remaining data were grouped into three session blocks (two sessions in the last block) yielding six blocks for the maintenance group and two blocks for the remedial group. Each block is the rough equivalent of a week's work. An error in the data retrieval system caused the loss of some data from terminal sessions, so some of the data blocks represent incomplete results. Table 5 contains the average percent correct and average answer latency for each session block for D and E level maintenance groups separately together. The first two lines in this table show the number of student sessions for which data were available in each block. Table 6 contains comparable data for the remedial groups.

TABLE 5

Average Accuracy and Latency Across Session Blocks for Maintenance Group

Session Block	I	II	III	IV	V	VI
Student Sessions in Block						
D	30	30	30	28	24	10
E	15	15	15	15	14	5
Percent Correct						
D	63	69	71	67	68	60
E	63	70	74	64	79	64
Total	63	69	72	67	71	62
Mean Latency in Seconds						
D	57	47	46	44	40	52
E	75	60	49	47	41	44
Total	61	50	47	49	40	50

TABLE 6

Average Accuracy and Latency Across Session Blocks for Remedial Group

	Session Block I	II
Student Sessions in Block		
D	27	15
E	12	7
Percent Correct		
D	56	69
E	74	75
Total	61	70
Mean Latency in Seconds		
D	56	51
E	54	40
Total	55	46

In the maintenance group, practice seemed to have very little effect on accuracy across blocks, with no block deviating greatly from 70 percent correct. Separate repeated measures analyses of variance were performed on D level subjects and all subjects combined, using phase as the treatment variable and using an arcsin transformation of the percentage scores. Only the first four phases were used since only these phases had data for all subjects. Both analyses yielded F-values of less than one ($df = 3, 10$, $F = .78$; $df = 3, 15$, $F = .26$) indicating no significant differences between phases in accuracy.

In the remedial group the four-week interval apparently produced a decrement in accuracy, at least in D level where the accuracy is 56 percent in block I. By block II, however, this deficit has disappeared and the percentage correct is indistinguishable from the maintenance group. Repeated measures analyses of variance for D level subjects alone and all subjects together indicated that this difference between phases in the remediation group was not significant ($df = 1, 8$, $F = 3.95$; $df = 1, 12$, $F = 1.43$; both $p > .05$).

The mean percentage correct for both maintenance and remedial groups is consistently below the 85 percent level at which each student had mastered the curriculum unit. In order to assess the significance of this difference, the 95 percent confidence intervals about the practice means were computed. These intervals were established for D level alone and D and E level together, for maintenance blocks I through IV separately and these blocks taken together, and for remedial blocks I and II separately and taken together. Computations were done on the arcsin transformations of the percentages and the resulting interval limits were then translated back into percentages. These intervals are shown in Table 7. The curriculum criterion of 85 percent falls within all intervals. Since a student can repeat posttests until he reaches the criterion level, it might be supposed that the 85 percent measure would be a biased estimate of the students's real knowledge. It seems likely that the students' real ability was close to the 70 percent level demonstrated in practice and that given that level of ability a student would have a good chance of putting together a run of correct answers sufficient to pass the posttest.

TABLE 7

95 Percent Confidence Intervals About Average Accuracy Means for Session Blocks and Totals (in percent)

	Lower Limit	Mean	Upper Limit
D Students Only MAINTENANCE Session Block			
I	36	67	92
II	25	71	99
III	32	71	97
IV	43	70	90
Total	35	70	95
REMEDIAL Session Block			
I	26	57	86
II	47	69	87
Total	34	63	88
D and E Students MAINTENANCE Session Block			
I	32	66	93
II	32	71	97
III	40	72	94
IV	43	69	90
Total	34	69	95
REMEDIAL Session Block			
I	27	64	93
II	45	71	91
Total	35	67	93

Latency in Practice. Answer latency declines across practice blocks for both maintenance and remedial groups (Tables 5 and 6). A repeated measures analysis of variance was performed on the main-

tenance scores for D level students alone and for D and E level students together using block as the treatment variable. Only the first four blocks for which there was a full complement of student data were included. Both analyses (D alone, and D and E together) indicated that the decline in latency was significant ($df = 3, 10$, $F = 8.83$, $p < .01$; $df = 3, 15$, $F = 45.84$, $p < .01$). A similar pair of analyses for the two remedial blocks yielded a significant F-value for D and E students together ($df = 1, 12$, $F = 8.79$, $p < .025$), but an F-value which was not significant for D level students alone ($df = 1, 8$, $F = 4.532$).

Retention. Retention scores can be examined in terms of the number of correct answers or in terms of the number of separate skills within the test for which the 85 percent mastery criterion was met. Both breakdowns are given in Table 8 for D level tests and Table 9 for E level. These tables also show the results for certain subsections of the entire test, including the objectives practiced in the program, those objectives in the unit but not practiced, objectives involving only the multi-digit algorithm and those involving only number facts. Entries in these tables are the number of answers correct, or skills passed, over total number of problems, or skills, and the resultant percentages, for all students within each group.

The story told in these tables is consistent and clear: There is no difference between any of the groups in any of the observed components of retention. There are some very small percentage differences in favor of the hypothesis that practice increases retention, e.g., 6 percent for D level skills passed and 8 percent for E level correct answers. Even these small differences, however, cannot be considered evidence in support of practice, since they are just as great for nonpracticed objectives as for practiced ones.

TABLE 8

Correct Answers and Objectives Passed for D Level Retention Tests

		Maintenance	Remedial	No Practice
Answers Correct				
	Total	1037/1210 86%	935/1099 86%	1027/1210 85%
	Practice	839/960 87%	758/864 88%	823/960 86%
	No-Practice	198/250 79%	177/225 79%	204/250 82%
	Algorithm	445/550 81%	398/505 79%	432/550 78%
	Number Facts	233/240 97%	210/216 97%	228/240 95%
Objectives Passed				
	Total	102/130 78%	89/117 76%	93/130 72%
	Practice	75/100 75%	69/90 77%	68/100 68%
	No Practice	27/30 90%	20/27 74%	25/30 83%
	Algorithm	39/60 65%	35/54 65%	35/60 58%
	Number Facts	19/20 95%	17/18 94%	18/20 90%

TABLE 9

Correct Answers and Objectives Passed for E Level Retention Tests

	Maintenance	Remedial	No-Practice
Answers Correct			
Total	351/425 83%	282/340 83%	318/425 75%
Practice	239/265 90%	192/212 90%	237/265 89%
No Practice	112/160 70%	90/128 70%	81/160 51%
Algorithm	56/60 93%	42/48 87%	53/60 88%
Number Facts	149/150 99%	120/120 100%	149/150 99%
Objectives Passed			
Total	18/35 51%	16/28 57%	18/35 51%
Practice	17/25 68%	12/20 60%	17/25 68%
No-Practice	1/10 10%	4/8 50%	1/10 10%
Algorithm	9/10 90%	6/8 75%	9/10 90%
Number Facts	5/5 100%	4/4 100%	5/5 100%

Discussion

In general, practice had little effect on student performance. Practice failed to cause an increase in accuracy of algorithmic computation and it produced little or no increase in the retention of the algorithm. This was so for the small doses of practice given the remedial group and the larger doses given the maintenance group. Practice did cause an increase in the speed of calculation, but compared with accuracy and retention, this effect must be considered of lesser educational importance.

