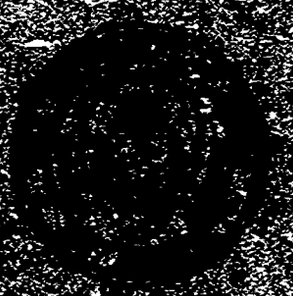
DOCUMENT RESUME

ED 072 633 EM 010 721

AUTHOR	Fine, Stephen Ronald
TITLE	Learner Control Commands for Computer-Assisted Instruction Systems. Technical Report 15.
INSTITUTION	Texas Univ., Austin. Computer-Assisted Instruction Lab.
SPONS AGENCY	National Science Foundation, Washington, D.C.
REPORT NO	TR-15
PUB DATE	May 72
NOTE	68p.; Thesis submitted to the University of Texas
EDRS PRICE	MF-$0.65 HC-$3.29
DESCRIPTORS	Autoinstructional Aids; Cognitive Processes; *Computer Assisted Instruction; Feedback; Independent Study; Individual Differences; *Individualized Instruction; *Instructional Design; Instructional Systems; *Programing; *Task Analysis
IDENTIFIERS	*Learner Control

ABSTRACT

Student control of computer-assisted instruction has a number of pragmatic advantages due to difficulties in providing a general program which is truly individualized. The attempt here has been to define and describe the commands which could be modified to provide learner control in any computer-assisted instruction course. Some of the commands are course-independent, and others will be dependent on the structure and content of the particular course involved. (RH)



LEARNER CONTROL COMMANDS FOR COMPUTER-ASSISTED INSTRUCTION SYSTEMS

TECHNICAL REPORT NO. 15

by

Stephen Ronald Fine

May 1972

Supported By:

THE NATIONAL SCIENCE FOUNDATION
Grant GJ 509 X

and A Contract from the MITRE Corporation
C. Victor Bunderson, Principal Investigator

The University of Texas at Austin
Computer-Assisted Instruction Laboratory
Austin, Texas 78712

Foreword

A major guiding principle of our NSF-supported research, "Foundations of Instructional Design for Computer-Based Systems", is the idea that through the cross-fertilization between computer-science concepts and instructional psychology concepts, decisive advances, leading toward a Design Science of Instruction, can occur.

One of the principal themes of this research has been the investigation of learner control and its relationship to task and program structure. Our IBM 1500 system-based research (referenced in Chapter II) on the effects of learner control vs. program control yielded interesting, if puzzling results. One of its primary benefits was to enable us to resort the entire question of the psychological purpose of learner control. Learner control, in this new view*, is important for the development of improved learning strategies, improved attitude (measured by voluntary approach to the subject matter), and an improved sense of responsibility on the part of the learner (measured as defined patterns of usage of learner control options). Mastery should not suffer in process of achieving these other objectives, but efficiency, measured as a ratio of mastery score units per unit time, can be weighted heavily only at the expense of the other variables.

This multivariate approach to educational objectives provides new leverage on the cost-effectiveness problem in assessing an instructional system. Not just mastery, but improved strategies, approach, and a more responsible attitude must be factored in on the effectiveness side. Efficiency is one of several cost factors. Using an implementation of learner

* C. Victor Bunderson, "Mainline CAI, Necessary But Not Oppressive", NSF Grant GJ 509 X; or, paper presented at the 1972 Spring Joint Computer Conference, Atlantic City, New Jersey, May 16-18, 1972.

control similar to the command language in this report, the TICCIT system (MITRE Corporation's Time-Shared Interactive, Computer Controlled Information Television) is the first to permit this multivariate assessment of cost-effectiveness.

For a semester preceding the initiation of the TICCIT project, and throughout the development of the TICCIT project to date, Steve Fine has been an active participant and contributor. He has faithfully drawn the systems implications for implementation of a variety of ideas emanating from the principles of Instructional Psychology. He has introduced a variety of novel learner control commands of his own invention. He has synthesized a conceptual framework showing the place of learner control in the interactions between the computer and the learners in the instructional situation. In so doing he has established an important landmark in the development of a rationale and an instrumentation for learner control as a central element in an emerging Science of Instruction.

C. Victor Bunderson

Austin, Texas
June, 1972

Acknowledgements

Primary thanks are due to C. Victor Bunderson whose help and suggestions have guided me at every stage in the development of this thesis. A number of the ideas and Learner Control commands described here originated with Dr. Bunderson and other members of the TICCIT project staff, and their work has resulted in modification and refinements to virtually every aspect of this work.

