SOME EXPERIMENTS WITH AN ADAPTIVE SELF-IMPROVING TEACHING SYSTEM

Tim O'Shea*

Technical Report NL 18

August 1973

Natural Language Research for CAI
Supported by
The National Science Foundation
Grant GJ 509X

The Department of Computer Sciences
and CAI Laboratory
The University of Texas
Austin, Texas 78712

*Currently visiting from the University of Leeds, U. K.

ABSTRACT

An adaptive self-improving teaching system is described. The teaching strategy is expressed in terms of production rules. These rules are manipulated by a deduction system which uses a set of modally qualified assertions to select changes in the teaching strategy and statistical inference to evaluate these changes. Experiments with the system are described. These demonstrate that the system is an effective teacher and that the system can improve itself in a goal directed fashion.

ACKNOWLEDGMENTS

I am particularly indebted to my dissertation supervisor, Dr. Derek Sleeman, for the advice, encouragement and assistance he has given me throughout this project. The implementation of the program would not have been possible without the support of Professor Robert Simmons and the technical assistance and advice of Robert Amsler and Mabry Tyson.

Thanks are due to the administrative officers of the Austin Independent School District, in particular Mr. Schilhab. Much kind assistance was provided by Mrs. Shelton, Mrs. Waggoner and Mrs. Conway, mathematics faculty of Austin High Schools.

I would also like to thank Martha Stone, Michael K. Smith and Cathy Eckert for their patient editorial assistance and perceptive comments during the production of this report.

INTRODUCTION

Contemporary Intelligent CAI.

Compared with a teacher in a one to one teaching situation, existing tutorial CAI programs (see Lekan [1]) are severely limited. Among the limitations are the inability to allow students to make freely constructed responses in English and inadequate modelling of the student resulting in ineffective teaching (see Atkinson [2]). Another limitation is inadequate task representation with the result that the program cannot 'understand' the task at an appropriate level and cannot accept alternative correct or interesting responses (see Siklóssy [3]). CAI programs also lack the ability to profit from experience or to experiment with the teaching strategy.

Work has been carried out on all of these problems, some of the work indirectly in Artificial Intelligence and Educational Psychology and some directly in CAI (see Hartley & Sleeman [4], Koffman [5]). Work on the Natural Language Question Answering problem falls in the area of Artificial Intelligence and is being pursued by Simmons [6,7] and many others (see Walker [8]). Good student models have been produced for some task domains including work by Woods, Tait, et. al.,[9, 10, 11] and Suppes [12] on arithmetic. Programs which have reasonable task representations include Sleeman's work on NMR spectra [13] and Brown's on metereology [14].

The only attempt with which I am familiar, to build a teaching program with a 'self-improving' capability is that of Smallwood [15]. Smallwood's approach (discussed further in O'Shea [16]) is limited in a number of ways and is only applicable if the teaching strategy is represented entirely as a branching logic. Branching logics alone are usually not sufficient to represent the

teaching strategy of an adaptive teaching program for a complex task

A certain amount of work however, has been done in Artificial Intelligence on the problem of constructing programs with some sort of learning or problem-solving capability. The attempts to tackle this in general such as Newell, Simon & Shaw's GPS [17] or the Graph Traverser [18] have proved rather disappointing. However, more recent attempts to construct such programs for highly specific subject areas, for example by Waterman [19] and Winston [20], have had a fair degree of success.

The system described in this report is a development of Waterman's work on the machine learning of heuristics. Waterman implemented a program to play poker, where the playing strategy was expressed in production rules which were automatically manipulated to maximize the program's winnings. There are a number of parallels between a game of imperfect information (see Rapoport [21]) such as poker and the teaching situation. Furthermore, production rules provide an excellent means of representing teaching strategies.

Overview of System

For clarity it is best to consider the system as two completely separate and distinct programs which happen to interact. First, there is an adaptive teaching program which attempts to model a teacher teaching an individual student how to solve quadratic equations by the discovery method [22]. Secondly, there is a self-improving program which bears a similar relation to an educational psychologist observing the teacher and occasionally changing his teaching strategy in an attempt to improve his teaching performance.

The teaching program conforms to the structure for adaptive teaching programs put forward by Hartley [23]. It comprises of a vocabulary of teaching operations, a model of the student, a representation of the task and a set of

means-end guidance rules. Both the model of the student and the means-end guidance rules are expressed as sets of production rules. The teaching program keeps a record of the individual student during a teaching session, which it passes to the self-improving program at the end of the session.

Improvement is defined for the self-improving program with respect to four goals. The goals are: decrease the time taken by a student to complete the session, decrease the amount of computer time used, increase the number of students successfully completing the teaching session and increase the scores of students on a post-test. In order to select actions facilitating these goals the system utilizes a deduction system rather like that of Black [24]. The deduction system operates on a set of modally qualified assertions relating actions (changes executable on the set of production rules) to the goal states.

The use of modal operators (see Hughes & Creswell [25]) makes it possible to distinguish consequences which will certainly follow actions from consequences which will probably follow actions. Given the nature of the task domain it is reasonable to interpret these operators in a probabilistic fashion in order to select actions most likely to facilitate a chosen goal whilst least likely to operate to the detriment of the other goals. The system, having made an action, adds assertions to its set of assertions as a result of statistical inference on the student records subsequent to the action. As this action is a change which was executed on the teaching strategy, if it results in overall improvement the changed production rules are kept. Overall improvement is defined in terms of weights associated with the goals. Thus, the system attempts to improve the teaching strategy and at the same time adds to the set of assertions which constitute its knowledge of the teaching domain.

The implementation of the system and the experiments carried out with it are discussed below. The overall design and further rationale and justification of the techniques used are discussed elsewhere [16, 26].

THE ADAPTIVE TEACHING PROGRAM

Subject Area

A preliminary study was carried out in which twenty students were taught, without the aid of a computer, how to solve quadratic equations by the discovery method. From this and other work in the area [22], it became apparent that this particular subject matter and teaching style could be incorporated in an adaptive teaching program.

The objectives of a lesson are twofold. First to teach the student how to solve simple quadratic equations. Second to develop the student's problem-solving and reasoning skills. The student is presented with equations of the form

$$x^2 + c = b \cdot x$$

and has completely solved the problem when he has discovered that $x1 + x2 = b$ ("ADD" rule) and $x1 \cdot x2 = c$ ("TIMES" rule) where x1 and x2 are solutions to the equation. Subsidiary rules which assist in the solution of the problem are that x_i divides c, and that if $b = c + 1$, $x1 = 1$ and $x2 = c$ ("ONE" rule). The teaching strategy involves giving the student carefully chosen examples which increase the likelihood of a student discovering a particular rule. After the student has mastered a rule he is presented with examples for which his rule is not sufficient, interspersed with examples for which his rule is adequate. This represents an attempt to discourage him from rejecting his rule, while at the same time encouraging him to try to discover other rules.

Student performance varies greatly, but most students eventually start developing hypotheses about ways in which the problem can be solved and testing them out. Whether the student finally succeeds depends primarily

on his arithmetic skills and his ability to attend to the problem.

A number of factors make this a fairly demanding teaching task. The student may have a fair degree of success on the basis of a completely spurious rule. Students discover, lose and rediscover rules with great frequency. Students reject correct rules as a result of incorrect arithmetic or faulty logic.

Also the range of ability in students is very great, some 'see' the rules almost instantly, others need to be exposed to twenty examples before they develop an idea of what sort of rule is needed to solve the problem.

Consideration of these factors indicate that for a teaching program to be effective it should be able to adapt to individual students. Also, it is very hard to determine what would consitute an optimum teaching strategy. Therefore, it is very appropriate to attempt to build an adaptive self-improving teaching system for this subject area.

A Teaching Session

Initially, the program administers a pretest. This tests whether the student has the basic arithmetic skills necessary to master the task. Depending on the students' performance the program selects one of two modes of presentation, either algebraic equations or an equivalent word problem. Pretest protocols for a fast and slow student are given in Appendix I. The difficulty of the problems generated in the pretest depend on the students progress. If the student is unable to correctly answer simple problems after some coaching, the lesson is terminated. The pretest program uses a simple adaptive presentation strategy (expressed in production rules) and a simple task difficulty model (number of 'carries' to solve problem).

Having passed the pretest a simple quadratic equation is presented to

the student and both solutions demonstrated. The student then proceeds to guess solutions to a series of quadratic equations. A protocol is given in Appendix II. The program gives encouragement - 'try again,' 'well done' etc. - and the student is presented with a different example if he makes an excessive number of guesses or makes absurd guesses on any particular example. The order, duration and type of examples are determined by the teaching strategy. Depending on how the student progresses he is either tested on some hard examples and if successful congratulated on successfully mastering the problem, or he is told the session is taking too long and stopped.

Teaching Strategy

At one level the teaching strategy may be said to consist of all the decisions made in the construction of the program, from the format of the questions to the order of examples generated. What is described here is essentially the set of means-end guidance rules which relate the student model to the task representation. These rules form a presentation strategy for the different example types and co-ordinate the other components of the program, which are the example administrator, the example generator and the hypothesis tester. This is illustrated in Figure 1.

The presentation strategy is expressed as a set of production rules with associated partitions and statevectors. The initial set used is given in Appendix III. The 19 elements of the statevectors are variables associated with teaching strategy such as the current subgoal, the reason for the last action executed and estimates of the student's ability on the various subgoals. In a complete cycle of operation the existing statevector is parsed using the partitions (for more detailed examples see Appendix 3 and [16, 26]). The left hand side of the rules are then searched for the first match with

Schematic Diagram of Teaching Program

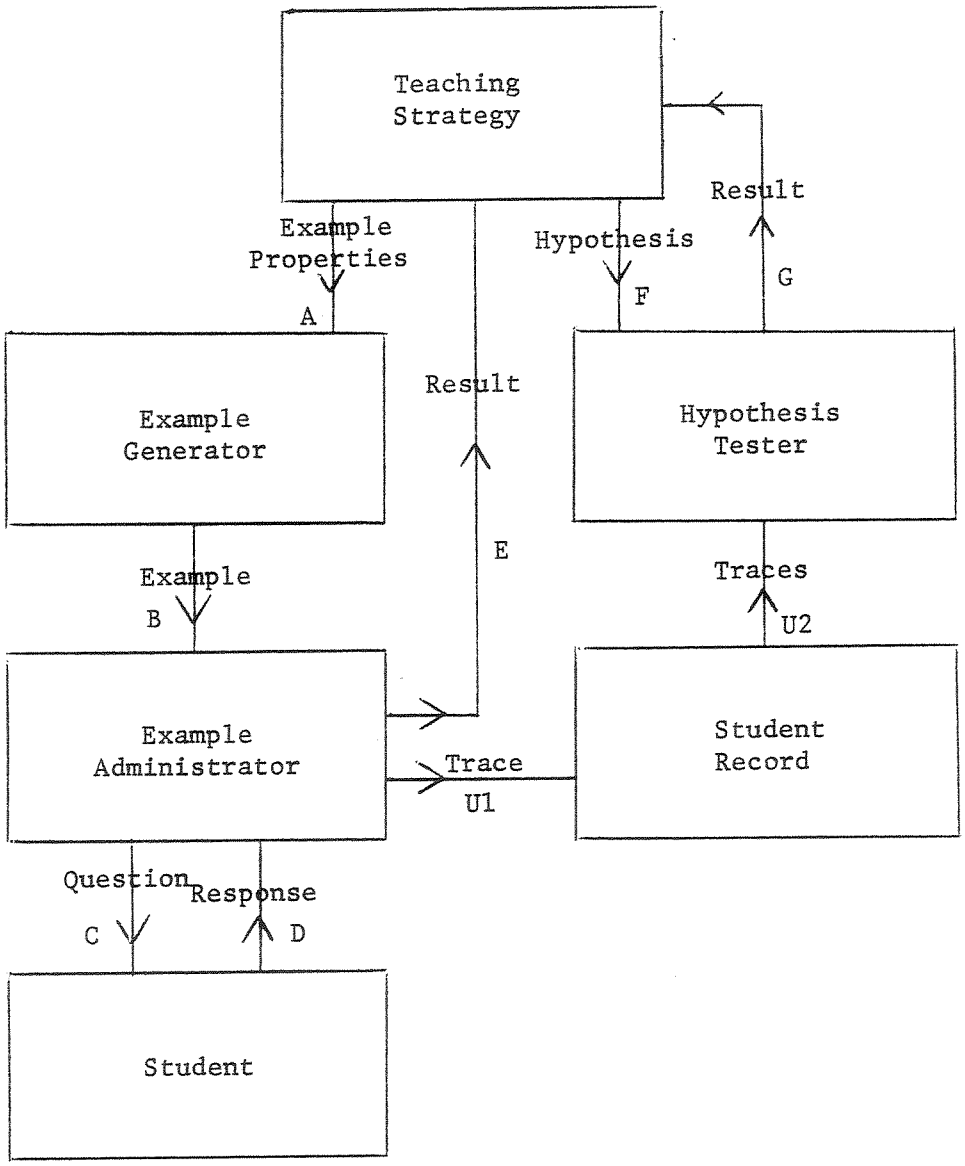


Figure 1

Note: i) A - G indicates flow of control

ii) U1, U2 indicates updating and accessing of student record respectively

the parsed vector. The corresponding right hand side, which is in fact a list of LISP [27] function calls, is then executed and the statevector is updated as a result.