. . . There are certain difficulties interpreting the retention test . . . results. One indication of this difficulty is the fact that students scored higher on the practiced objectives in the retention test than they did during practice. This anomaly can be explained by the fact that the hierarchy-based practice program tended to give students the most difficult problems whereas the retention test sampled all levels of difficulty uniformly. Nevertheless, despite this explanation, it is not possible to determine exactly how much forgetting, if any, took place.

The possibility that, in this study, a ceiling effect obscures the influence of practice might be entertained. The no-practice group did so well on the retention test that it might be argued that the practice groups could not show any superiority. We reject that argument by pointing out that in terms of the ultimate educational objective, these students were not close to the ceiling. About 70 percent of the practiced objectives were passed at the 85 percent criterion on the retention test. While this does indicate considerable learning and retention, it is definitely below the school expectation of 100 percent. Thus, although it cannot be determined whether the deficit was in initial learning or in a failure to retain what was learned, it is clear that these students had not

fully acquired an ability to use the multiplication algorithm accurately, and that practice did nothing to aid this acquisition.

Negative results such as these do not, of course, allow unequivocal interpretation. Perhaps practice does not have the beneficial effects initially hypothesized. Or perhaps the particular situational parameters used in this practice were wrong and the right kind of practice would have helped. Admitting that the decision procedures of standard methodology do not allow us to decide between these two interpretations, yet still requiring a decision in order to proceed, we chose the former interpretation as most reasonable. That is, we decided that our practice was as good as any and reformulated our ideas about the role of practice in general. In defense of this interpretation it can be said that no independent measure of program quality, in the domain of student attention and motivation and teacher evaluation indicated that there was anything wrong with it.

Here then is our tentative model of the learning of computational algorithms. This model is consistent with our results, and other data, and is guiding our next attempt at algorithm training. According to our model there are two phases in acquisition. In the first phase a student acquires an algorithm to a high proficiency, being able to compute accurate solutions for something like the 85 percent rate required in IPI. In phase two the student continues to increase his accuracy rate to an asymptote near 100 percent and succeeds in storing the routine in memory such that it is permanently retained.

At present we know very little about this second phase. Educational folklore has it that practice will bring about phase two acquisition but our data indicate that it will not. Other studies which have shown increases in the accuracy of long computational algorithms due to practice (e.g., Suppes & Morningstar, 1972) do not contradict our findings

or interpretation, since the final student accuracy rate in these studies does not approach 100 percent.

One explanation of phase two is that it is not acquisition at all, but a general lowering of error rate for all cognitive processes. During phase one a student learns an algorithm perfectly but fails to respond at 100 percent accuracy because of random errors occurring in the execution of the algorithm. Later, and by a process entirely independent of the knowledge of the algorithm, this general error rate is reduced (perhaps by the acquisition of self-checking routines, perhaps simply through maturation of the nervous system). The result would be what we have been calling a phase two increase in solution accuracy.

This explanation is not relevant to putative changes in long-term retention occurring during phase two. Perhaps this type of retention occurs only through very long-term spaced practice extending over several years. Another possibility is that it is a result of a reorganization of the storage structure of the algorithmic knowledge making it permanently retrievable. Speculation on phase two acquisition is hampered greatly by the lack of normative data on accuracy and retention in adults. It may be the case that adults do not perform above a mean of 85 percent correct and except for intermittent review through application would not retain an algorithm indefinitely.

Students in our study were practicing during phase two, that is, after they had already reached the 85 percent accuracy level. This practice had little or no effect on accuracy or retention. The students who practiced in studies reported by Suppes and his co-workers (Suppes & Morningstar, 1972) were practicing during phase one. They worked in nonindividualized systems in which individual mastery was not and could not be assured. Therefore, when they began practicing they were still

in phase one, and practice aided acquisition. Examination of pre- and posttests scores reported by Suppes and Morningstar (1972) bear this interpretation out. Although there was demonstrated improvement for each of the 114 practice block-school units used in their comparison (Chapter 5, Table 1, pp. 208-210), for only 51 such units was the final accuracy 80 percent or better, and for only 14 was the accuracy 90 percent or greater.

Thus, we cannot expect large amounts of practice administered after initial acquisition to either produce a further increase toward complete accuracy nor to increase retention. Practice, however, occurring during initial acquisition can aid that acquisition. Suppes and Morningstar have shown that practice is a sufficient condition for learning but not that it is either a necessary or optimal condition. We return to an earlier assertion that early in learning verbal explanations and demonstrations are relatively more efficient means of transmitting information. It may turn out that practice is not necessary at all for acquisition, or at least that it has an optimal effect in very small amounts.

Future Work. At the present we are working on what we have called phase one acquisition, attempting to take beginning students and bring them up to a reasonably high level of responding. At the outset we make two presumptions. First, optimal learning will require a sequence and/or mix of different instructional modes and, second, there will be individual differences between students in the optimum mix or sequence of modes.

There are actually four instructional modes we can make available to students. First, we can prepare short booklets that describe verbally the operation of the algorithm. Second, we have written computer routines that perform the algorithm in real time on a cathode-ray

tube and thereby demonstrate to the student the steps of the algorithm. Third, we have our standard practice programs that call for a student to use an algorithm and receive feedback on the result. We have sacrificed conceptual clarity in the pursuit of instructional effectiveness here by using the demonstrator routines to demonstrate the correct solution whenever the student gives a wrong answer. Finally, the fourth mode of instruction is the standard IPI instruction we began with, which is a mixture of verbal descriptions, static displays, and a small amount of practice with delayed feedback.

Initially we plan to make these modes available to the school and to try them out in different combinations before we formalize and test specific sequences. For example, with one possible procedure a student would take his completed unit pretest to the computer terminal and submit the problems he missed to the demonstration program to see what the correct algorithm looked like. Second, he would read a booklet describing the correct algorithm. Finally, he would try out the algorithm using the practice program and consult the booklet whenever questions arose. One of our ultimate goals will be a set of prescriptive procedures which will place each student into the mix of instructional modes that will optimize his learning.

Such prescriptions would be the basis of a management system that could use both previous student history and specific unit information to assign a certain student to a certain type of input mode. For example, a student who had shown ability in the past for acquiring algorithms from verbal descriptions could be assigned a booklet initially, or a particular novel algorithm could be introduced to all students via a demonstration. Such a system would in no way preclude student control of the options. A student could simply be given a choice among available instructional modes, if that were deemed advantageous.