Thanks are also due to Laurent Siklossy and Robert F. Simmons for their many suggestions and comments during the preparation of this thesis, to Karl L. Zinn of the University of Michigan whose seminar stimulated many of the ideas included here, and to Leonard Uhr of the University of Wisconsin who first introduced me to CAI--an introduction which still guides my work.

Table of Contents

List of Figures

I. Introduction

The use of computers for instruction has three primary justifications. One is economic--it is claimed that Computer-Assisted Instruction (CAI) will soon be less expensive than current classroom methods (Bunderson, 1970b). A second justification is that the computer can provide tools not otherwise available. An example of this is the simulation of laboratory experiments too expensive or dangerous for the classroom. The third justification is the ability of CAI to adapt instruction to the individual needs of each student. It is this third goal which will be of interest here.

Much has been written about the advantages of individualized instruction. It is said to result in greater learning, more rapid learning, and better attitudes toward study. For the purposes of this discussion, the validity of this proposition will simply be assumed. The concern here will be with certain of the consequences of individualization, rather than the rationale behind it.

The primary concern of this paper will be techniques designed to give the student a measure of control over the instruction he receives. The author is prepared to defend this principle of Learner Control on philosophical grounds; however, a defense on pragmatic grounds seems more appropriate here.

One justification for Learner Control arises from deficiencies in current knowledge about individualization and from limitations in available CAI techniques. When there is no algorithm available to select

the proper instructional tool for a given student, the choice can often be left to the student himself. Examples of this situation will be described later.

A second justification for Learner Control is based on psychological theories of learning. Two effects of Learner Control are postulated. First, it forces the student to organize and direct his own learning. In this way, the student learns not only the particular subject matter of the course, but also study skills which can be applied to other areas. Second, Learner Control is believed to result in an improved attitude toward the course on the part of the student. This in turn should result in more learning and/or better retention (based in part on the theory that a student will work harder if he enjoys his work). Research into these effects will be discussed briefly later; unfortunately, the results of most of the research to date have been inconclusive.

In a sense, Learner-Controlled instruction can be thought of as a particular type of individualized instruction. Individualization places two main demands upon a CAI course. First, the course must be easily modifiable to fit the requirements of each individual. Second, there must be a means of determining those individual requirements. Learner Control is one such means.

It is not yet known with any degree of certainty what sorts of variation in instruction are necessary for individualization. Until definite conclusions can be reached, it seems reasonable to design CAI systems so as to maximize the possible adaptability of any course. Thus, throughout this paper, emphasis will be placed on expanding to the fullest

the possible ways in which the computer can modify the course material.

A wide variety of programs and purposes are subsumed under the general heading of Computer-Assisted Instruction. The discussion here will be centered on what has been termed "mainline" CAI (Bunderson; 1970b). "Mainline" CAI refers to the automation of all or most of a given course. It is contrasted with "adjunct" CAI, in which the computer is used for supplementary material: single lessons or groups of lessons taken by a student for purposes of remediation or as enrichment material. As is often true of computer applications, many of the problems discussed here only become significant when a large-scale application is involved. While only mainline applications will be discussed directly, much of what is said is also applicable to adjunct programs.

II. Overview of CAI and Learner Control Techniques

A. Types of CAI

Any discussion of Learner Control techniques is made more difficult by the wide variations possible in CAI program structure. Most Learner Control techniques (see Section III) must themselves vary to fit the requirements imposed by different CAI structures. Three dimensions will be defined to classify the variations in structure. The first dimension is "control" which details the relative division of control over instruction between teacher and student. The second dimension is "individualization" which describes the ability of a course to adapt in order to meet the needs of different students. The third dimension is "frame specification" which marks the degree to which the instructional materials are pre-defined by the author or are generated on-line. These three continua are diagrammed in Figure 1.

Perhaps the simplest model for CAI is the linear program. In this mode, all students see exactly the same material, unaffected by either entering behavior or actual responses. Individualization is negligible; the only variability arises from the fact that each student works at his own pace. This type of CAI makes no significant use of the computer, but has certain theoretical advantages, according to a theory developed by B. F. Skinner(1964).