The following functions are used on the right hand side.

i) (SETQ NAME X) - used to set parameters such as GUESSLIM: the maximum number of incorrect guesses permitted on any one example.

ii) (HYP RULE) - test the hypothesis that the student has mastered the rule in question. (see following section on Student Model).

iii) (GEN ((RULE1, HELP), (RULE2, HIN)) (PROPERTY)) - generate an example which will help the student acquire or apply RULE1, which will hinder application of RULE2 and for which PROPERTY is true.

iv) (CONTINUE N) - generate and administer 'N' more examples with the same properties as the previous one.

v) (GOAL NAME) - sets a flag in the statevector indicating the current subgoal.

vi) (DPRINTER NAME) - output the text labelled NAME.

The production rules can mostly be divided into those relating to the testing of hypothesis and those concerned with example generation. An example of the former is: ((1 A3 19 K1) ((HYP ≡ONE))).* This reads "if the program is in the hypothesis testing cycle and the current subgoal is the ONE rule, only test whether the student has acquired this rule." The rationale behind this rule is that testing the other hypotheses would be a waste of computer time. However, continual checking of all hypotheses would result in the program detecting incidental learning of other rules sooner. This is the kind of trade-off problem which the self-improving program tries to resolve.

---

*LISP notation is used throughout. The prefix ≡ indicates a literal atom, its absence indicates a variable name.

The rules relating to example generation may be considered as describing a partially ordered set of goals. The goals fall into four classes:

i) ADD, TIMES & ONE - get the student to master the rule named.

ii) ONETOADD, ONETOTIMES, ADDTOONE - get the student to master the second named rule without losing the first named rule.

iii) EASY - give the student problems solvable by inspection.

iv) HARD - give the student examples only solvable with complete mastery of the task.

The interrelation of the goals is illustrated by the directed graph given in Figure 2. Each node represents a goal, the outgoing arcs correspond to the production rules associated with the goal. For example GOAL TIMES may be left if conditions expressed by the right hand sides of any of the rules labelled T1 through T6 are met. (see Appendix 3) The conditions these correspond to are:

i) T1 - 'has the student mastered and successfully applied both the ADD and TIMES rule for the last three examples?'

ii) T2 - 'has the student mastered the TIMES rule but not the ADD rule?'

iii) T3 - 'is incidental learning on the ADD rule greater than learning on the TIMES rule?'

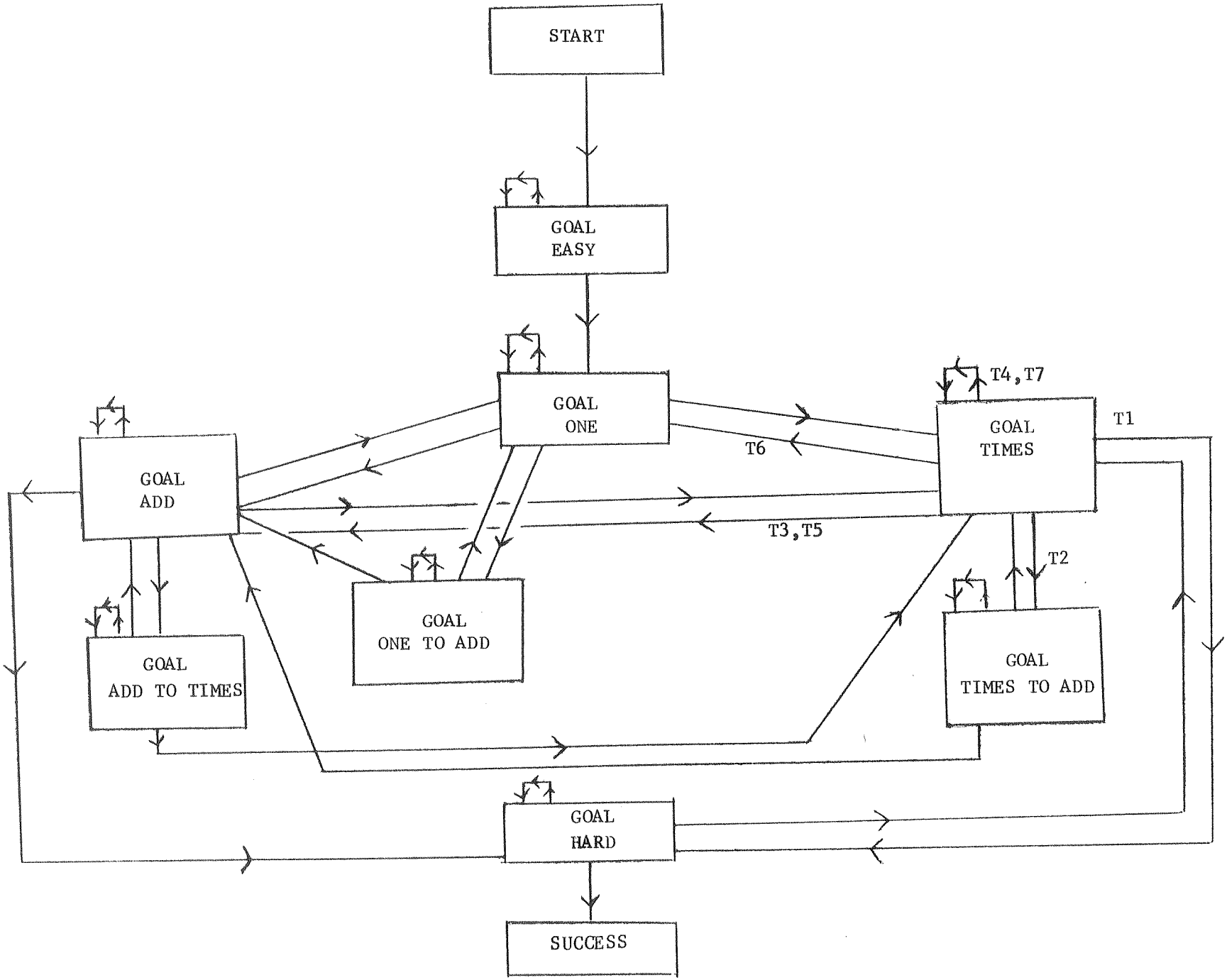
iv) T4 - 'is it "very possible" or "certain" that the student has mastered the TIMES rule?'

v) T5, T6 - check for incidental learning on ADD and ONE rules respectively.

In the event of none of these conditions being fulfilled, rule T7 insures that example generation for this goal continues.

Figure 2

DIRECTED GRAPH SHOWING INTERRELATION OF GOALS



The complete set of production rules expresses the presentation strategy in a completely explicit fashion amenable to automatic manipulation.

Student Model.

Whilst some assumptions about the student are implicit in the presentation strategy, inferences about the student's current state of knowledge are made in the hypothesis tester and this accordingly constitutes the major part of the student model.

The hypotheses that may be tested are as follows:

i) WORULE - that the student has no rule at all.

ii) WRULE - that the student has completely mastered all the rules.

iii) TIMES, ADD or ONE - that the student has mastered the particular rule in question.

Evaluating these hypotheses presents certain difficulties. In particular there is often not sufficient evidence in the students responses to enable the hypothesis tester to accept or reject the hypothesis.

This is partly resolved by returning one of the range of values: CERTAIN, VERY POSSIBLE, POSSIBLE, DON'T KNOW, POSSIBLY NOT, CERTAINLY NOT. This is very reasonable as there are many occasions in which it is not possible to distinguish which rule is being applied.

In addition some other factors such as the size of the problem space and previous extimated performance are used as well as the student's most recent responses.

Associated with each hypothesis is a set of production rules. These express a simple explicit decision procedure for each hypothesis. The elements of the statevectors include the number of guesses, the gap between

the correct solutions, the value of the previous hypothesis with respect to the rule in question and a measure of the size of the problem space. Further, each statevector has variables associated with the presence of particular key features in the student responses related to the hypothesis being evaluated. For example, the statevector for the TIMES rule includes a variable reflecting the proportion of guesses at the last example which were factors of C, where the equation is $x^2 + c = b \cdot x$. The complete set of rules for evaluating the hypothesis that the student has the TIMES rule is given in Appendix IV.

Task Representation

The program incorporates a model of task difficulty expressed in terms of a task difficulty matrix which relates teaching goals to example features.

These features are weighted relative to each other as to how much they help or hinder acquisition of the various subgoals. The task difficulty matrix is given in Appendix V.

The example generation procedure operates as follows. Using the task difficulty matrix, the example features are ordered with respect to their relation to the current subgoals.

The features are then sorted according to two criteria. Firstly, incompatible features are eliminated, an example of a pair of incompatible features is 'that one of the solutions to the problem be 1' and that 'none of the solutions be 1.' Then currently impossible features are eliminated. Consider the pair of features 'that the examples have not been previously generated' and 'that both solutions have numerical values less than 5.' After a number of problems have been generated it may be impossible to find an example with these features. The table of examples is then searched for

the best fit to the remaining set of features; if no fit is found, the set of features is added to the list of sets of features now impossible to obtain. Then, the feature with the smallest relative weight is dropped and the search repeated.

In an earlier version of the program a non-deterministic select function (see Floyd [28]), was used to generate examples. This proved to be very inefficient and table look-up was found to yield equally effective examples for considerably less computer time.

THE SELF-IMPROVING PROGRAM

Changing Production Rules

The number of possible changes which may be executed on a non-trivial set of production rules is very large. However, the changes which are meaningful within the context of changing the strategy of a teaching program are few and can be defined without too much difficulty.

The self-improving program incorporates an amender which takes as arguments a list of function calls. These define a change or changes to be executed on the set of production rules. Changes may be executed on the condition or action part of a production rule, or on the ordering of the goals explicitly represented in the production rules.