A COMPUTER-BASED INDUCTIVE APPROACH TO TEACHING ELEMENTARY SCHOOL MATHEMATICS

Richard Roman

This paper describes the MATH FUNCTIONS curriculum package, designed to teach both mathematical content and problem solving skills. The package is one component of an individualized mathematics system operating in an elementary school. All instruction in the curriculum is accomplished through a computer program implemented on the school's computer resource. Other components of the mathematics system teach the same content material, giving students a choice of instructional media. The problem solving curriculum package teaches more than 100 behavioral objectives, or one-fourth of the elementary school mathematics for grades three through six.

The ability to successfully solve problems should develop inquiry skills. The curriculum package provides a structured envrionment which encourages problem solving activities such as stating the problem clearly, gathering and organizing data, using feedback, formulating and testing hypotheses, knowing you have finished, dividing problems into subproblems, and combining subsolutions into complete solutions. Students who experience success at solving problems in this structured environment increase their sense of competence and their desire to solve more problems.

From our research to date, some conclusions can be made. Both mathematical content and problem solving skills can be taught by structuring the mathematical content as a sequence of problems in which the student induces the organizational rule from examples. Appropriate

structuring of the environment in which rules are induced teaches problem solving skills. Mathematical content which requires simple rules, simple stimulus material and for which the student has mastered the prerequisite skills is taught effectively in conjunction with problem solving skills. Many students learn mathematical content in an environment which also teaches problem solving skills.

The mathematics curriculum for which this program is designed has been described in a book by Lindvall and Cox (1970). For our purposes it is enough to note that the school curriculum consists of approximately 400 objectives that are grouped into levels which correspond roughly to grades in school, and which are further grouped within levels into areas such as addition, subtraction, fractions, and money. An individualized booklet and coordinated manipulative materials teach each objective to those students who have met the prerequisites, but have not demonstrated mastery on the objectives as shown on a pretest for the area. Instruction in the booklets is programmed. Generally, the rules for operations are stated explicitly and followed by many examples on which to practice the rules.

While we accepted the objectives of the individualized curriculum, we designed instruction which differed in style from the booklet design. We believed that inducing the rules from carefully structured examples would give the students deeper understanding of the material. Our context seemed perfect for a project in teaching by guided discovery since most of the preconditions for such learning were assured by the way instruction is managed at Oakleaf (Gagné, 1966). Specifically, we are nearly certain that the child has met the prerequisites for the objective and is in a state of readiness for learning.

In the IPI-Functions program, we therefore offer many examples of the rules required to perform the objective, but never explicitly state the rules for the child. Careful sequencing of small inductive steps within each objective allows the student to master the objective by discovering the regularities embodied in those examples and applying these regularities to make predictions during self-testing.

To be more specific, approximately one-third of elementary school mathematics curriculum objectives, both at Oakleaf and in more traditional curricula, require the student to master a functional relationship. The student is given an element of a domain set and responds with an element of the range set. The functional rule always produces a single range element for each domain element. We designed a program which would present material of this type to the student and allow him to induce the rule by which the function operated.

An understanding of the rudiments of the program operation is essential for interpreting the results reported. (See Figure 8.) Figure 8 contains a series of displays as a student would see them on the cathode-ray tube screen while he was learning to round off to the nearest multiple of 10 or 100, an objective identified as "36-F" in Frame 1 of Figure 8. Frame 3 is the first instructional display. Notice that it tells what the student will learn and gives two samples of the function for him to assimilate. It also offers him the choice of trying to input an element of the domain, "A PLACE VALUE AND A NUMERAL," or of taking a test or saying done, meaning that he has seen enough of this level and wishes to change to the next level of problem.

Allowing the student to control the number of examples and the content of the examples is an important feature of the instructional design which allows him to formulate and test his own hypothesis through

Student responses appear following the three asterisks (•••). After the student has made a response (••• "360" - frame 3), this response and everything between it and the last sample ("100 30 BECOMES 0" - frame 3) is erased. The new line ("360 IS NOT A ———" - frame 4) and directions are then generated.

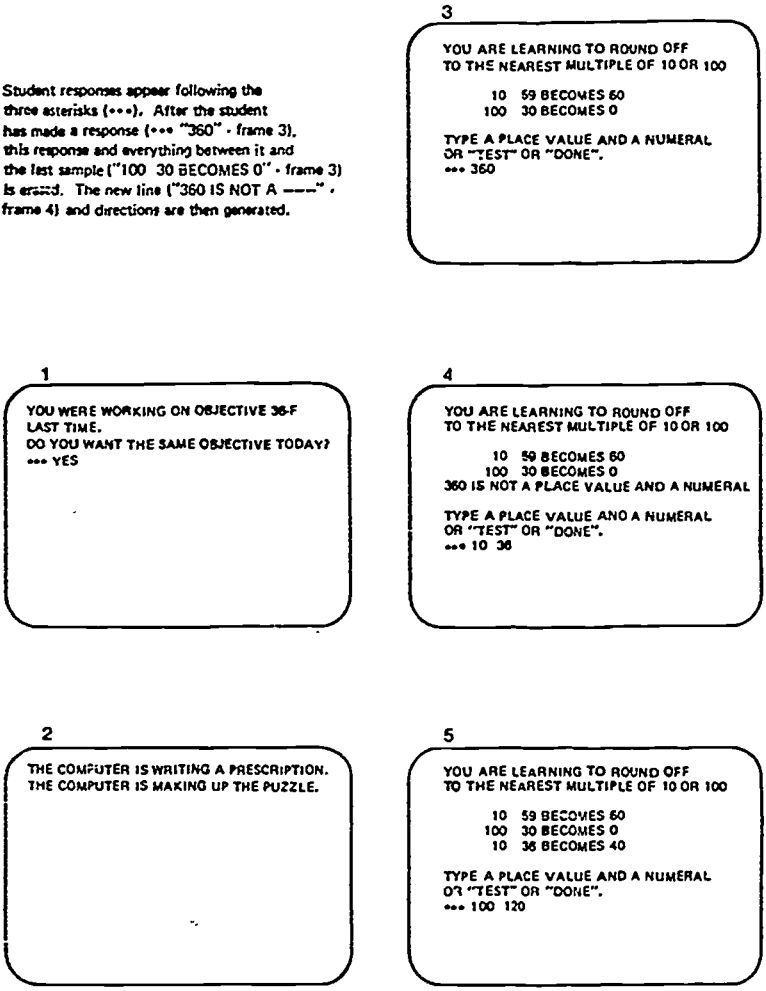

Figure 8. Sample CRT displays of a student interacting with the IPI-FUNCTIONS program.