A second type of CAI is the simple branching program. A common form of this program is the "intrinsic programming" model developed by

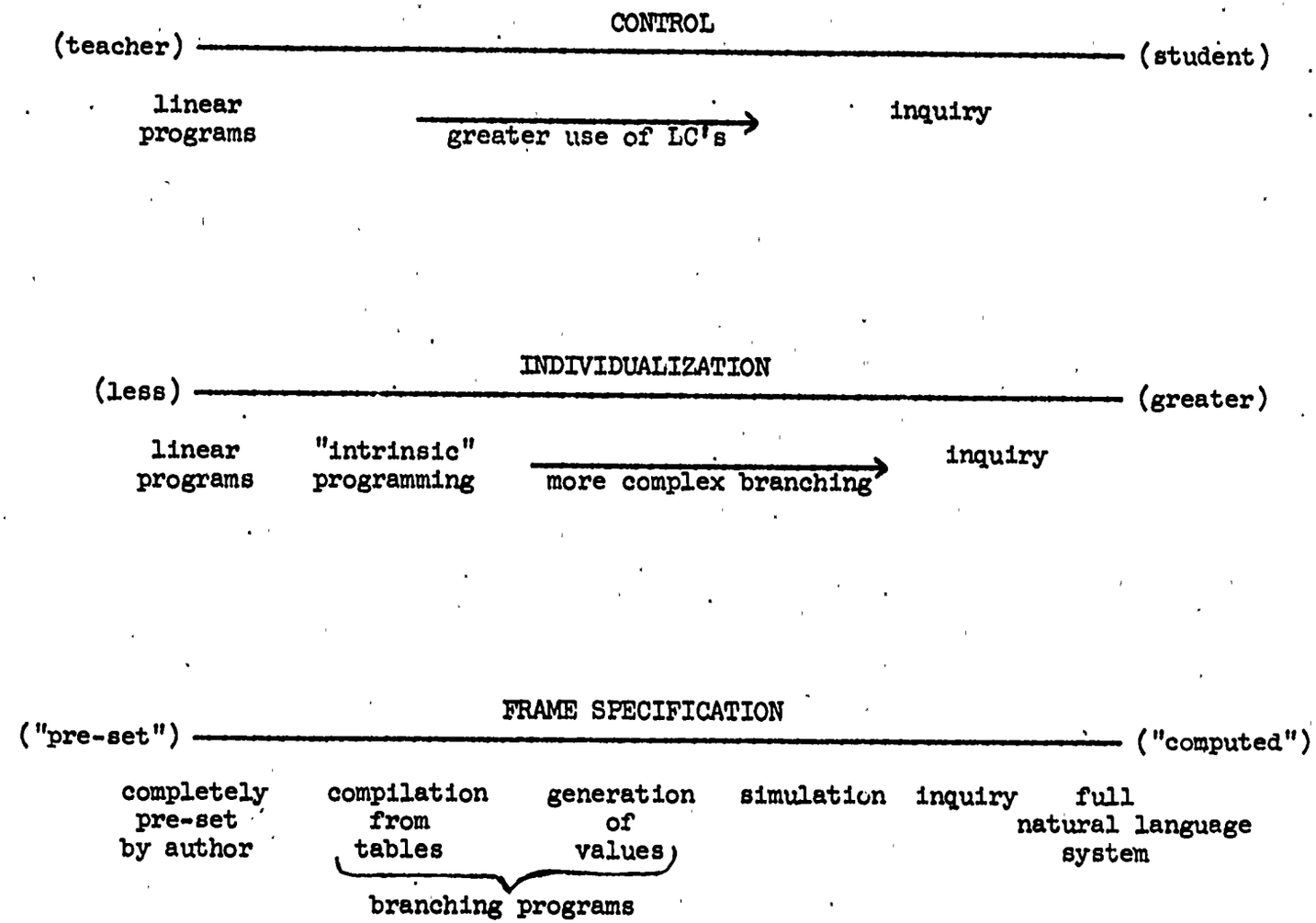


Figure 1: Dimensions of variation among CAI programs

Crowder (1964). In this model the questions are multiple-choice and different responses result in different instructional sequences. In the simplest case, often used in "scrambled book" Programmed Instruction texts, the student who answers incorrectly sees a remedial message and then is returned to the main sequence of instruction. More complex schemes can shunt the student to lengthy remedial sequences, or to enrichment material.