There are two types of functions: a) those which identify the production rules to be changed, and b) those which identify the changes. The functions are:

a) i) PRESENT, ABSENT. (PRESENT X) returns true for a given production rule if X in an element of the condition side on which a restriction is placed or if X is a member of the list of actions on the right hand side. ABSENT is similarly defined.

ii) LESS, GREATER, EQUAL, NOTEQUAL. (LESS X Y) returns true for a given production rule if (PRESENT X) is true and if the restriction on X is such that it is less than Y numerically. Where X is a variable associated with a certainty value from the student model it returns true if X is less certain than Y. GREATER, EQUAL, NOTEQUAL are similarly defined.

b) i) RAISE, LOWER. (RAISE X) is interpreted as raise the threshold or parameter identified by X. The magnitude of the increase is determined by a parameter associated with the running of experiments (see following

section). LOWER is similarly defined.

ii) BEFORE, AFTER. (BEFORE X) may be interpreted in two ways. If X identifies a goal, (BEFORE X) identifies the production rules which correspond to the incoming arcs of the node corresponding to the goal. (See Figure 2). If X identifies an action, (BEFORE X) identifies the position immediately before X in the sequence of actions on the right hand side of the production rule. AFTER is similarly defined.

iii) INSERT, REMOVE, REPLACE. (INSERT X) is defined as adding the appropriate condition, as defined by X, to the left hand side, or adding the action X to the right hand side. REMOVE and REPLACE are defined similarly. When X is a goal the interpretation of these functions is illustrated in Figure 3.

The arguments of the functions in the list of function calls for the amender must be either all related to conditions and actions or all related to goals. This restriction was imposed solely for ease of implementation, and could be removed in a subsequent implementation.

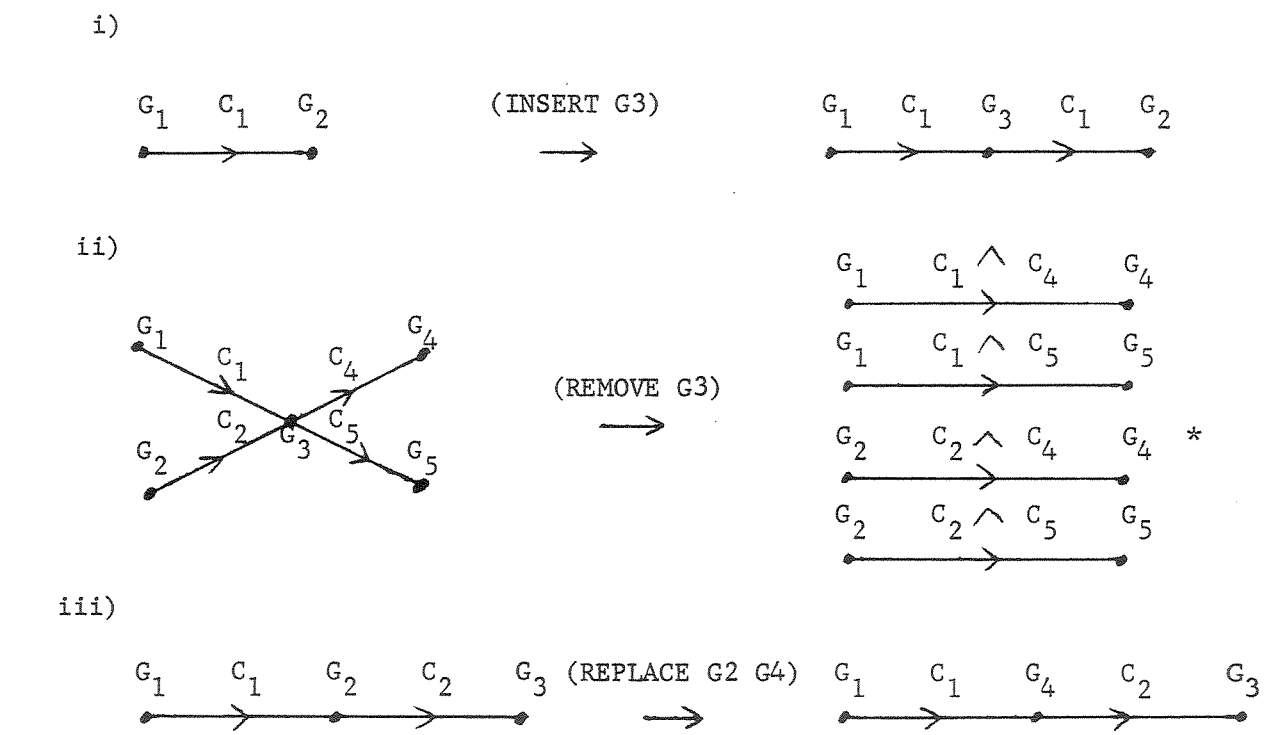
Where the arguments are related to conditions and actions the amender operates by using the functions listed under a) and interpreting them as PRESENT to identify the rules to be changed. Then, using the commands listed under b), the changes are executed.

If the arguments are goals, the production rules to be changed are identified by interpreting BEFORE and AFTER as described and by interpreting any of (REMOVE X), (REPLACE X Y), or (INSERT X) as both (BEFORE X) and (AFTER X). The changes are then executed.

It is possible to express changes to be executed on the set of production rules succinctly and without undue difficulty.

Figure 3

Note: G_i are goals. C_i are conditions on the goals and correspond to the left hand sides of production rules.



*$C_i \wedge C_j$ stands for the condition (C_i AND C_j) with the restriction that if $C_i \wedge C_j$ is impossible, C_i takes precedence over C_j.

Selecting Changes

In order to identify which changes should be executed on the set of production rules, a deduction system, rather like that of Black [24], is used. The deduction system operates in the following fashion:

i) Given a desired goal such as 'decrease computer time,' it returns a list of sets of actions which may achieve this goal.

ii) Given a list of actions it returns a list of possible consequences of these actions.

Thus the system behaves like a set of PLANNER antecedent and consequent theorems. (See Hewitt [29]). The actions referred to above will be changes executable on the set of production rules.

The deduction system operates on a set of assertions. Two types of assertions are distinguished; causal assertions and definitional assertions. The assertions used by the system are given in Appendix VI. Causal assertions are qualified by the operators "CERT" or "POSS." Two examples of causal assertions are:

i) (CERT (REMOVE MINOR) (DECREASE (STUDENT TIME)))

ii) (POSS (LOWER EXLIMIT) (DECREASE SCORES))

The first assertion may be interpreted as 'for any set of production rules relating to the adaptive teaching program, executing (REMOVE MINOR) will result in the decrease of student time.' The second assertion may be interpreted as 'for some set of production rules relating to the adaptive teaching program, executing (LOWER EXLIMIT) will result in the decrease of scores.'

Causal assertions are of the general form (OPERATOR LSIDE RSIDE) and have the following restrictions. LSIDE may be of the form (AND T_1 ... T_n).

T_i must be a predicate applied to arguments which are not predicates, i.e., the arguments must be names. RSIDE may be of the form (OR T_1 ... T_n). Note that (OPERATOR (OR X_1 X_2) Y), which is excluded, may be rewritten (OPERATOR X_1 Y), (OPERATOR X_2 Y). Similarly, we may rewrite (OPERATOR X (AND Y_1 Y_2)) as (OPERATOR X Y_1), (OPERATOR X Y_2).

The definitional assertions are qualified by the operator EQUIV. Two examples of definitional assertions are:

i) (EQUIV (OR (GOAL ≅EASY) (GOAL ≅HARD)) MINOR)

ii) (EQUIV (AND (STUDENT SCORE) (POST SCORE)) SCORES)

The first assertion states that anything which is true for MINOR is also true for (GOAL ≅EASY) or (GOAL ≅HARD) and vice versa. The second assertion states that anything which is true for (STUDENT SCORE) and (POST SCORE) is true for SCORES and vice versa.

The definitional assertions reduce the number of causal assertions needed. Where a causal assertion is true in a number of instances, a name such as MINOR can be defined.

The use of definitional assertions removes the possibility of endless deductions as a result of repeated substitution. Definitional assertions are used in the deduction procedure to match the predicates of causal assertions but are not substituted for them. As in our context there could be no possibility of having both the causal assertions (OPERATOR X Y) and (OPERATOR Y X) repeated substitution cannot occur.

Given a goal such as (DECREASE (COMPUTER TIME)), the deduction procedure is as follows:

1. Perform a search for matches on the right hand sides of the causal assertions with the goals. The definitional assertions are used to check for

matches. For example, MINOR will match (GOAL ≡EASY) if we have the definitional assertion (EQUIV (OR ((GOAL ≡EASY) (GOAL ≡HARD)) MINOR).

2. The corresponding left hand sides are detached and step 1 is repeated on them for non-executable predicates. Executable predicates correspond to the functions listed in the previous subsection such as REMOVE and BEFORE.

3. The lists of executable predicates thus obtained are compared to find those lists $\{L_i\}$ which used the least number of causal assertions involving the POSS operator.

4. For each list of executable predicates, L_i, left to right matches are then carried out and all the predicates, $\{M_j\}$, operating on 'goal' variables, such as (COMPUTER TIME), are detached.

5. The predicates $\{M_j\}$ identified in 4 are divided into those resulting in the achievement of goals $\{G_k\}$, such as (INCREASE (STUDENT SCORE)), and those resulting in the deterioration in the progress towards goals $\{B_l\}$, such as (INCREASE (COMPUTER TIME)). A measure mu relating to the likelihood of the overall achievement of goals is computed in each instance.

$$mu = \sum_{k=1}^{n} P(G_k) - \sum_{l=1}^{m} P(B_l)$$

$P(M_j)$ is computed as $0.75\uparrow POSS(M_j)$ where $POSS(M_j)$ is the number of causal assertions involving the POSS operator used in the deduction of M_j from L_i.

6. The set of executable predicates, L_i, associated with the maximum value of mu is then selected.

7. The arguments of the executable predicates, L_i, are then tested against a list of primitive arguments. These primitive arguments are the

names of the variables of the state vector and the names of function calls which may be on the right hand side of a production rule. Where a primitive argument is not found, the definitional assertions are used to instantiate one. For example, (REMOVE MINOR) is changed to (REMOVE (GOAL =EASY)).

8. The set of executable predicates are then executed.

The operation described in step 3 and the measure mu defined and used in steps 5 and 6 are heuristics used to identify sets of executable actions likely to result in the achievement of goals. For a given assertion, (POSS X Y), and a given set of production rules, there may well be a probability value which could be associated with the occurrence of Y given the execution of X. However, this is not known prior to executing X, and this probability value will vary with the set of production rules X is executed on. Further, we do not have a measure for the magnitude of Y. But we must choose between actions with possible or certain 'good' or 'bad' effects. Also, examining the assertions, it seems desirable that 'possibly possibly' be weaker than 'possibly.' Accordingly, (POSS X Y) has been arbitrarily assigned the probability 0.75 in the evaluation of mu. As will be discussed below, this procedure does not eliminate the possibility of executing various executable predicates, but determines the order in which they are executed.

Whilst some aspects of the formulation described above, such as the CERT and POSS operators are adapted from modal logic, (See Hughes & Cresswell [25], McCarthy & Hayes [30]), the formulation cannot be identified with any of the recognized modal calculi. It is essentially a 'common sense' approach to the problem of reasoning about actions, restricted to the particular context of selecting changes to be executed on a set of production rules.

Evaluating Changes

The changes executed on the set of production rules are evaluated by statistical inference on the resulting changes in student performance. A variable is associated with each of the four goals of the program. The variables are: whether a student successfully completes the teaching session - SS, the score on post test - PS, the amount of student time used - ST, and the amount of computer time used - CT. The goals are to increase SS and PS and to decrease ST and CT. These goals conflict, and a further variable which is a measure of overall improvement is defined as:

$$OI = W_1 \cdot SS + W_2 \cdot PS - W_3 \cdot CT - W_4 \cdot ST$$

The weights W_i are fixed according to the relative values of the goals.

T - tests are carried out after each additional student has been taught. The T-test is used to compare the mean score for the variable prior to the change in production rules to the mean score after the change. If there is a significant change using a 90% confidence interval in any of the variables, the change is then evaluated to determine if it should be permanently incorporated in the set of production rules. If the mean score of OI has increased, the change in the set of production rules is kept. If there has been a decrease in OI, the set of production rules prior to the change is restored and the student records subsequent to the change are deleted. The latter operation insures that after a series of changes resulting in decreases in OI the mean for OI is not depressed. If this was not done, after a series of bad changes, a change with a small bad effect might be incorporated into the set of rules.