6

```
YOU ARE LEARNING TO ROUND OFF
TO THE NEAREST MULTIPLE OF 10 OR 100

      10   69 BECOMES 60
     100   30 BECOMES 0
      10   36 BECOMES 40
     100   120 BECOMES 100

TYPE A PLACE VALUE AND A NUMERAL
OR "TEST" OR "DONE".
••• DONE
```

9

```
YOU ARE LEARNING TO ROUND OFF
TO THE NEAREST MULTIPLE OF 10 OR 100

      10   69 BECOMES 60
     100   30 BECOMES 0
      10   36 BECOMES 40
     100   120 BECOMES 100
      10   45 BECOMES 50
                NOT 40

TYPE A PLACE VALUE AND A NUMERAL
OR "TEST" OR "DONE",
••• TEST
```

7

```
YOU ARE LEARNING TO ROUND OFF
TO THE NEAREST MULTIPLE OF 10 OR 100

      10   69 BECOMES 60
     100   30 BECOMES 0
      10   36 BECOMES 40
     100   120 BECOMES 100

YOU MUST TAKE MORE TESTS BEFORE
SAYING "DONE"
TYPE A PLACE VALUE AND A NUMERAL
OR "TEST" OR "DONE".
••• TEST
```

10

```
YOU ARE LEARNING TO ROUND OFF
TO THE NEAREST MULTIPLE OF 10 OR 100

      10   59 BECOMES 60
     100   30 BECOMES 0
      10   36 BECOMES 40
     100   120 BECOMES 100
      10   45 BECOMES 50

TYPE "DONE" OR THE ANSWER
TO  10  11 BECOMES
••• 10
```

8

```
YOU ARE LEARNING TO ROUND OFF
TO THE NEAREST MULTIPLE OF 10 OR 100

      10   69 BECOMES 60
     100   30 BECOMES 0
      10   36 BECOMES 40
     100   120 BECOMES 100

TYPE "DONE" OR THE ANSWER
TO  10  45 BECOMES
••• 40
```

11

```
YOU ARE LEARNING TO ROUND OFF
TO THE NEAREST MULTIPLE OF 10 OR 100

      10   59 BECOMES 60
     100   30 BECOMES 0
      10   36 BECOMES 40
     100   120 BECOMES 100
      10   45 BECOMES 50
      10   11 BECOMES 10
      GREAT!

TYPE "DONE" OR THE ANSWER
TO  100  880 BECOMES
••• 900
```

Figure 8 (Cont'd). Sample CRT displays of a student interacting with the IPI-FUNCTIONS program.

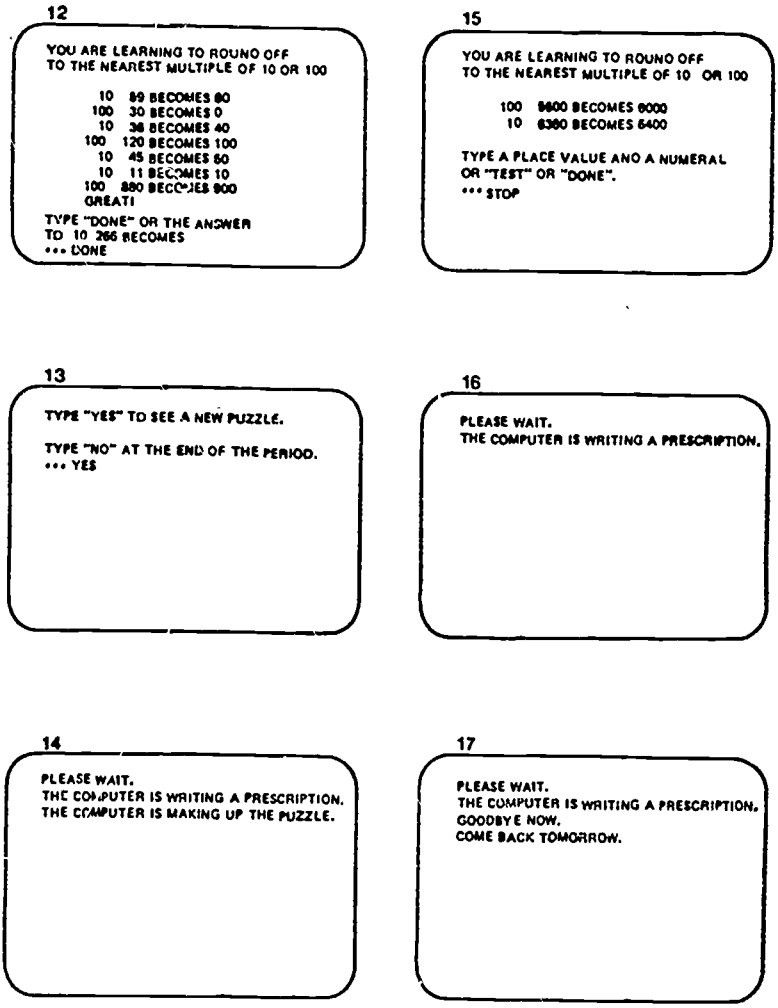

Figure 8 (Cont'd). Sample CRT displays of a student interacting with the IPI-FUNCTIONS porgram.

the use of appropriate examples. The student may stay in the example phase of the program as long as he likes.

In Frame 3, the student has misunderstood the domain, and Frame 4 explains that his input is incorrect. In Frames 4 and 5 he succeeds in making correct domain inputs and the computer displays the corresponding range elements. The student then tries to go on to a new level, which is illegal until he has taken some tests (see Frame 7). In Frame 8 a test item is generated randomly from the domain set. The student responds incorrectly and gets feedback in Frame 9. He continues to test himself in Frames 10, 11, and 12.

Another aspect of the instruction is illustrated here. The student may take tests as long as he likes. Hopefully, he will do so until he is certain that he understands the rule. The program is adaptive here since the number of tests required depends on the information contained in each item and the particular students' ability to assimilate the information. In addition, we feel it is important for students to exercise control over their own learning. An important aspect of self-controlled learning is the ability to know when one understands, is finished, and ready to move on.

In Figure 8 we examined the instructional features that recur with each function. In summary these were: the samples of the function; the consistent format of presentation; feedback on inputs; feedback on tests; and the opportunity for the student to control the number of inputs and the number of tests in a way that is consistent with the hypotheses that he is testing, and his own preferred working style. Students can and do make meaningful variations in their use of these features, and individuals change their behavior with respect to these options as they use the program over time.

We have spoken about the instruction to help a student induce a single rule. We call that instruction one puzzle, and the full instruction for a single objective requires two to 12 puzzles, corresponding approximately to single pages in a test. Figure 9 shows all the puzzles used to teach a single objective on summing fractions and comparing the result to one.

A glance at Figure 9 will give an idea of the way the difficulty of a function changes between puzzles. In this particular objective the student learns to sum two fractions with the same denominator and compare that sum to one. In the curriculum the comparison is always done with the fraction on the left and the number 1 on the right. This allows the student to respond correctly by simply knowing if the numerator or denominator of the sum is larger. He can ignore the 1 in the comparison and still pass the curriculum embedded test. Accordingly, puzzle 5 of the sequence is equivalent to the curriculum embedded test and mastery can be assumed after that puzzle. However, in this and most other objectives we teach, there are post terminal skills that extend the domain, or the generality of the rule learned, beyond the puzzle required for mastery.