Branching programs can be even more complex. The branching can be based not only on the current response, but on the general response history, or even on an analysis of the student's learning patterns and style. However, such procedures remain primarily theoretical, since current programs rarely use more than the immediate response for branching. The reason for this lies in the difficulty of preparing programs with complex decision structures and varied branching sequences. Some ways of dealing with this problem will be discussed later.

At the highest level of complexity are programs for which there really is no explicit pre-defined path through the material; the path is a function of the interaction between the computer and the student. Inquiry programs and simulation programs generally fall into this class. The inquiry program is based on a large computerized data base. The learning is essentially an information-retrieval process in which the student conducts his own investigation of the subject matter with the assistance of the computer.

The computer can take different roles in this process. In some cases it is primarily a retrieval tool, not dependent in any major way on

the nature of the subject matter. Such a program might also be able to generate questions and answers as well as manipulate the data base in other ways. However, the program "knows" very little about the subject matter; it "knows" only how to manipulate the information in whatever data base it is given. The Wexler(1970) and Carbonell(1970) systems discussed in a later section are examples of this approach.

In other cases the computer can perform the operations being taught and thus generate both questions and answers. This is the type of system referred to by Siklossy as a "knowledgeable computer tutor" (1970, 1971). Such programs are currently possible only in a few areas, primarily in well-defined algorithmic subjects such as mathematics.

A truly knowledgeable computer tutor must go even a step further: Not only must it "know" the subject matter, it must also "know" English. Such a program could handle students with the same flexibility that is possible for a human tutor. However, this requires a natural language processing capability that is well beyond anything available currently.

The simulation program is much like the knowledgeable tutor described above, except that it cannot perform any actions, but merely model their performance. Usually, the program provides a model of some real-world situation which the student can manipulate. Most simulation programs are used for adjunct CAI and thus fall outside the scope of this paper, although one or more simulations may be programmed as a part of a larger CAI course. An interesting exception is a recent experimental program by Brown, Burton and Zdybel(1972) which combines a simulation model, natural

language processing, and some pre-stored textual material to build a small course in meteorology.

B. Learner Control

1. Interactive Control in CAI

CAI is an interactive medium. Siklossy has developed a model showing the basic channels along which this interaction can proceed(1971). This model is diagrammed in Figure 2. For purposes of control, the significant paths in the model are the T-S path and the S-T path. Information flow along the S-T path is often quite limited. In the linear type of program, the only real information used is the fact that a response has been made, which is a signal to continue. In most branching programs, the information flow consists of student responses to questions, in whatever limited form the program is designed to accept those responses. In more complex systems, the S-T path may be used for information and instructions about the teaching process, as well as responses to specific questions.

A more complex model is required to make clear the different types of information which can be transmitted along the S-T and T-S paths. One such model has been developed by Pask(1967). He asserts that all communication between student and computer takes place in one or more special languages. The specific flow of instructional information and student answers to questions and problems comprises what Pask calls the L^0 language. Discussion <u>about</u> the instructional process itself, and attempts

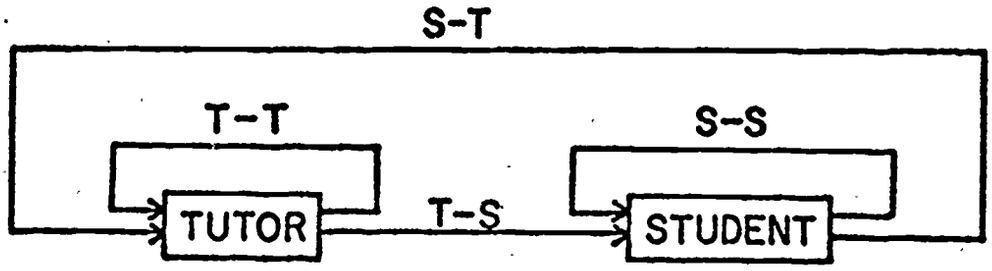


Figure 2: Transfer of Information in a Tutorial Environment(Siklossy, 1971)

by the student to control that process in some way, take place in L^1. It is possible also to define an L^2 language in which the control process can be discussed and modified. This definition process can be extended indefinitely.