Small sample statistics and evaluation after each student were necessary in view of the number of students on which it was expected to run the program. If the program were run on a large number of students it would be possible to

use more sensitive statistical tests.

Before any changes may be executed, the teaching program must be run on sufficient students to provide a data base with respect to which the changes may be evaluated.

After a change is evaluated, causal assertions are added to the set of assertions operated on by the deduction system.

Adding Assertions

Suppose at step 6 of the deductive procedure we have isolated the executable predicate, (REMOVE (MINOR)), and then after applying (EQUIV (OR (GOAL≦EASY) (GOAL≧HARD)) we execute the action (REMOVE (GOAL≦EASY)). If, for example, the consequence is an increase in computer time, then the following assertion is added to the set of causal assertions:

(CERT (REMOVE (GOAL≦EASY)) (INCREASE (COMPUTER TIME))). In addition, the generalization:

(POSS (REMOVE MINOR) (INCREASE (COMPUTER TIME)))

is added to the set of causal assertions.

Adding assertions in this manner has the effect of inhibiting the use of changes found experimentally to result in the deterioration of the teaching performance and of making more likely the use of changes found experimentally to result in the improvement of teaching performance.

An attempt may be made to execute an executable predicate which is in fact impossible to execute. For example, there may be no (GOAL≦EASY) for the command (REMOVE (GOAL≦EASY)) to operate on. In this instance, the assertion:

(CERT (REMOVE (MINOR)) IMPOSSIBLE) is added to the set of assertions.

Assertions may also be added to the set of assertions at any time by a

person observing the program and wishing to give it some 'advice.'

Limitations

There are a number of severe limitations in the deduction schema described. Consider the causal assertion (CAUSE X Y) which could be interpreted as 'for the current set of production rules, executing change X will result in the occurrence of event Y.' In many cases, particularly when assertions are being added, assertions of this form would seem to be more natural and more useful than assertions of the form (POSS X Y). The difficulty arises because of what is known as the 'frame problem' (See McCarthy & Hayes [30]). In our case, while (CAUSE X_1 Y) may be true for a given set of production rules, we have no way, apart from experiment, of determining if it is still true after the execution of say, change X_2. Accordingly, the weaker (POSS X_1 Y) is used.

However, in some instances, for example, if with the current set of production rules, a change has a statistically significant effect on a variable, we wish to distinguish that change. In this circumstance an assertion of the form (CERT X Y) is added. This solution is very unsatisfactory in that this use of CERT does not strictly conform to the interpretation of CERT given earlier.

A related difficulty arises with 'impossible' changes. In our current formulation, a change X_1 executed upon the set of rules may result in the impossibility of applying change X_2. X_2 may then be tagged IMPOSSIBLE to prevent the deduction system from repeatedly deducing that this is the best candidate for execution. But a further change X_3 may result in X_2 now being possible and in fact the best candidate. In our existing implementation it is not possible to remove the tag IMPOSSIBLE from X_2.

The program is also liable to 'hill-climbing' problems in that at any

time it always executes the most likely change. It may be the case, however, that the execution of some change with a small 'bad' effect will make possible the execution of a sequence of changes with very 'good' effects. The program may, therefore, never carry out some optimum sequence or combination of changes even though each particular change may be described in some causal assertion. Work is in progress on the possibility of introducing the CAUSE operator and on the use of a simple planning algorithm to estimate the consequences of sequences of changes.

Despite its limitations, the deduction system has proved adequate for representing and performing deductions with simple causal assertions relating to the changes in the teaching strategy. For example, the assertions given in Appendix VI include very general statements such as:

(POSS (REPLACE MAIN MAIN) (CHANGE SOMEGOAL))

which indicates the changing of the order of main goals may have some effect, and highly specific statements such as:

(POSS (RAISE ENCRATE) (INCREASE (POST SCORE)))

which indicates that increasing the rate of encouragement may possibly increase the post score.

Using the system described, it proved possible to select, execute, and evaluate a number of non-trivial changes in the set of production rules.

EXPERIMENTS WITH THE SYSTEM

Implementation Details

The system used for the experiments was implemented in LISP [27] and run interactively under TAURUS, a time-sharing system on the CDC 6600/6400 at the University of Texas at Austin.

The system comprised five overlays each occupying 33K and four disc resident files. The operation of the system is illustrated in Figure 4. Wherever possible the LISP code had been compiled (See Greenawalt [31]) for speed. An entire lesson used on average about 50 seconds of processor time. When the deduction system and amender were used, a further 60 to 90 seconds of processor time were used.

The response time of the teaching program ranged from two to five seconds when the program was run between 8:00 a.m. and 10:00 a.m. Occasionally, teaching sessions were held later in the day, when there were many more users on the time-sharing system. The response time could then be as poor as one to two minutes.

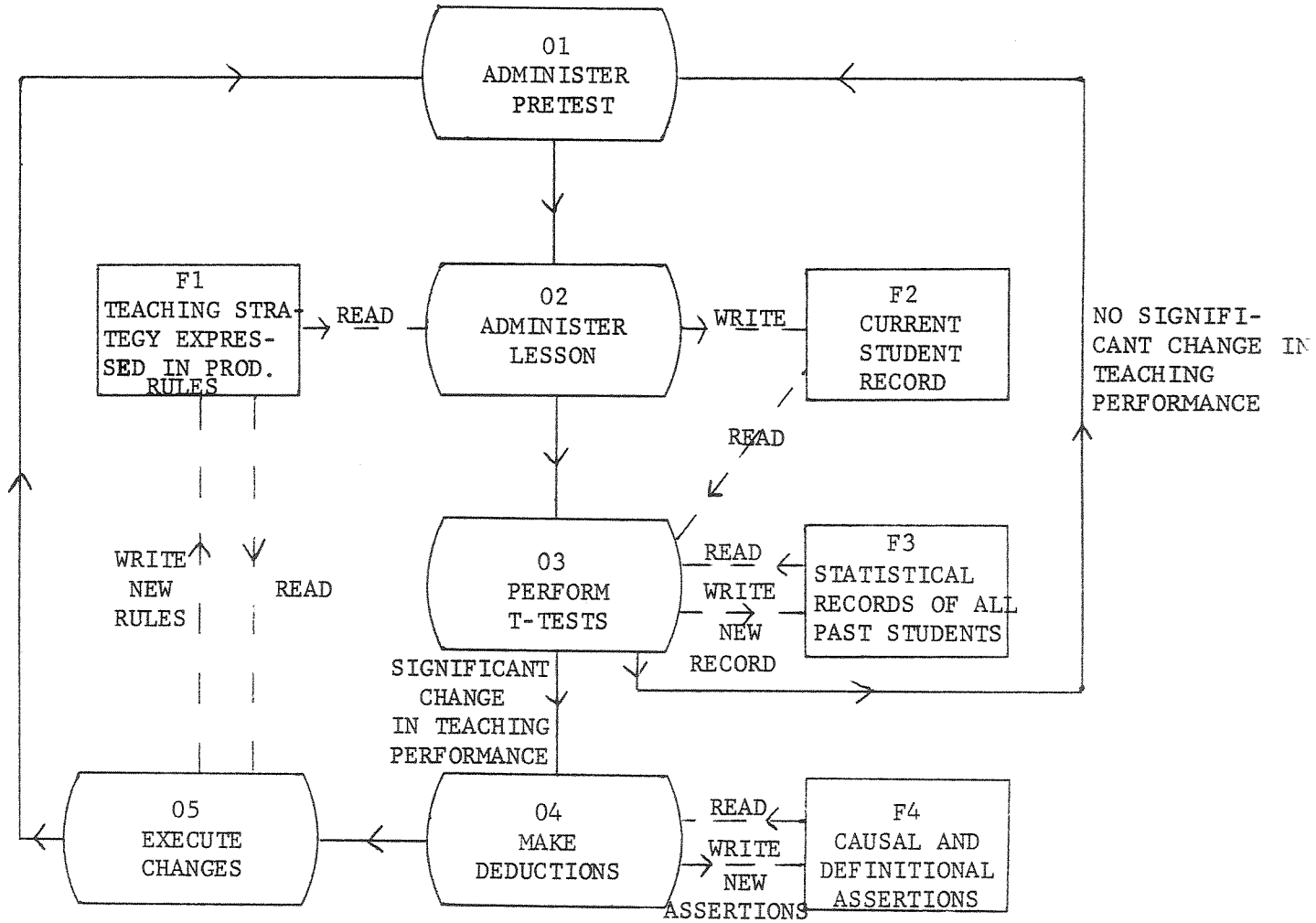
Experimental Procedure

The subjects for the experiments were fifty-one high school students aged between thirteen and fifteen years. Although these students did not belong to the age group (nine to eleven years) for which the teaching program had been designed, they could all be identified as having difficulty with mathematics. The students had all failed at least one high school mathematics course. None of the students were able to solve quadratic equations at the start of the lesson.

The lesson was presented to the students on an interactive terminal located at the school. The lessons lasted on average about an hour, during

Figure 4

Operation of System



Note:
i) 01 - 05 are overlays.
ii) F1 - F4 are disc resident files.
iii) ——→—— represents transfer of control.
iv) — READ → — represents reading a file.
v) — WRITE → — represents writing a file.

which time the student attempted an average of 24 problems. The day after the lesson the students completed a written post-test comprising 6 problems to be completed within ten minutes.

The teaching program without the self-improving program was used to give lessons to the first twenty students. The students records so collected, provided the basis for evaluating the effects of subsequent changes in the teaching strategy on teaching performance.

The self-improving program was initialized as follows:

i) The goals namely: increase student score, increase post-score, decrease student time and decrease computer time were weighted 2:2:1:1.

ii) The magnitude of changes executed by the (RAISE X) and (LOWER X) function calls was set at 50% of the current value of X.

iii) The confidence interval for the T-tests was set at 90%.

iv) The maximum interval between changes in the teaching strategy was set at seven lessons. That is if no statistically significant change had occurred after seven students had been run a new change was selected and executed. The previous change was to be incorporated if OI, (the measure of overall improvement) had increased, otherwise it was to be removed.

Had a much larger number of students been available it would have been possible to select a smaller magnitude of change in ii), a greater confidence interval in iii) and dispense with iv). For the purposes of the experiment i) - iv) were set to insure that the self-improving program made a reasonable number of changes without being completely erratic.

The system incorporating the self-improving program was used to administer lessons to the remaining thirty-one students.

Results

The self-improving program carried out five 'experiments.' These are briefly summarized below.

i) With the goal of decreasing the amount of student time the program selected and executed (REMOVE (GOAL =EASY)). The result of this change was that the teaching program no longer presented two easy introductory problems. No statistically significant change in teaching performance was detected, but as the value of OI had increased the change in the teaching program was kept.

ii) With the goal of increasing the post-test score (REPLACE HYPS (HYPALL)) was executed. This change resulted in the teaching program testing all its hypotheses about the student after administering each example. Statistically significant increases in post-test score and decreases in computer time were detected. Appropriate assertions were added to the set of assertions and as the value of OI had increased the change in the teaching strategy was kept.

iii) With the goal of increasing the student score, the program tried to execute (REPLACE HYPS (HYPALL)). As a result of ii), this was not possible and the assertion (CERT (REPLACE HYPS (HYPALL)) IMPOSSIBLE) was created. It then executed (RAISE ENCRATE) which increased the amount of encouragement given the student. This resulted in a significant decrease in post-test score and a decrease in OI. The program removed the change from the teaching strategy.

iv) With the goal of decreasing computer time the program executed (LOWER TIMESCH). This was equivalent to lowering the criterion for deciding that the TIMES rule had been completely mastered. A significant decrease in

computer time was detected. As OI had increased, the change was kept.

v) With the goal of decreasing student time (LOWER ADDCH) was executed. The effect and results were the same as in iv).