When the student begins his work on Objective 42-A, he knows how to write fractions equivalent to two wholes (Objective 41-D). He can also, given two indicated sums, write greater than, less than, or equal to between the sums to complete the number sentence (Objective 33-F). He has not learned to add two fractions with the same denominator. Instruction in the program begins with puzzle three, with the student comparing a single fraction to one. If he cannot figure out what is required, he is given a similar problem at puzzle two, but without the fraction sign. If he masters, he is advanced to puzzle four, where he

Puzzle 1, Remedial Level

Domain: Two numbers up to 9 with a "?" in between

Range: The signs > , <, and = .

Rule: Type the signs which must replace the "?" to make a true statement.

Description for Student:

YOU ARE LEARNING TO ADD FRACTIONS WITH THE SAME DENOMINATOR AND COMPARE TO 1

USE < FOR LESS THAN
> FOR MORE THAN
AND = FOR EQUAL TO

2 ? 2 BECOMES =
1 ? 5 BECOMES <
8 ? 4 BECOMES >

Puzzle 2, Remedial Level

Domain: Two numbers up to 99 with a "?" in between

Range: The signs > , <, and = .

Rule: Type the sign which must replace the "?" to make a true statement.

Description for Student:

YOU ARE LEARNING TO ADD FRACTIONS WITH THE SAME DENOMINATOR AND COMPARE TO 1

USE < FOR LESS THAN
> FOR MORE THAN
AND = FOR EQUAL TO

36 ? 8 BECOMES >
4 ? 87 BECOMES <
85 ? 85 BECOMES =

Figure 9. Instructional puzzles for adding fractions and comparing the sum to one.

Puzzle 3, Entry Level

Domain: A fraction with denominator less than 100 and numerator less than 100 and less than twice the denominator.

Range: The signs "< ," " = " and "> ."

Rule: Type the sign which must replace the "/" to make a true statement.

Description for Student:

YOU ARE LEARNING TO ADD FRACTIONS WITH THE SAME DENOMINATOR AND COMPARE TO 1

USE < FOR LESS THAN
> FOR MORE THAN
AND = FOR EQUAL TO

1/16 BECOMES <
22/22 BECOMES =

Puzzle 4, Instructional Level

Domain: Two fractions as in Puzzle 3, with a "+" in between.

Range: Fractions with the same denominator up to 100 and numerators less than twice the denominator and less than 100.

Rule: The numerator of the range element is computed by summing the numerators of the two domain functions. The denominator is the same as the denominator of the fractions.

Description for Student:

YOU ARE LEARNING TO ADD FRACTIONS WITH THE SAME DENOMINATOR AND COMPARE TO 1

FRACTIONS MUST HAVE THE SAME DENOMINATOR

15/11 + 1/11 BECOMES 16/11
19/20 + 17/20 BECOMES 36/20

Figure 9 (Cont'd). Instructional puzzles for adding fractions and comparing the sum to one.

Puzzle 5, Criterion Level

Domain: Two fractions as in Puzzle 3, with a "+" in between.

Range: Fractions as in Puzzle 4 followed by one of the signs "< ," "= " or "> ."

Rule: The fraction is computed as in Puzzle 4, and the sign is that which must replace the "/" of the sum in order to make a true statement.

Description for Student:

YOU ARE LEARNING TO ADD FRACTIONS WITH THE SAME DENOMINATOR AND COMPARE TO 1

USE < FOR LESS THAN
> FOR MORE THAN
AND = FOR EQUAL TO

FRACTIONS MUST HAVE THE SAME DENOMINATOR

15/11 + 1/11 BECOMES 16/11 >
19/20 + 17/20 BECOMES 36/20 >
2/17 + 5/17 BECOMES 7/17 <

Puzzle 6, Post-Criterion Level

Domain: Fractions as in Puzzle 3, with a "+" in between.

Range: The signs "< ," "= " and "> ."

Rule: The fractions are summed as in Puzzle 4. The range element is the sign needed to replace the "/" of the sum in order to make a true sentence.

Description for Student:

YOU ARE LEARNING TO ADD FRACTIONS WITH THE SAME DENOMINATOR AND COMPARE TO 1

USE < FOR LESS THAN
> FOR MORE THAN
AND = FOR EQUAL TO

FRACTIONS MUST HAVE THE SAME DENOMINATOR

18/23 + 21/23 BECOMES >
1/4 + 2/4 BECOMES <

Figure 9 (Cont'd). Instructional puzzles for adding fractions and comparing the sum to one.

discovers how to add two fractions with common denominators, and then moves up to the terminal skill, puzzle five. The post-mastery skill involves generalization of the techniques.

Many dimensions of the program are interesting for evaluation purposes. We have used the program in the school for one year now, and have taught 150 children, a total of 530 different lessons. We collect data on every display and input the students see, and analyze them with the goal of improving the instructional algorithm and the sequencing of levels for the various objectives. We are presently undertaking more formal evaluation efforts; here we discuss the preliminary impressions from the data.

Results

We attempted to teach a full spectrum of objectives to third, fourth, fifth, and sixth graders. During the year, it has become clear that certain kinds of objectives lend themselves to presentation in this format more fully than others. Objectives with simple displays to present the domain elements work better than complex ones with many components. Thus, reducing a fraction is likely to be easier to teach than adding two fractions, simply because the display is less complex. Objectives that require much typing for either the range or domain will be difficult to teach because the students lack the typing facility required. Objectives in which complex calculations are required of the student are more difficult because such calculations are subject to error, and because such processing slows the rate of instruction. For example, the rule about placing decimal points in a three- or four-digit multiplication problem will be difficult to teach not because the rule is complex, but because the calculation is long and subject to error.

One of two outcomes occurs each time a student works on an objective in MATH FUNCTIONS: The student masters the criterion puzzle (mastery); or the student does not master the criterion puzzle (nonmastery). A further refinement of the outcome occurs for students who take the curriculum-embedded test (CET) as a transfer task: The student passes the CET (mastery-pass); or the student fails the CET (mastery-fail).

The frequency of the various outcomes is reported in Table 10. For the moment, only the overall totals are relevant. Of the total sample of 601 outcomes since Spring 1972, 529 (88 percent) were masteries. Some students (46) who mastered the objective did not take the CET transfer test. Of those who did take the test, 381 of 483 (78.9 percent) passed it, indicating a high transfer rate from the program-defined criterion task to the paper and pencil test.

TABLE 10

Outcomes on Objectives

	Nonmastery	Mastery	Mastery Fail	Mastery Pass
Overall	72 (12.0)	529 (88.0)	102 (21.1)	381 (78.9)
Spring 1972	9 (8.3)	99 (91.7)	23 (25.0)	69 (75.0)
Fall 1972	15 (14.7)	87 (85.3)	23 (33.7)	59 (66.3)
Winter & Spring 1973	42 (13.1)	279 (86.9)	56 (18.4)	197 (81.1)
Fall 1973	6 (8.6)	64 (91.4)	6 (9.7)	56 (90.3)

Note: Numbers in parentheses are percents.