For the purposes of this paper, Pask's model will be modified slightly. Rather than languages, the L^0, L^1, and L^2 will be thought of as progressively higher levels of discourse. Their use remains basically the same. An L^3 level will be defined as that level in which communication between humans takes place. Furthermore, the discussions taking place at the various levels of discourse involve more than just a 'student' and a 'tutor'. Additional participants in the CAI process must also be considered. A complete CAI system might involve as many as five components. These five participants are:

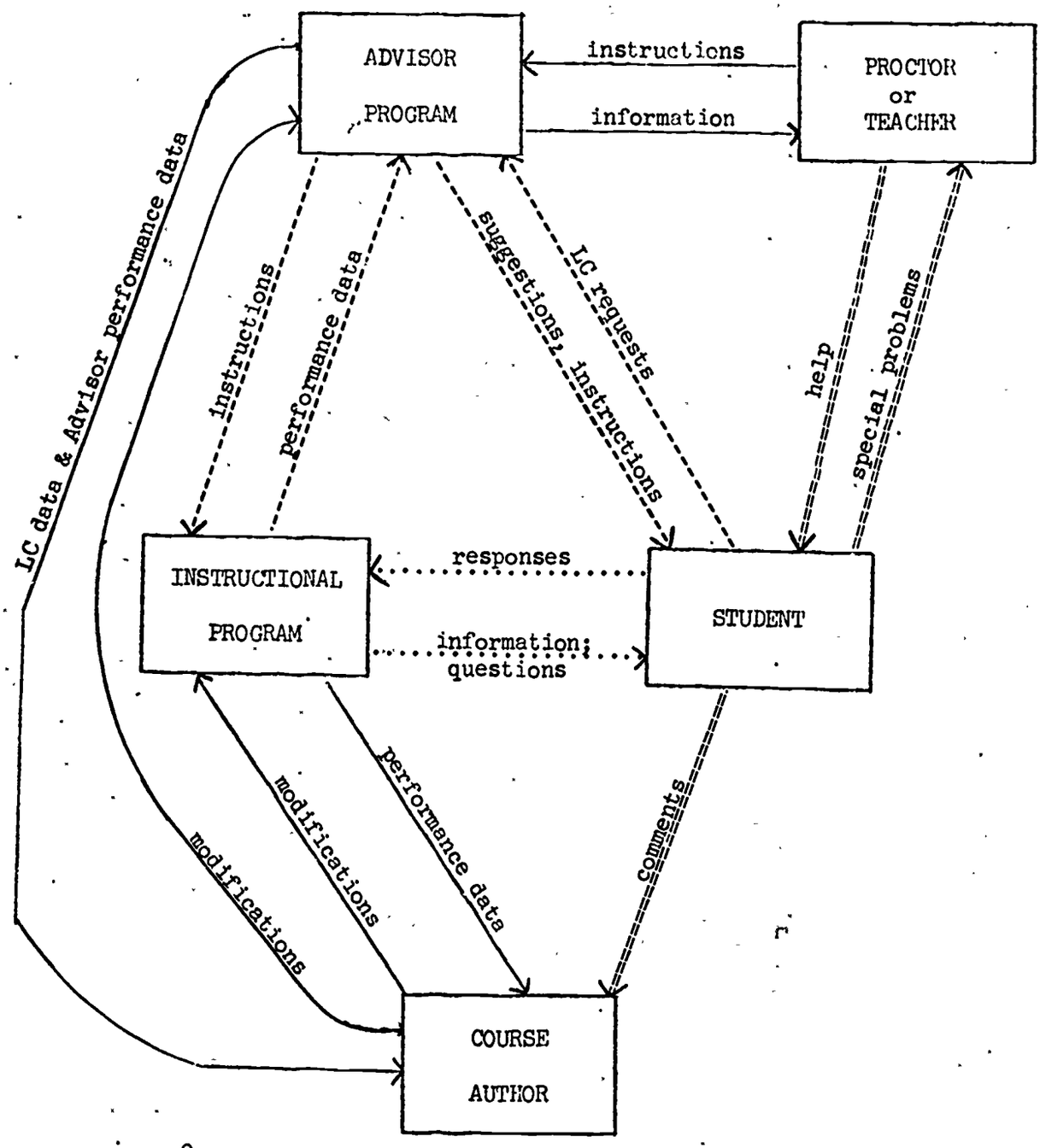
1) the student

2) the instructional program

This component is concerned only with the actual subject matter being taught; it interacts with the student only in L^0. It presents information and questions to the student, and processes student responses.