In each case assertions were created and added to the set of assertions. After being run on the thirty-one students the program had added nine assertions to its set of assertions. Comparing the teaching performance of the program over the first twenty students and the last ten there was a decrease in mean computer time of 26% which is significant using a T-test with a 99% confidence interval.

The mean student score, mean post-test and mean student-time all improved. None of these changes, however, were statistically significant using a 95% confidence interval.

Discussion

The system was a very effective teaching program. Before using the program most of the students characterized themselves as very poor at mathematics and expressed very low confidence in their ability to learn mathematics. However, 80% of the students discovered the coefficient rules for solving the equations and could solve equations such as $x^2 + 48 = x * 19$ by inspection, after completing the lesson.

The scores on the post-test were mostly high, the mean score being 86%. Most of the students enjoyed using the program and were very enthusiastic about both using the terminal and the teaching style of the program. A selection of student comments is given in Appendix VII.

With respect to the four goals set for the system, an improvement in teaching performance was obtained. This improvement was mostly in terms of a decrease in computer time. The main reason for this was that with the

number of students available it was difficult to detect improvement in the other three variables. The post-test score and student score were near their maximum values and the variable student time had a large standard deviation.

In addition to the improvement in teaching strategy the program's set of causal assertions had been refined. An incorrect assertion had been selected, evaluated and corrected. Several other assertions relating to the results of experiments had been created. In fact the fifth experiment was partly a result of the assertions created after the fourth experiment. Given the general nature of some of the assertions the system could probably have indefinitely executed changes in the teaching strategy.

There are a number of weaknesses in the existing system which would become apparent if large numbers of students had been taught. The set of assertions would continue to grow and become unwieldy and expensive in computer time to evaluate. Also the system having tried out all its 'likely' changes would start trying out the 'unlikely' ones. This would result in frequent short-term deteriorationsin teaching performance.

If a large number of students were to be taught it would be necessary to find ways of removing redundant assertions from the set of assertions. Constraints could be applied to the set of teaching strategies experimented with to prevent severe deterioration in teaching performance (See O'Shea [26]).

The system described is capable of improving its teaching performance. The potential for improvement is dependent on the set of assertions used initially and the limitations of the teaching style. Further possible extensions of the system are discussed elsewhere [16].

CONCLUSIONS

It is possible to construct an adaptive self-improving teaching system for a complex teaching task. The essential features of the system are the use of production rules as a means of representing the teaching strategy, the use of modally qualified assertions which express possible changes and their relation to the overall goals of the system and the use of statistical inference to evaluate the changes carried out. The approach described is applicable to any teaching task where the teaching strategy may be formulated as a set of production rules and where observable measures for the overall goals can be defined. Work is currently in progress to identify the classes of teaching tasks whose associated teaching strategies may be amenable to expression as sets of production rules and into ways of strengthening the deduction procedure.

REFERENCES

1. H. A. Lekan, "Index to computer assisted instruction." Harcourt Brace Inc., New York. 1971.

2. R. C. Atkinson, "Ingredients for a theory of instruction." American Psychologist, October, 1972, p. 921-931.

3. L. Siklòssy, "Computer tutors that know what they teach." Proc. FJCC, 1970, p.251-255.

4. J. R. Hartley and D. H. Sleeman, "Towards more intelligent teaching systems." International Journal of Man-Machine Studies, 1973.

5. E. Koffman, "Artificial intelligence and automatic programming in CAI." Proceedings of Third Joint International Conference on AI, 1973.

6. R. F. Simmons, "Linguistic analysis of constructed student responses in CAI." Technical report TNN-86, University of Texas at Austin, 1968.

7. R. F. Simmons, "Integrated systems for natural language processing." Technical report NL-8, University of Texas at Austin, 1972.

8. D. E. Walker, "Automated language processing." Annual Review of Information Science and Technology, Volume 8. Carlos A. Cuadra, editor, 1973.

9. P. Woods, J. R. Hartley, K. Lovell, and D. H. Sleeman, "The teaching of arithmetic using a computer based system." In Mann and Brunstrom's Aspects of Educational Technology, III. Pitman, London, 1969.

10. P. Woods, and J. R. Hartley, "Some learning models for arithmetic tasks and their use in computer based learning." British Journal of Educational Psychology, Vol 41, Part I, 1971.

11. K. Tait, J. R. Hartley, & R. C. Anderson, "Feedback procedures in computer assisted arithmetic." British Journal of Educational Psychology (in press).

12. P. Suppes, M. Jerman & D. Brian, "Computer assisted instruction." Stanford's 1963-1966 Arithmetic Programs, Academic Press, New York, 1968.

13. D. H. Sleeman, "A problem solving monitor for a deductive reasoning task." Technical report NL-17, University of Texas at Austin, 1973.

14. J. C. Browne & R. R. Burton, "A model driven question answering system for a CAI environment." Technical report 13, University of California at Irvine, 1972.

15. R. D. Smallwood, "A decision structure for teaching machines." MIT Ph.D. Thesis, MIT Press, 1962.

16. T. O'Shea, "Self-improving teaching systems." Dissertation (in preparation) to be submitted to University of Leeds.

17. G. Ernst and A. Newell, "GPS, a case study in generality and problem solving." ACM Monograph Series, Academic Press, New York, 1969.

18. D. Michie & R. Ross, "Experiments with the adaptive graph traverser." in B. Meltzer & D. Michie (eds.) Machine Intelligence 5, p. 301-318, New York, 1970.

19. D. A. Waterman, "Generalization learning techniques for automating the learning of heuristics." Artificial Intelligence, Vols. 1,2, 1970.

20. P. E. Winston, "Learning sturctural descriptions from examples." Ph.D. Thesis, Project Mac TR-76, 1970.

21. A. Rapoport, "Two person game theory." University of Michigan Press, 1966.

22. L. S. Schulman, and E. R. Keisler, "Learning by discovery." Rand McNally and Co., Chicago, 1966.

23. J. R. Hartley, "The adaptive control of learning tasks," Internal Paper Computer Based Learning Project, University of Leeds, 1972.

24. F. Black, "A deductive question-answering system." In M. Minsky (ed.) Semantic Information Processing, MIT press, 1968.

25. C. E. Hughes & M. S. Cresswell, "Introduction to modal logic." Methuen, London, 1972.

26. T. O'Shea & D. H. Sleeman, "A design for an adaptive self-improving teaching system." in Advances in Cybernetics, Gordon & Breach., 1973.

27. C. Weissman, "A LISP 1.5 primer," Publishers Dickenson, 1967.

28. R. W. Floyd, "Nondeterministic algorithms." JACM, Vol. 14, No. 4, 1967.

29. C. Hewitt, "Procedural embedding of knowledge in planner." In Proceedings of Second International Joint Conference on Artificial Intelligence, London, 1971.

30. J. McCarthy & P. Hayes, "Some philosophical problems from the standpoint of artificial intelligence." In B. Meltzer & D. Michie (eds.) Machine Intelligence 4, p. 463-502, 1969.

31. M. Greenawalt, "U.T. LISP manual." Report (in preparation), University of Texas at Austin, 1973.

APPENDIX I

Two protocols for the pretest are given. Student responses are indicated by "S:"

a) Student familiar with algebraic equations.

```
HULLO I AM GOING TO TEACH YOU SOME MATH
EVERY TIME I ASK YOU A QUESTION
AFTER YOU HAVE FINISHED TYPING YOUR ANSWER
PLEASE REMEMBER TO PRESS THE RETURN KEY
ON THE RIGHT SIDE OF THE CONSOLE

WHAT IS YOUR CHRISTIAN NAME?

S:  JOHN

DO YOU UNDERSTAND EQUATIONS OF THE FORM
X - B = C

S:  YES
FINE

SOLVE FOR X IN THE EQUATION
X - 4 = 7

S:  3
WRONG TRY AGAIN

S:  11
RIGHT

SOLVE FOR X IN THE EQUATION
X - 5 = 10

S:  15
RIGHT

SOLVE FOR X IN THE EQUATION
X - 7 = 14

S:  21
RIGHT

DO YOU UNDERSTAND EQUATIONS OF THE FORM
X + B = C

S:  YES
FINE

SOLVE FOR X IN THE EQUATION
X + 3 = 6

S:  3
RIGHT
```

```
SOLVE FOR X IN THE EQUATION
X + 5 = 10

S:  5
RIGHT
SOLVE FOR X IN THE EQUATION
X + 8 = 15

S:  7
RIGHT
DO YOU UNDERSTAND EQUATIONS OF THE FORM
X TIMES X = B

S:  YES
FINE
SOLVE FOR X IN THE EQUATION
X TIMES X = 64

S:  6
WRONG TRY AGAIN

S:  8
RIGHT
CONSIDER THE EQUATION
X TIMES X + 6 = X TIMES 5
******************************
FIND VALUES FOR X WHICH MAKE THIS EQUATION TRUE

S:  5
WRONG TRY AGAIN

S:  3
RIGHT
YES THATS RIGHT BECAUSE
3 TIMES 3 EQUALS 9
IF WE ADD 6 THAT EQUALS 15
WHICH IS EQUAL TO 3 TIMES 5
WHICH IS 15
THERE IS ANOTHER SOLUTION SEE IF YOU CAN GUESS IT

S:  9
WRONG TRY AGAIN

S:  1
NO SORRY STILL WRONG
NO THE OTHER ANSWER IS 2 BECAUSE
2 TIMES 2 EQUALS 4
IF WE ADD 6 THAT EQUALS 10
WHICH IS EQUAL TO 2 TIMES 5
WHICH IS 10
***
```

NOW LOOK OUT FOR RULES WHICH WILL HELP YOU SOLVE THE EQUATIONS IN GENERAL

***

b) Student unfamiliar with algebraic equations

DO YOU UNDERSTAND EQUATIONS OF THE FORM
X - B = C

S: YES
FINE

SOLVE FOR X IN THE EQUATION
X - 4 = 7

S: 3
WRONG TRY AGAIN

S: 2
NO SORRY STILL WRONG

LETS LOOK AT THE QUESTION ANOTHER WAY

SUPPOSE WE HAVE A BAG WITH SOME MARBLES IN
AND WE TAKE FROM THE BAG 4 MARBLES
AND THEN TIP OUT THE BAG AND FIND 7 MARBLES
HOW MANY MARBLES WERE IN THE BAG TO START WITH

S: 1
WRONG TRY AGAIN

S: 2
NO SORRY STILL WRONG

NO WHEN WE EMPTIED THE BAG WE FOUND 7 MARBLES
NOW FROM THE BAG WE HAD TAKEN 4 MARBLES
AND SO TO START WITH WE HAD 7 PLUS 4 MARBLES
MAKING THE RIGHT ANSWER 11 MARBLES

NOW SUPPOSE WE TAKE FROM THE BAG 2 MARBLES
AND THEN FIND 4
HOW MANY MARBLES WERE IN THE BAG

S: 6
RIGHT

NOW SUPPOSE WE TAKE FROM THE BAG 3 MARBLES
AND THEN FIND 7
HOW MANY MARBLES WERE IN THE BAG

S: 10
RIGHT

NOW SUPPOSE WE TAKE FROM THE BAG 6 MARBLES
AND THEN FIND 11
HOW MANY MARBLES WERE IN THE BAG

S: 17
RIGHT

NOW SUPPOSE WE TAKE FROM THE BAG 7 MARBLES
AND THEN FIND 14
HOW MANY MARBLES WERE IN THE BAG

S: 21
RIGHT

SUPPOSE WE HAVE A BAG WITH SOME MARBLES IN
AND WE ADD TO THE BAG 3 MARBLES
AND THEN TIP OUT THE BAG AND FIND 6 MARBLES
HOW MANY MARBLES WERE IN THE BAG TO START WITH