Since the MATH FUNCTIONS program changed during the period of this evaluation, a partition of the results across time should reveal if changes affected the outcomes. Four natural divisions occur: Spring 1972 with only 20 objectives in a highly supervised setting; Fall 1972 with 120 objectives in a moderately supervised setting; Winter and Spring 1973 with 108 objectives in a lightly supervised setting; and Fall 1973 with 90 objectives in a lightly supervised setting. Outcomes for these periods are reported separately in Table 10. A Chi Square test of association of nonmastery versus mastery with time is not significant, making the best estimate of the pass rate 88 percent, the overall rate. The transfer rate does change over time ($\chi^2 = 15.9$, 3 df, $p < .05$).

The only drop in transfer rate occurs between Spring 1972 and Fall 1972 when six times the number of teaching sequences were available, and the amount of program supervision decreased. Notice a parallel drop in the mastery rate at the same time period. The mastery-pass rate in Fall 1973 is significantly greater than all other time periods ($\chi^2 = 5.6$, 1 df, $p < .05$), and is best estimated as 90.3 percent. The higher transfer rate results from revision of teaching sequences and selective elimination of objectives with high mastery-fail rates.

Summary

The results reported in the last section demonstrate that mathematics content can be taught in an environment which encourages problem solving. This section summarizes the procedure and the conclusions and points out directions for further work.

The FUNCTIONS program was designed as a problem solving environment. Features were included to teach skills such as stating the problem, gathering data, organizing data, using feedback, subdividing the problem, integrating subsolutions, and knowing when you are

finished. Important features include the organized display, the two modes (input and test) of interaction, and the lack of compulsion in the program.

Since mathematics contains many examples of functions, much of elementary school mathematics can be taught using the FUNCTIONS program. Special features of the Individualized Mathematics system such as self-paced learning, clear behavioral objectives and prerequisites, and mastery criteria make it especially adaptable to computerized, individual instruction. One hundred objectives of the Individualized Mathematics system were analyzed, and teaching sequences were prepared for use with the MATH FUNCTIONS program. Each teaching sequence consists of an ordered set of puzzles each of which is slightly more complex than its predecessor.

In more than one year of continuous use, students have mastered 88 percent of the objectives they studied, and have transferred their knowledge to a paper and pencil test 78.9 percent of the time. Improvements in the program and teaching sequence increased the transfer rate to 90.3 percent during Fall 1973.

While the overall success of the program in teaching mathematics content has been demonstrated, much work remains to be done. Some additional lines of investigation include: determining if student aptitudes interact with the treatment; developing a model of ideal problem solving in the program and demonstrating that students approach the ideal with experience; clarifying the methodology for creating teaching sequences; and, finally, developing additional programs of a similar nature to broaden the scope of application.

The success of this one effort at combining the teaching of problem solving with the teaching of mathematics content suggests that other similar efforts should be undertaken. What kinds of programs can be adapted to these uses? What other fundamental ideas occurring in mathematics can be exploited as functions have been in MATH FUNCTIONS? Additional work in this area is addressed toward answering these and other important questions.

COMPUTER-ASSISTED INSTRUCTIONAL SYSTEM: COMPUTER REQUIREMENTS

Robert J. Fitzhugh

During the past decade, there has been a continuing and growing debate in the computing field between the proponents of extremely large, utility-like computer systems each serving many users with diverse computing requirements and the proponents of small- and medium-scale systems each serving a more limited, homogenous population. The issues involved are complex and the final results are not in; however, the weight of the argument during the past few years seems to have drifted toward the small system concept although some in the computing field might disagree. Those who favor the large utility argue that there are economies of scale in computing just as there are in electric power generation and that a single, very large system is far more cost effective than multiple, small systems each with its own support staff and facility. In addition, it is argued that a large system can offer a powerful and varied range of computing services, some of which simply cannot be provided by a smaller system. Large-scale number crunching is an example. The small computer enthusiasts counter by pointing to the dramatic decline in the cost of small computers that is making them increasingly cost effective. Microcomputers the size of a pack of cigarettes are already available for under $200. The small computer enthusiasts argue that the very large systems that seek to be all things to all users have grown cumbersome, overly complex, and difficult to manage. A substantial amount of the memory and processing power of these machines is often consumed by massive general-purpose operating systems.

It is clear that this small versus large computer issue will not be settled for some time. In fact, a probable outcome is that neither will prevail and that there will be a mix of large and small computers, depending upon their application. At the Learning Research and Development Center, the application is computer assistance to the elementary school, and a major question is what is the most reasonable means of providing this computer assistance? Should it be through a tie-in to a large, remote, general-utility system, or can a small computer system located on site provide the required services?

The Learning Research and Development Center has maintained a computer research program since 1964 with the early years devoted to the development and use of a computer-controlled, on-line psychological research laboratory. Through developing and operating the On-Line Laboratory, considerable experience was acquired in the use of small computers, on their limits and on their use as an educational tool that can be made highly adaptive to individual differences. It was felt that existing small-computer-based educational systems had not fully exploited the powers of the small computer and that a system could be developed which could provide a broad range of computer services and which could satisfy the computing needs of a single school in a cost-effective manner.

To investigate this and other hypotheses about the roles a computer might play in a school, a small computer and associated peripherals were acquired and were housed in a modified van at a school site. The van provides a controlled environment and the option of easily relocating the system. The system requires approximately 250 to 300 square feet of floor space for a comfortable configuration and five tons of air conditioning. The current van configuration with office space for the support staff is shown in Figure 10.