3) the Advisor program

This component is similar to Stolurow's "Professor" function(1969). It monitors the interaction and can both modify the sequence of instruction, and make comments and suggestions to the student. Furthermore, it is the major component in the Learner Control process, accepting and acting upon Learner Control commands from the student in L^1. More will be said about the role of the Advisor in this process later.



····· L^0: The basic instructional process
----- L^1: Control instructions and data
——— L^2: Material relating to the overall instructional process
===== L^3: Human, natural-language communications

Figure 3: Model of a Complete CAI System

4) the course author(s)

This might be a single person, or a group. Bunderson(1970a) proposes that course materials be developed by a team including subject matter specialists (such as teachers), instructional designers and psychologists (who are concerned with the way the subject matter is organized and presented), and computer programmers and coders (who must translate the work of the other members of the team into executable programs). Thus, in many places in this paper where reference is made to the CAI "author", this is really a shorthand for "the author or authoring team."

5) the proctor or teacher

This refers to a person who is available whenever students are on-line. He can interact both with the student and the system while the instruction is taking place.

Figure 3 diagrams the various interactions among these participants and the levels at which they occur. This model represents a "complete" CAI system. In many existing systems some of the participants may be missing and some of the paths atrophied. Such limited systems must (the author believes) have reduced flexibility and ability to adapt to the needs of each student. The Learner Controls described in this paper are discussed with the complete model in mind, although most do not require it.

As shown in the model, then, Learner Control is concerned with the ways in which the student can affect the information flow along the L^0 path from the instructional program to the student. Where along the Control

continuum a given system will lie depends on the richness of its Student→Advisor path, and the Advisor→Instructional Program path, as well as the severity of the limitations imposed on the student by the Advisor.

2. Learner Control Research

Research on Learner Control has been severely hampered by the continuum nature of the control question as well as by the general state-of-the-art in CAI. Most studies compare "Learner Control" with "Program Control" as if there were exactly two distinct control possibilities. Most studies are so poorly designed that any conclusions at all must be suspect.

A few results are cited here, without any attempt to explain the experimental designs used, or the limitations placed on the results. The purpose is merely to give an idea of the sort of results obtained; it is not clear that they can supply much useful guidance for future CAI systems.

A number of studies permitted the student to determine his own sequence of topics for study. Of five such studies, three(Mager & McCann, 1961; Thomas, 1970; Grubb, 1969) found that the Learner Control(LC) group performed better than the Program Control(PC) group, while in two studies (Brown et. al., 1970; Oliver, 1971) there was no significant difference between the groups. When the student was also permitted to avoid some topics entirely, three studies(Judd, Buhderson & Bessent, 1970; Brown et. al., 1970; Fry, 1970) showed better performance for the PC groups. In two studies in which the student could control the amount of practice on a single skill, one favored the LC group(Dean, 1971), while the other

showed no significant differences between groups(Judd, Bunderson & Bessent, 1970). Various studies comparing time required for completion, choice of problem type, choice of instructional media, etc. show no clear advantage of one group over the other(Judd, Bunderson & Bessent, 1970; Brown et. al., 1970; Thomas, 1970; Newkirk, undated; Barnes, 1970; Mager & McCann, 1961).

The research to date does not contradict, and in some cases supports a number of assumptions which will be made about Learner Control. It is hoped that future research will be undertaken to validate or negate these assumptions:

1) Unrestricted Learner Control is harmful to some students.
2) Unrestricted Learner Control is helpful to some students.
3) A student must have background knowledge, advice, and experience in order to use Learner Controls effectively; in such cases, the student will benefit from Learner Control.
4) A student with Learner Control will be less anxious and frustrated, and will develop a better attitude toward the course.
5) Learner Control will increase the student's ability to work on his own, to seek out needed information, and to draw his own conclusions.

These assumptions are used to guide the selection and arrangement of Learner Controls as specified in Sections III and IV. In particular, it is assumed that availability of Learner Controls must vary from student to student, offering to each the appropriate measure of control.

3. Learner Control Commands

The exercise of Learner Control over instruction requires that the student be able to communicate with the computer at the L^1 level. The set of Learner Control(LC) commands described in this paper may be thought of as specifying a special-purpose programming language for this communication. While the LC commands are said to exist on the L^1 level, it is clear that their effects are felt on every level, particularly at the L^0 level.