S: 3
RIGHT

NOW SUPPOSE WE ADD TO THE BAG 6 MARBLES
AND THEN FIND 11
HOW MANY MARBLES WERE IN THE BAG

S: 5
RIGHT

NOW SUPPOSE WE ADD TO THE BAG 8 MARBLES
AND THEN FIND 15
HOW MANY MARBLES WERE IN THE BAG

S: 7
RIGHT

SUPPOSE WE HAVE SOME BOYS IN A ROOM
AND EACH BOY HAS AS MANY MARBLES AS THERE ARE
BOYS IN THE ROOM AND THEY PUT ALL THEIR MARBLES TOGETHER
AND FIND THEY HAVE 64 MARBLES
HOW MANY BOYS WERE IN THE ROOM

S: 64
WRONG TRY AGAIN

S: 12
NO SORRY STILL WRONG

NO WE FOUND 64 MARBLES IN THE ROOM
SO AS THERE WERE AS MANY BOYS AS THERE WERE MARBLES
THERE MUST HAVE BEEN 8 BOYS WITH 8 MARBLES
MAKING ALTOGETHER 8 TIMES 8 MARBLES WHICH IS 64

NOW SUPPOSE WE FIND 36 MARBLES
HOW MANY BOYS WERE THERE

S: 6
RIGHT

NOW SUPPOSE WE FIND 49 MARBLES
HOW MANY BOYS WERE THERE

S: 7
RIGHT

SUPPOSE WE HAVE A GROUP OF BOYS WITH SOME MARBLES
EACH BOY HAVING AS MANY MARBLES AS THERE ARE BOYS IN THE GROUP
AND THEY ALL PUT ALL THEIR MARBLES IN A PILE
AND THEN SOMEBODY ADDS *** 6 *** MARBLES TO THE PILE
THEY THEN SHARE OUT THE MARBLES AND FIND THERE ARE EXACTLY *** 5 ***
MARBLES EACH WITH NONE LEFT OVER
CAN YOU GUESS HOW MANY BOYS WERE IN THE GROUP

S: 1
WRONG TRY AGAIN

S: 2
RIGHT

YES THATS RIGHT BECAUSE

2 TIMES 2 EQUALS 4
IF WE ADD 6 THAT EQUALS 10
WHICH IS EQUAL TO 2 TIMES 5
WHICH IS 10

THERE IS ANOTHER SOLUTION SEE IF YOU CAN GUESS IT

S: 3
RIGHT
FINE CHECK THAT IS RIGHT

FINE NOW WE WILL GO ON TO THE MAIN PROBLEM

NOW LOOK OUT FOR RULES WHICH WILL HELP YOU SOLVE THE EQUATIONS IN
GENERAL

APPENDIX II

Protocol of Student After Completing Post-Test

Note: The first problem is in the actual format used for all the problems. The subsequent problems have been edited in order to save space.

NOW LOOK OUT FOR RULES WHICH WILL HELP YOU SOLVE THE EQUATIONS IN GENERAL.

CONSIDER THE EQUATION X TIMES X + 4 = X TIMES 5
FIND VALUES FOR X WHICH MAKE THIS EQUATION TRUE
TRY AND GUESS THE ANSWERS

S: 4
THATS RIGHT SEE IF YOU CAN GET THE OTHER ANSWER

S: 3
NOPE

S: 2
NOPE

S: 4
YES BUT YOU HAVE GUESSED THAT CORRECTLY BEFORE

S: 5
NOPE

S: 3
NOPE

S: 6
NOPE

S: 7
NOPE

S: 8
NOPE

S: 9
NOPE

S: 10
NOPE

S: 3
NOPE

S: 0
NOPE

S: 11
NOPE

S: 12
NOPE
THAT WAS A LOT OF GUESSES
LETS TRY ANOTHER EXAMPLE

THE RIGHT ANSWERS ARE IN FACT 1 AND 4

CONSIDER THE EQUATION X TIMES X + 7 = X TIMES 8 GUESS THE ANSWERS

S: 7
THATS RIGHT SEE IF YOU CAN GET THE OTHER ANSWER

S: 1
WELL DONE YOU HAVE GOT BOTH THE ANSWERS

CONSIDER THE EQUATION X TIMES X + 2 = X TIMES 3 GUESS THE ANSWERS

S: 2
THATS RIGHT SEE IF YOU CAN GET THE OTHER ANSWER

S: 1
WELL DONE YOU HAVE GOT BOTH THE ANSWERS

CONSIDER THE EQUATION X TIMES X + 5 = X TIMES 6 GUESS THE ANSWERS

S: 1
THATS RIGHT SEE IF YOU CAN GET THE OTHER ANSWER

S: 5
WELL DONE YOU HAVE GOT BOTH THE ANSWERS

CONSIDER THE EQUATION X TIMES X + 6 = X TIMES 7 GUESS THE ANSWERS

S: 1
THATS RIGHT SEE IF YOU CAN GET THE OTHER ANSWER

S: 6
WELL DONE YOU HAVE GOT BOTH THE ANSWERS

```
CONSIDER THE EQUATION   X TIMES X + 8 = X TIMES 9   GUESS THE ANSWERS

S:  1
THATS RIGHT SEE IF YOU CAN GET THE OTHER ANSWER
S:  8
WELL DONE YOU HAVE GOT BOTH THE ANSWERS
CONSIDER THE EQUATION   X TIMES X + 18 = X TIMES 9   GUESS THE ANSWERS
S:  WHAT HAPPENS IF I CANT
PLEASE TYPE IN A NUMBER OR TYPE STOP IF YOU WISH TO STOP

S:  1, 2
NOPE
S:  3
THATS RIGHT SEE IF YOU CAN GET THE OTHER ANSWER
S:  4, 5
NOPE
S:  6
WELL DONE YOU HAVE GOT BOTH THE ANSWERS
CONSIDER THE EQUATION   X TIMES X + 18 = X TIMES 19   GUESS THE ANSWERS

S:  3
NOPE KEEP TRYING
S:  6, 12, 2, 3
NOPE KEEP TRYING
S:  4, 5, 6, 7, 8, 9, 10
YOU ARE NOT TRYING TO THINK OF ANY RULES TO HELP YOU
WE WILL LOOK AT ANOTHER EXAMPLE
TRY TO FIND A RULE
THE RIGHT ANSWERS ARE IN FACT 1 AND 18
CONSIDER THE EQUATION   X TIMES X + 48 = X TIMES 19   GUESS THE ANSWERS

S:  1, 18, 12, 48, 19, 64, 9, 2
NOPE
S:  3
THATS RIGHT SEE IF YOU CAN GET THE OTHER ANSWER
S:  6
NOPE KEEP TRYING
S:  3
YES BUT YOU HAVE GUESSED THAT CORRECTLY BEFORE
S:  15, 18, 9, 1
NOPE
THAT WAS A LOT OF GUESSES
LETS TRY ANOTHER EXAMPLE
THE RIGHT ANSWERS ARE IN FACT 3 AND 16
CONSIDER THE EQUATION   X TIMES X + 48 = X TIMES 49   GUESS THE ANSWERS

S:  1
THATS RIGHT SEE IF YOU CAN GET THE OTHER ANSWER
S:  48
WELL DONE YOU HAVE GOT BOTH THE ANSWERS
```

```
CONSIDER THE EQUATION   X TIMES X + 9 = X TIMES 10   GUESS THE ANSWERS

S:  1
THATS RIGHT SEE IF YOU CAN GET THE OTHER ANSWER
S:  9
WELL DONE YOU HAVE GOT BOTH THE ANSWERS

CONSIDER THE EQUATION   X TIMES X + 16 = X TIMES 10   GUESS THE ANSWERS

S:  6
NOPE KEEP TRYING
S:  2
THATS RIGHT SEE IF YOU CAN GET THE OTHER ANSWER
S:  20, 32, 4, 21, 12, 13, 9, 6, 14, 13, 19, 46, 13, 1000
THAT WAS A WILD GUESS
TRY AND BE MORE SENSIBLE SUNSHINE
THE RIGHT ANSWERS ARE IN FACT 2 AND 8

CONSIDER THE EQUATION   X TIMES X + 21 = X TIMES 10   GUESS THE ANSWERS

S:  2, 8, 1
NOPE KEEP TRYING
S:  3
THATS RIGHT SEE IF YOU CAN GET THE OTHER ANSWER
S:  9, 4, 5, 6
NOPE
S:  7
WELL DONE YOU HAVE GOT BOTH THE ANSWERS

CONSIDER THE EQUATION   X TIMES X + 24 = X TIMES 10   GUESS THE ANSWERS

S:  4
THATS RIGHT SEE IF YOU CAN GET THE OTHER ANSWER
S:  6
WELL DONE YOU HAVE GOT BOTH THE ANSWERS

CONSIDER THE EQUATION   X TIMES X + 12 = X TIMES 8   GUESS THE ANSWERS

S:  6
THATS RIGHT SEE IF YOU CAN GET THE OTHER ANSWER
S:  4, 12, 8, 9, 15, 12, 16, 13, 14, 16, 17, 18, 19, 20, 21
THAT WAS A LOT OF GUESSES
LETS TRY ANOTHER EXAMPLE
THE RIGHT ANSWERS ARE IN FACT 2 AND 6

CONSIDER THE EQUATION   X TIMES X + 15 = X TIMES 8   GUESS THE ANSWERS

S:  7, 0
NOPE KEEP TRYING
S:  3
THATS RIGHT SEE IF YOU CAN GET THE OTHER ANSWER
S:  5
WELL DONE YOU HAVE GOT BOTH THE ANSWERS

CONSIDER THE EQUATION   X TIMES X + 8 = X TIMES 6   GUESS THE ANSWERS
S:  24
NOPE
S:  2
THATS RIGHT SEE IF YOU CAN GET THE OTHER ANSWER
S:  4
WELL DONE YOU HAVE GOT BOTH THE ANSWERS
```

```
CONSIDER THE EQUATION   X TIMES X + 10 = X TIMES 7   GUESS THE ANSWERS

S:  2
THATS RIGHT SEE IF YOU CAN GET THE OTHER ANSWER
S:  5
WELL DONE YOU HAVE GOT BOTH THE ANSWERS
CONSIDER THE EQUATION   X TIMES X + 12 = X TIMES 7   GUESS THE ANSWERS

S:  3
THATS RIGHT SEE IF YOU CAN GET THE OTHER ANSWER
S:  4
WELL DONE YOU HAVE GOT BOTH THE ANSWERS
CONSIDER THE EQUATION   X TIMES X + 10 = X TIMES 11   GUESS THE ANSWERS

S:  1
THATS RIGHT SEE IF YOU CAN GET THE OTHER ANSWER
S:  10
WELL DONE YOU HAVE GOT BOTH THE ANSWERS
CONSIDER THE EQUATION   X TIMES X + 18 = X TIMES 11   GUESS THE ANSWERS

S:  2
THATS RIGHT SEE IF YOU CAN GET THE OTHER ANSWER
S:  9
WELL DONE YOU HAVE GOT BOTH THE ANSWERS
CONSIDER THE EQUATION   X TIMES X + 24 = X TIMES 11   GUESS THE ANSWERS

S:  2, 4, 1
NOPE
S:  3
THATS RIGHT SEE IF YOU CAN GET THE OTHER ANSWER
S:  6
NOPE KEEP TRYING
S:  8
WELL DONE YOU HAVE GOT BOTH THE ANSWERS
CONSIDER THE EQUATION   X TIMES X + 28 = X TIMES 11   GUESS THE ANSWERS
S:  7
THATS RIGHT SEE IF YOU CAN GET THE OTHER ANSWER
S:  4
WELL DONE YOU HAVE GOT BOTH THE ANSWERS
CONSIDER THE EQUATION   X TIMES X + 11 = X TIMES 12   GUESS THE ANSWERS

S:  1
THATS RIGHT SEE IF YOU CAN GET THE OTHER ANSWER
S:  11
WELL DONE YOU HAVE GOT BOTH THE ANSWERS
CONSIDER THE EQUATION   X TIMES X + 48 = X TIMES 16   GUESS THE ANSWERS

S:  4
THATS RIGHT SEE IF YOU CAN GET THE OTHER ANSWER
S:  12
WELL DONE YOU HAVE GOT BOTH THE ANSWERS
```

```
CONSIDER THE EQUATION   X TIMES X + 48 = X TIMES 26   GUESS THE ANSWERS

S:  6
NOPE KEEP TRYING
S:  2
THATS RIGHT SEE IF YOU CAN GET THE OTHER ANSWER
S:  24
WELL DONE YOU HAVE GOT BOTH THE ANSWERS

CONSIDER THE EQUATION   X TIMES X + 48 = X TIMES 14   GUESS THE ANSWERS
S:  3, 4
NOPE
S:  6
THATS RIGHT SEE IF YOU CAN GET THE OTHER ANSWER
S:  8
WELL DONE YOU HAVE GOT BOTH THE ANSWERS

CONSIDER THE EQUATION   X TIMES X + 300 = X TIMES 103   GUESS THE ANSWERS

S:  3
THATS RIGHT SEE IF YOU CAN GET THE OTHER ANSWER
S:  100
WELL DONE YOU HAVE GOT BOTH THE ANSWERS

WELL DONE I THINK YOU HAVE MASTERED THIS COMPLETELY
BYE BYE CHARLIE
```