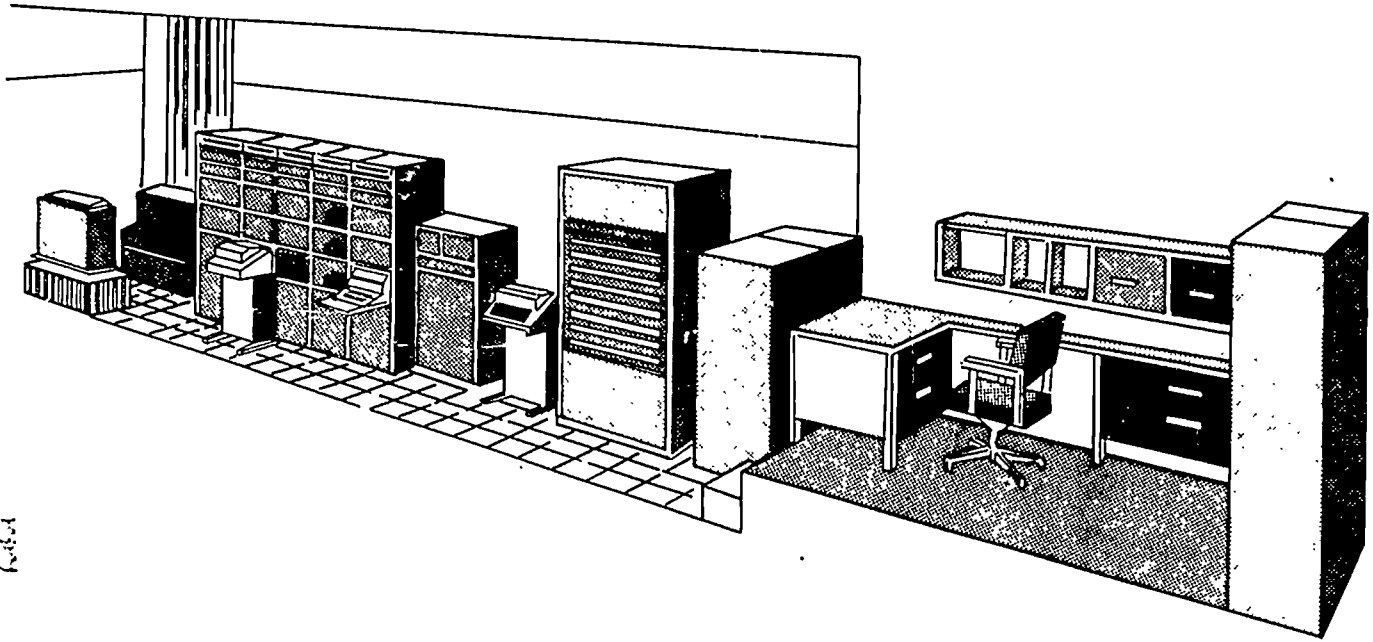

Figure 10. Computer van -- internal view.

The system was located at the school primarily to avoid the telephone system and telephone line charges although there is the less tangible factor of total local control over the computer resource which is not possible with a remote system. Although the line charges that would be incurred with the current terminal configuration would be large but not prohibitive, the system was located on site to retain the option of upgrading to more powerful terminals, possibly with very high data rates.

The major hardware components of the system were purchased from a number of vendors or were designed and constructed by LRDC as no single manufacturer at that time offered all of the items that were required. It is important to note, however, that no unusual modifications were made either to the computer or to the associated hardware so that the hardware remains comparable to other similar small- and medium-scale systems that are now readily available from computer manufacturers. Figure 11 depicts this hardware configuration. The important elements are the DEC PDP-15 computer with 32 thousand words of memory, a drum memory unit specially modified to increase the rate at which data can be transferred between it and the computer memory, two disk units which together provide 50 million characters of storage space, and a terminal controller capable of controlling up to 64 terminals of varying types and data rates both on site and remote telephone dial-up.

While the hardware is relatively standard, it is the time-sharing software, called ETSS for Experimental Time-Sharing System, that is the significant contribution by offering features, services, and interactive capabilities presently available only on large-scale computer time-sharing systems. Although many of the characteristics of ETSS were implemented in response to application-specific requirements, ETSS is a general-purpose time-sharing system rather than a special-purpose system

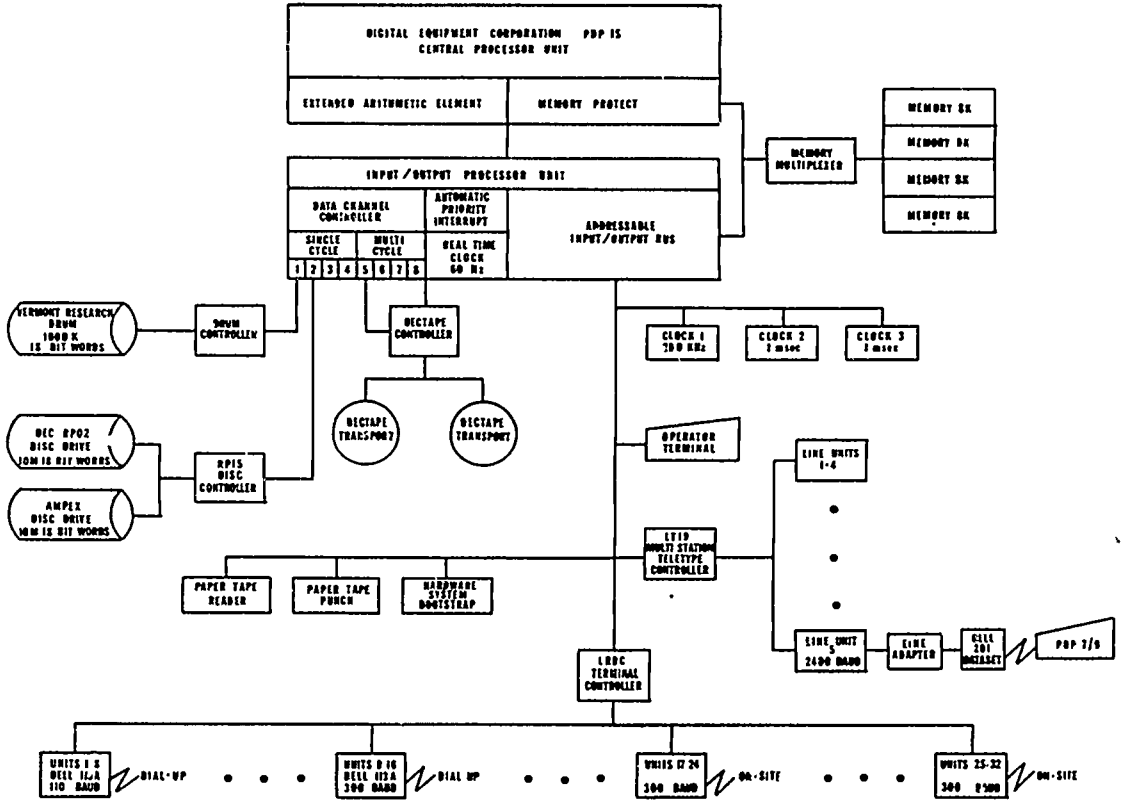

Figure 11. LRDC Experimental Time-Sharing System
Hardware Configuration

designed only for an educational environment. This decision to develop a general-purpose system places LRDC at variance with several similar projects at other institutions and was based on four primary considerations:

1. The system was developed primarily as a research venture rather than as a prototype of a system that might be replicated and produced commercially. As a research tool, the ability to meet changing requirements was essential. It was felt that the researcher should encounter as few artificial constraints as possible even though the price of generality is often increased programming difficulty.

2. The system must support several dissimilar applications. Although computer-assisted testing and instruction are nearly identical from a computer point of view, the data management and retrieval application posed an entirely different set of requirements. Rather than attempt to define and isolate common requirements that could be incorporated in a "special-purpose" system, the development of a general-purpose system was the cleanest solution.

3. The development of a special-purpose system would most probably preclude the use of commonly available and widely understood programming languages such as FORTRAN or BASIC. A programming language would need to be defined and implemented, this step was felt to be a major one which would draw heavily on available resources and detract from the primary objectives of the project. In addition, there was a reluctance to contribute another unusual, one-of-a-kind language to the existing Tower of Babel, particularly in the absence of a clearer understanding of its potential applications. The development, or at least the definition, of an application-specific language will be more appropriate once substantial in-school experience has been acquired.

4. The decision to develop a general-purpose, small computer time-sharing system was further reinforced by the belief that systems much like ETSS will be commercially available from computer manufacturers within several years. A number of systems are currently available that offer the language BASIC in a time-shared mode, and several multi-language systems are planned or under development. The growing success experienced in education markets by small-computer companies indicates that future more powerful systems such as ETSS will see wide use.

From a user standpoint, the system behaves very much like other general-purpose time-sharing systems now operating on large-scale computers although the system offers a number of interesting special features required by this application. A user, who might be a student or teacher within the school or a researcher or program developer accessing the computer from a remote site, locates an available terminal and identifies himself to the computer. Once the user has gained access, he is at a decision point and must specify the services he requires or the program he wishes to run. A student within the school might enter the name of an instructional or testing program which the system automatically retrieves from disk storage and initiates. A teacher or school administrator might wish to interrogate the student data base and would request that a data management program be started. A computer programmer developing new programs might ask to enter the Editor, one of the language processors such as FORTRAN, Focal, or T64 or any of a number of other utility or debugging programs.

From an internal programming standpoint, ETSS offers a wide range of programming options and a flexible, yet straightforward, input/output structure. The system masks device idiosyncrasies and offers

complete device independence. User programs interface with a virtual input/output structure in which "files" are referenced which may be assigned to any appropriate device or dataset.

For this educational application, the file structure and the ways in which data and programs may be stored and accessed are very important and several features were provided which are not commonly available on smaller, commercial systems. Programs and data may be stored in datasets which are "cataloged" in a directory unique to each user, or they may be stored in a common directory called the "Library." A user can access any program in the Library and may access a dataset in a user's private directory if that user has declared that this is permissible. In addition, any number of programs and users may simultaneously access common disk files without interference. For example, multiple users can interrogate and update a common student data base simultaneously and without interference, each unaware of the other's actions. Programs can test to determine whether or not a specified dataset is currently being accessed, and multiple programs can write to common "log" files which can later be sorted and analyzed. Facilities are also provided for the accurate timing of input/output events permitting response latency measurements to be made. Finally, the system has a number of internal, self-monitoring features which enable the system designers to collect and analyze data on system performance and use.

User programs running under ETSS may range up to 16K words or, roughly, 32K bytes in size. To put this limit in more operational terms, the largest instructional program written in FORTRAN that is currently operational is more than 1,500 FORTRAN statements in length with 4,000 additional words of dimensional arrays. The typical instructional program falls in the 900 to 1,300 statement range. This 16K limit

on program size has proved to be reasonable and one that has not inhibited or unduly constrained program development.

The disk requirement is more difficult to determine and varies greatly from program to program and from subject matter to subject matter. Programs in the language arts area in which textual material is displayed tend to have far greater on-line storage requirements than instructional programs in mathematics. Beyond citing examples, it is difficult to be more precise. The current range is from the 2.5 million characters required by a reading comprehension program to the nominal storage requirements of several mathematics programs that generate items for display to the student. Student history files can vary in size depending upon the position in the school year and the extent to which students have had computer exposure. For the Individually Prescribed Instruction environment, 4,000 characters per student have proved to be a good estimate for planning purposes. From a total project point of view, the 50 million characters of on-line storage the system now provides appear to be ample for most conceivable applications in a 300-student elementary school.

The experience of the coming year will provide the information required to refine these estimates and further define the computer requirements of the elementary school. However, with the experience base of nearly two years of successful operation in the school, it now seems clear that a small, local system is potentially a viable and cost-effective alternative to the large general utility.

References

Carlson, M., & Hsu, T. Formative evaluation of the computer-assisted testing programs 1971-72. Unpublished manuscript, University of Pittsburgh, Learning Research and Development Center, 1973.

Cox, R. C., & Graham, G. T. The development of a sequentially scaled achievement test. Journal of Educational Measurement, 1966, 3(2), 147-150.

Ferguson, R. L. Computer assistance for individualized measurement. Pittsburgh: University of Pittsburgh, Learning Research and Development Center, 1971. (Publication 1971/8)

Ferguson, R. L., & Hsu, T. The application of item generators for individualizing mathematics testing and instruction. Pittsburgh: University of Pittsburgh, Learning Research and Development Center, 1971. (Publication 1971/14)

Gagné, R. M. Varieties of learning and the concept of discovery. In L. S. Shulman & E. R. Keislar (Eds.), Learning by discovery: A critical appraisal. Chicago: Rand McNally, 1966. Pp. 135-150.

Glaser, R. Evaluation of instruction and changing educational models. In M. C. Wittrock & D. Wiley (Eds.), Evaluation of instruction. New York: Holt, Rinehart and Winston, 1970. Pp. 70-86.

Hively, W., II, Patterson, H. L., & Page, S. H. A "universe defined" system of arithmetic achievement test. Journal of Educational Measurement, 1968, 5, 275-290.

Hsu, T. et al. 1973 CAT reference manual. Unpublished manual, University of Pittsburgh, Learning Research and Development Center, 1973.

Hsu, T., Fox, J., & Lerner, E. Design of an on-line management system for instructional data. Unpublished manuscript, University of Pittsburgh, Learning Research and Development Center, 1973.

Hsu, T., & Pingel, K. A Bayesian approach in sequential testing. Paper presented at the annual meeting of the American Educational Research Association, Chicago, April 1974.

Hsu, T., & Sebatane, E. An empirical study of procedures for evaluating the quality of tests which are generated and administered by a digital computer. Paper presented at the annual meeting of the National Council of Measurement in Education, New Orleans, February 1973.

Individualized Mathematics Project Staff. The LRDC Individualized Mathematics Project: The scope and goals of the current development program. Unpublished manuscript, University of Pittsburgh, Learning Research and Development Center, 1971.

Lindvall, C. M., & Bolvin, J. O. Programmed instruction in the schools: An application of programming principles in "Individually Prescribed Instruction." In Sixty-sixth yearbook of the National Society for the Study of Education, Part II. Chicago: NSSE, 1967. Pp. 217-254. (Also LRDC Reprint 16)

Lindvall, C. M., & Cox, R. C. Evaluation as a tool in curriculum development: The IPI evaluation program. Chicago: Rand McNally, 1970.

Macready, G. B., & Mervin, J. C. Homogeneity within item forms in domain referenced testing. Educational and Psychological Measurement, 1973, 33, 351-360.

Roman, R. A. Teaching problem solving and mathematics by computer: An interim report. Pittsburgh: University of Pittsburgh, Learning Research and Development Center, 1974. (Publication 1974/15)

Suppes, P., & Morningstar, M. Computer-assisted instruction at Stanford, 1966-68: Data models and evaluation of the arithmetic programs. New York: Academic Press, 1972.

Wald, A. Sequential analysis. New York: Wiley, 1947.

Weiss, D. J., & Betz, N. E. Ability measurement: Conventional or adaptive? Minnesota: University of Minnesota, Department of Psychology, 1973. (Research Report 73-1, Psychometric Methods Program)