APPENDIX III

The Initial Set of Production Rules

a) The elements of the statevector

i) CYCLE - the operation last performed by teaching program.

ii) TIME - the number of examples administered.

iii) NUMG - the number of guesses made by the student on the last example.

iv) EXDETAILS - a list of the features of the last example.

v) EXTYPE - a list of the parameters for GEN for the last example generated.

vi) TIMES - the hypothesized current ability of the student with the times rule.

vii) - x) ADD, ONE, WRULE, WORULE - as in vi) for the respective rules.

xi) TIMES - the time and direction of the last change in hypothesized ability with the TIMES rule.

xii) - xv) ADDCH, ONECH, WRULECH, WORULECH - as in xi) for the respective rules.

xvi) PRETEST - the student score on the pretest.

xvii) REASON - the reason for termination of administration of last example.

xviii) GOALTIME - the amount of time since the selection of the current goal.

xix) GOAL - the current goal.

b) The set of condition-action rules

```
( (2 B1) ( (DPRINER START1)(GOAL ΞONE)(GEN Ξ((ONE HELP)) NIL)
           (CONTINUE 1) ))
( (1 A3 17 I6) ( (DPRINER TERM)(STOP) ))
( (1 A3 2 B3) ( (DPRINER TIMETERM)(STOP) ))
( (1 A2 9 F1 18 J3 19 K8) ( (DPRINER SUCCTERM)(STOP) ))
( (1 A2 18 J3 19 K8) ( (GOAL ΞTIMES)(GEN Ξ((TIMES HELP)) NIL) ))
( (1 A2 19 K8) ( (CONTINUE 1) ) )
( (1 A3 18 J4) ( (HYPALL) ))
( (1 A3 18 J3) ( (HYPALL) ))
( (1 A3 6 F1 11 G3 19 K3) ( (HYPALL) ))
( (1 A3 7 F1 12 G3 19 K2) ( (HYPALL) ))
( (1 A3 8 F1 13 G3 19 K1) ( (HYPALL) ))
( (1 A3 2 B2) ( (HYPALL) ))
( (1 A2 2 B2 10 F8) ( (DPRINER TIMETERM)(STOP)))
( (1 A3 19 K1) ((HYP ΞONE) ))
( (1 A3 19 K2) ( (HYP ΞADD) ))
( (1 A3 19 K3) ( (HYP ΞTIMES) ))
T1: ( (1 A2 6 F1 7 F1 11 G3 12 G3) ( (GOAL ΞHARD)(GEN Ξ ((WRULE HIN)) NIL)
 ))
( (1 A2 10 F7 19 K8)( (GOAL ΞTIMES)(GEN Ξ((WRULE HELP)(TIMES HELP)) NIL)
))
T2: ( (1 A2 6 F1 11 G3 19 K3) ( (GOAL ΞTIMESTOADD)(GEN Ξ((TIMES HELP)
(ADD HELP)) NIL) ))
T3: ( (1 A2 6 F4 7 F7 18 J3 19 K3) ( (GOAL ΞADD)(GEN Ξ((ADD HELP)) NIL) ))
T4: ( (1 A2 6 F8 19 K3) ( (CONTINUE 1) ))
T5: ( (1 A2 7 (F8 F7) 19 K3) ( (GOAL ΞADD)(GEN Ξ((ADD HELP)) NIL) ))
T6: ( (1 A2 8 (F8 F7) 19 K3) ( (GOAL ΞONE)(GEN Ξ((ONE HELP)) NIL) ))
( (1 A2 7 F1 12 G3 19 K2) ( (GOAL ΞADDTOTIMES)(GEN Ξ((ADD HELP)
(TIMES HELP)) NIL) ))
( (1 A2 6 F7 7 F4 18 J4 19 K2)((GOAL ΞTIMES)(GEN Ξ((TIMES HELP)) NIL) ))
( (1 A2 7 F8 19 K2) ( (CONTINUE 1) ))
( (1 A2 6 (F8 F7) 19 K2) ( (GOAL ΞTIMES)(GEN Ξ((TIMES HELP)) NIL) ))
( (1 A2 8 (F8 F7) 19 K2) ( (GOAL ΞONE)(GEN Ξ((ONE HELP)) NIL) ))
( (1 A2 8 F1 13 G3 19 K1) ( (GOAL ΞONETOADD)(GEN Ξ((ADD HELP)
(ONE HIN)) NIL) ))
( (1 A2 7 F7 18 J4 19 K1) ( (GOAL ΞADD)(GEN Ξ((ADD HELP)) NIL) ))
( (1 A2 8 F8 19 K1) ( (CONTINUE 1) ))
( (1 A2 6 (F8 F7) 19 K1) ( (GOAL ΞTIMES)(GEN Ξ((TIMES HELP)) NIL) ))
( (1 A2 7 (F8 F7) 19 K1) ( (GOAL ΞADD)(GEN Ξ((ADD HELP)) NIL) ))
( (1 A2 7 F4 19 K6) ( (GOAL ΞADD)(GEN Ξ((ADD HELP)) NIL) ))
( (1 A2 18 J3 19 K6) ( (GOAL ΞTIMES)(GEN Ξ((TIMES HELP)(ADD HIN))
NIL) ))
( (1 A2 19 K6) ( (GEN Ξ((TIMES HELP)(ADD HIN)) NIL)(HYP ΞTIMES)
(GEN Ξ((ADD HELP) (TIMES HIN) ) NIL)(HYP ΞADD) ))
( (1 A2 6 F4 19 K7) ( (GOAL ΞTIMES)(GEN Ξ((TIMES HELP)) NIL) ))
( (1 A2 18 J3 19 K7) ( (GOAL ΞADD)(GEN Ξ((ADD HELP)(TIMES HIN))
NIL) ))
( (1 A2 19 K7)((GEN Ξ((ADD HELP)(TIMES HIN)) NIL)(HYP ΞADD)
(GEN Ξ((TIMES HELP)(ADD HIN)) NIL)(HYP ΞTIMES) ))
( (1 A2 8 F4 19 K4) ( (GOAL ΞONE)(GEN Ξ((ONE HELP)) NIL) ))
((1 A2 18 J3 19 K4) ((GOAL ΞADD)(GEN Ξ((ADD HELP)(ONE HIN)) NIL)
))
( (1 A2 19 K4) ( (GEN Ξ((ONE HELP)(ADD HIN)) NIL)(HYP ΞONE)
(GEN Ξ((ADD HELP)(ONE HIN)) NIL)(HYP ΞADD) ))
T7: ( (1 A2) ( (CONTINUE 1) ))
( (1 A3) ( (HYPALL) ))
( () ( (PRIN1 ΞFAIL) ))
```

c) The set of partitions

```
(PUT ΞQUADPARTS ΞCYCLE Ξ(
      (A1 EQUAL ELEMENT ΞGEN)
      (A2 EQUAL ELEMENT ΞHYP)
      (A3 EQUAL ELEMENT ΞADMIN) ))
(PUT ΞQUADPARTS ΞTIME Ξ(
      (B1 EQUAL ELEMENT 0)
      (B2 EQUAL ELEMENT 10)
      (B3 GREATERP ELEMENT 30) ))
(PUT ΞQUADPARTS ΞNUMG Ξ(
      (C1 EQUAL ELEMENT 2)
      (C2 BETWEEN ELEMENT 11 1)
      (C3 GREATERP ELEMENT 10) ))
(PUT ΞQUADPARTS ΞEXDETAILS Ξ(
      (D1 MEMBER ΞNEW ELEMENT)
      (D2 MEMBER ΞOLD ELEMENT) ))
(PUT ΞQUADPARTS ΞEXTYPE Ξ(
   (E1 MEMBER  Ξ(TIMES HELP) ELEMENT)
   (E2 MEMBER Ξ(ADD HELP) ELEMENT)
   (E3 MEMBER Ξ(ONE HELP) ELEMENT)
   (E4 MEMBER Ξ(WRULE HIN) ELEMENT) ))
(PUT ΞQUADPARTS ΞTIMES Ξ(
      (F1 EQUAL ELEMENT ΞCERT)
      (F2 MEMBER ELEMENT Ξ(CERT VPOSS POSS))
      (F3 EQUAL ELEMENT ΞDUNNO)
      (F4 MEMBER ELEMENT Ξ(POSSNOT CERTNOT))
      (F5 EQUAL ELEMENT ΞCERTNOT)
      (F6 EQUAL ELEMENT ΞPOSS)
      (F7 NEQ ELEMENT ΞCERT)
      (F8 MEMBER ELEMENT Ξ(CERT VPOSS))
      (F9  EQUAL ELEMENT ΞNOCHECK) ))
(PUT ΞQUADPARTS ΞTIMESCH Ξ(
      (G1 EQUAL ELEMENT 0)
      (G2 EQUAL ELEMENT 1)
      (G3 GREATERP ELEMENT 2)
      (G4 EQUAL ELEMENT -1)
      (G5 LESSP ELEMENT 2) ))
(PUT ΞQUADPARTS ΞPRETEST NIL)
(PUT ΞQUADPARTS ΞREASON Ξ(
      (I1 EQUAL ELEMENT ΞCORRECT)
      (I2 EQUAL ELEMENT ΞGUESSLIM)
      (I3 EQUAL ELEMENT ΞSYSLIM)
      (I4 EQUAL ELEMENT ΞWILDLIM)
      (I5 EQUAL ELEMENT ΞREPLIM)
      (I6 EQUAL ELEMENT ΞSTOP) ))
(PUT ΞQUADPARTS ΞGOALTIME Ξ(
      (J1 EQUAL ELEMENT 1)
      (J2 BETWEEN ELEMENT 4 0)
      (J3 EQUAL ELEMENT 4)
      (J3 EQUAL ELEMENT 5)
      (J4 GREATERP ELEMENT 7) ))
(PUT ΞQUADPARTS ΞGOAL Ξ(
      (K1 EQUAL ELEMENT ΞONE)
      (K2 EQUAL ELEMENT ΞADD)
      (K3 EQUAL ELEMENT ΞTIMES)
      (K4 EQUAL ELEMENT ΞONETOADD)
      (K5 EQUAL ELEMENT ΞONETOTIMES)
      (K6 EQUAL ELEMENT ΞADDTOTIMES)
      (K7 EQUAL ELEMENT ΞTIMESTOADD)
      (K8 EQUAL ELEMENT ΞHARD)
      (K9 EQUAL ELEMENT ΞEASY) ))
```

Note: The initial partitions for ADD, ONE, WRULE, WORULE are set the same as that for TIMES. Likewise the partitions for ADDCH, ONECH, WRULECH, WORULECH are the same as that for TIMESCH.

d) Example of cycle of operation for set of production rules.

The statevector: (≡HYP, 17, 8, ≡(NEW, EQ1), ≡((ONE, HELP)), CERT, CERT, VPOSS, VPOSS, CERTNOT, 4, 3, 1, 2, 7, 10, ≡CORRECT, 5, ≡ADD) when parsed with the set of partition gives the parsed vector: (A2, NIL, C2, D1, E3, (F1, F2, F8), (F1, F2, F8), (F2, F7, F8), (F2, F7, F8), (F6, F5), G3, G3, (G2, G5), NIL, G3, NIL, I1, J3, K2). This matches the rule labelled T1 and the right hand side is executed.

APPENDIX IV

The Set of Rules for Evaluating the Hypothesis that the Student has the TIMES Rule

a) Statevector: (PAIRMULT, NUMFACT, NUMGUESSES, GAP, LASTC, HASTIMES)

b) Condition-action rules:

```
( (OR(NIL NIL 1 NIL NIL 1)
      (NIL 2 NIL NIL NIL 1)
     (3 NIL NIL 1 NIL 6)
     (NIL 3 NIL 1 NIL 6)
     (2 2 2 1 3 6)) ΞCERT)
( (OR (3 NIL 4 2 3 2)
      (NIL 3 4 2 3 2)
      (NIL NIL NIL 1 NIL 6)
      (2 2 4 2 3 2)) ΞVPOSS)
( (OR (2 NIL NIL 2 NIL NIL)
      (NIL NIL NIL NIL NIL 6)
      (NIL 2 NIL 2 NIL NIL)) ΞPOSS)
( (OR (NIL NIL NIL 4 NIL NIL)
      (1 NIL NIL NIL NIL NIL)
      (NIL 1 NIL NIL NIL NIL)
      (NIL NIL 3 1 NIL NIL)) ΞCERTNOT)
( (OR (NIL NIL NIL 3 NIL NIL)
      (NIL NIL NIL NIL  NIL 4)
      (NIL NIL 3 3 NIL NIL)) ΞPOSSNOT)
( (NIL NIL NIL NIL NIL NIL) ΞDUNNO) ))
```

c) Partitions:

```
(PUT ΞHYPPARTS ΞPAIRMULT Ξ(
      (1 LESSP ELEMENT 50)
      (2 GREATERP ELEMENT 50)
      (3 EQUAL ELEMENT 100) ))
(PUT ΞHYPPARTS ΞNUMFACT Ξ(
      (1 LESSP ELEMENT 50)
      (2 GREATERP ELEMENT 50)
      (3 EQUAL ELEMENT 100) ))
(PUT ΞHYPPARTS ΞNUMGUESSES Ξ(
      (1 EQUAL ELEMENT 2)
      (2 GREATERP ELEMENT 2)
      (3 GREATERP ELEMENT 10)
      (4 LESSP ELEMENT 6) ))
(PUT ΞHYPPARTS ΞGAP Ξ(
      (1 EQUAL ELEMENT 0)
      (2 BETWEEN ELEMENT 5 -1)
      (4 GREATERP ELEMENT 1000)
      (3 GREATERP ELEMENT 4) ))
(PUT ΞHYPPARTS ΞLASTC Ξ(
      (1 LESSP ELEMENT 5)
      (2 GREATERP ELEMENT 5)
      (3 GREATERP ELEMENT 10)
      (4 GREATERP ELEMENT 20)
      (5 LESSP ELEMENT 10) ))
(PUT ΞHYPPARTS ΞHASTIMES Ξ(
      (1 EQUAL ELEMENT ΞCERT)
      (2 MEMBER ELEMENT Ξ(CERT VPOSS POSS))
      (3 EQUAL ELEMENT ΞDUNNO)
      (4 MEMBER ELEMENT Ξ(POSSNOT CERTNOT))
      (5 EQUAL ELEMENT ΞCERTNOT)
      (6 MEMBER ELEMENT Ξ(VPOSS CERT)) ))
```

APPENDIX V

Task Difficulty Matrix for $x^2 + c = b \cdot x$ with Solutions x1, x2

	NO RULE	ALL RULES	ONE	TIMES	ADD
$b < 15$	0	2	0	0	4
$1 \leq c < 8$	5	3	3	3	3
$8 \leq c \ 13$	3	2	2	2	2
$13 \leq c < 31$	-3	1	1	1	0
$31 \leq c < 51$	-5	0	-1	0	0
b PRIME	0	0	0	0	2
x1 CONSTANT	0	2	0	3	3
x1 = 1	0	0	5	0	0
b CONSTANT	0	0	0	-2	5
c CONSTANT	0	3	0	4	-3
c 2 FACTORS	0	5	0	0	0
c 4 FACTORS	0	2	0	4	0
c 6 FACTORS	-1	-1	0	-1	0
c 8 FACTORS	-2	-2	0	-3	0
c 10 FACTORS	-3	-3	0	-4	0
x1 = 100	-1	1	0	0	1
x1, x2 $\neq$ 1	0	0	-5	0	0

Note: b CONSTANT means b to be set equal to the value of b in the previous example

APPENDIX VI

Set of Assertions Used by the Deduction System

a) Causal assertions

```
(POSS (DECREASE (HYPOTHESIS TESTING)) (OR(DECREASE (COMPUTER TIME))
      (INCREASE SCORES) (INCREASE (STUDENT TIME))) )
(CERT (REMOVE HYPS) (DECREASE (HYPOTHESIS TESTING)) )
(CERT (AND(EQUAL (VAR CYCLE) ADMIN)(RAISE CONTINUE))
   (DECREASE (HYPOTHESIS TESTING)) )
(CERT (AND(EQUAL SUBRULES CERT)(REPLACE (HYPALL) ((HYP ΞONE)
   (HYP ΞADD)(HYP ΞTIMES)))) (DECREASE (HYPOTHESIS TESTING)) )
(POSS (INCREASE (HYPOTHESIS TESTING)) (OR(INCREASE (COMPUTER TIME))
   (INCREASE SCORES) (DECREASE (STUDENT TIME)) ))
(CERT (REPLACE HYPS (HYPALL)) (INCREASE (HYPOTHESIS TESTING)) )
(POSS (REPLACE MAIN MAIN) (CHANGE SOMEGOAL) )
(POSS (REMOVE MINOR) (DECREASE (STUDENT TIME)) )
(POSS (REMOVE (GOAL ΞEASY)) (DECREASE (COMPUTER TIME)) )
(POSS (REMOVE (GOAL ΞHARD)) (DECREASE (POST SCORE)) )
(POSS (AND(BEFORE (GOAL ΞTIMES))(AFTER (GOAL ΞONE))(INSERT
   (GOAL ΞADD))) (CHANGE SCORES) )
(POSS (AND (REMOVE (GOAL ΞONE))(REMOVE (GOAL ΞONETOADD))
   (REMOVE (GOAL ΞONETOTIMES))) (OR(DECREASE SCORES)(DECREASE TIMES)) )
(POSS (LOWER EXLIMIT) (DECREASE (STUDENT TIME)) )
(POSS (AND(AFTER (GOAL ΞEASY))(REPLACE (GOAL ΞONE) (GOAL ΞTIMES)))
   (SHORTEN SESSIONS) )
(POSS (AND(LOWER PROGLIMIT)(PRESENT (STOP))) (SHORTEN SESSIONS) )
(CERT (SHORTEN SESSIONS) (DECREASE TIMES) )
(POSS (SHORTEN SESSIONS) (DECREASE SCORES) )
(CERT (KEEP BORED STUDENTS) (INCREASE TIMES) )
(POSS (KEEP BORED STUDENTS) (OR(DECREASE(POST SCORE))(INCREASE
   (STUDENT SCORE))) )
(CERT (AND(REPLACE (DPRINER ΞTERM)(DPRINER ΞCONTINUE))
   (REPLACE (STOP) (CONTINUE 1))) (KEEP BORED STUDENTS) )
(POSS (LOWER HYPTIME) (OR (DECREASE TIMES)(DECREASE (POST
     SCORE))))
(POSS (LOWER (VAR GOALTIME)) (OR (DECREASE TIMES)(DECREASE (POST
      SCORE))))
(POSS (LOWER GUESSLIM) (DECREASE (STUDENT TIME)) )
(POSS (LOWER EXLIMIT) (DECREASE SCORES) )
(POSS (AND (LOWER GUESSLIM) (INSERT (ADMIN RIGHTANS)))
   (CHANGE SCORES) )
(POSS (RAISE ENCRATE) (DECREASE (POST SCORE)) )
(POSS (DECREASE WILDLIM)(OR(DECREASE SCORES)(DECREASE (STUDENT TIME)))
   )
(POSS (vAISE SYSLIM) (INCvEASE (STUDENT SCOvE)) )
```

b) Definitional Assertions

```
(EQUIV AFTER (NOT BEFORE))
(EQUIV ABSENT (NOT PRESENT))
(EQUIV LOWER (NOT RAISE))
(EQUIV INCREASE (NOT DECREASE))
(EQUIV CHANGE (OR INCREASE DECREASE))
(EQUIV (AND TIMES SCORES) ALLGOALS)
(EQUIV (AND (STUDENT TIME)(COMPUTER TIME)) TIMES)
(EQUIV (AND (STUDENT SCORE)(POST SCORE)) SCORES)
(EQUIV (OR (COMPUTER TIME)(STUDENT TIME)(STUDENT SCORE)
   (POST SCORE) SCORES TIMES ALLGOALS) SOMEGOAL)
(EQUIV (OR REPLIM SYSLIM GUESSLIM WILDLIM) EXLIMIT)
(EQUIV (OR (VAR TIME )(VAR PRETEST)) PROGLIMIT)
(EQUIV (OR (GOAL ΞONETOADD)(GOAL ΞADDTOTIMES)
   (GOAL ΞTIMESTOADD)) SWITCH)
(EQUIV (OR (GOAL ΞADD)(GOAL ΞTIMES)(GOAL ΞONE)) MAIN)
(EQUIV (OR (GOAL ΞHARD)(GOAL ΞEASY)) MINOR)
(EQUIV (OR MAIN MINOR SWITCH) PROGGOALS)
(EQUIV (AND(HYP ΞONE)(HYP ΞADD)(HYP ΞTIMES)(HYP ΞWRULE)
   (HYP ΞWORULE)(HYPALL) ) HYPS)
(EQUIV (OR (VAR ADD)(VAR ONE)(VAR TIMES)) SUBRULES)
(EQUIV (OR (VAR TIMESCH)(VAR ADDCH)(VAR ONECH)(VAR WRULECH)
    (VAR WORULECH)) HYPTIME)
```

APPENDIX VII

Student Comments

The written post-test included a space for comments on the program. A selection of these is given below:

'I caught onto the formula. It's easy and fun.'

'This is a really interesting program. I enjoyed working at the computer a lot. I learned the pattern but it took me a long time.'

'I began to feel like a machine too. If I had to learn like this all the time I would get bored. I prefer personal contact. It's probably all right for science students but arts students need more stimulation.'

'I enjoyed fooling with it. I wish I had one in my bedroom.'

'It would be better if it could talk to you.'

'It's a far-out machine.'

'I think learning by this method is a good brain exercise.'

'I enjoyed working on your program.'

'This was a very good teaching method you have thought up.'

'It was fun and I love the way you talk!'

'Why? What was the meaning of this?'

'Its really a great program, in time am sure it will help many people who are now having trouble with math! But it will take some time!'

'It makes you think of your own shortcuts to the problems.'

'At first I didn't realize that x times x was really x^2, or that in fact I was working a quadradic equasion. It really is a good system because I have never really understood the quadradic equasion or how to work it.'

'I think it was a fun and interesting way to learn.'

'This program is very effective and I hope when I have kids they will be able to learn this way. I like it very much

-it is a lot easier learning this way rather than a teacher. The learning lasts a whole lot longer. Also, computers are not prejudice!'

'The computer is real cool, I think it should be used in school.'