No claim is made that the forty LC commands described here comprise either a complete or an exclusive set. As with any artificial language it must be judged on the basis of its adequacy and suitability for the task for which it is intended. The LC commands described in Sections III and IV are intended to deal with all aspects of CAI materials which are susceptible to student manipulation, given the current state-of-the-art in both computing and education. This is not to say that the LC commands perform all possible manipulations of the material, but only that all types of material are included. The specific commands are designed on the basis of past CAI experiences, some common educational beliefs, and the assumptions made earlier in this account.

In the descriptions which follow, the emphasis is placed on a functional description of each Learner Control, rather than an algorithm for implementation. To describe the LC's algorithmically would be to reduce their generality; the implementation must depend on the specific system configuration, and involves decisions which are in the province of the instructional designer or psychologist, not the computer scientist.

C. Implementation of Learner Control Commands

1. Learner Control and the CAI system

Learner Controls require a great deal of effort to implement. Since the time and programming ability of a given course author is likely to be quite limited, the usual result is the development of primarily linear programs, with no Learner Control. For this reason, the LC commands described in this paper are described in such a way as to be included within a larger CAI system. This means that the individual author need only select the LC's appropriate to his course, and make relatively minor changes to incorporate them.

The most straightforward way in which these LC procedures can be implemented is as system-defined subroutines. A special implementation of a command needed for a specific course could replace the system version for that course, if necessary. Most variations could probably be included as parameters to the system routine.

The manner in which a student uses the LC commands is system-dependent. The most direct procedure is for the LC's to be treated as system interrupts. Thus, the student presses a special "interrupt key" to identify what follows as a LC command, and then types the command. In a dedicated CAI system, frequently used LC's might have special keys so that the student can use them with a single key press. If a general interrupt capability is not available, it can clearly be simulated by inserting a call to an "interrupt routine" whenever a student input is expected.

The LC commands should be "human engineered." Any sensible input by a student should receive a sensible response. If an LC designed for use in a given situation has some other logical meaning in a different situation, it should be possible for the student to do it that way. The system should accept any distinguishable short form of any of the commands.

While it should be possible to implement most LC's in any CAI system environment, the difficulties involved may in some cases be prohibitive. Many of the special-purpose CAI languages and systems have very limited data storage capability, as well as extreme limitations on procedural definition capability. No attempt will be made here to specify all of the facilities useful for implementation of LC's; the EDUCOM study of CAI languages shows the wide range of elements which would be useful (Zinn, 1969). The requirements can best be summed up by stating that all of the capabilities of a general-purpose programming language, including string and list processing, would make implementation of LC's both easier and more powerful.

2. Dynamic control of LC commands

The availability and effects of a given LC command may vary, depending on the circumstances in which it is called. Often, the reason for the variation derives from the nature of the material being presented at the time of the command; such variations will be discussed later in connection with specific LC commands. In other cases the availability of LC's may depend on the progress being made by the specific student. An author may wish to limit the LC's available to a student who is "fooling

around" or encourage the use of LC's by students who do not take proper advantage of them. These decisions are the responsibility of the Advisor program shown in Figure 3.

There are a number of ways of implementing this variable control; one will be described here, and is used in the discussion of specific LC commands. For the purposes of discussion, let a given course be divided into units; let each unit consist of a number of lessons. Within-lesson structure will be discussed in detail later.

For each unit, lesson, and distinguishable element within the lesson, the author prepares a decision table for LC. This table contains switches and parameters which control the use of LC's. At execution time, each student has his own decision table. When the student begins a new unit or lesson, the values in his table are updated to match the specifications given by the suthor. Changes in the table values can also be made by the ADVISOR program, and by the execution of the LC's themselves.

When an LC command is executed, the values in the student decision table relating to that command are checked. Switches checked at that time might indicate that the LC is not available, or specify which of a number of actions is to be carried out. Parameters might specify a display to be made to the student, the location of a subroutine to be executed, or a transfer address.

3. The TICCIT system

Many of the LC commands described in this paper are being implemented as part of the TICCIT (Time-shared, Interactive Computer Controlled Information Television) CAI system currently under development by the MITRE Corporation, under NSF sponsorship(Stetten, 1971). For this reason, the TICCIT project will be used as an example in many of the descriptions which follow.

The TICCIT system will use a dedicated computer system, involving two minicomputers (16-bit, 32K main processor, 12K terminal processor) with virtual memory, and an ALGOL-based special CAI compiler. A pre-processor will simplify the programming task still further. The terminals used will be ordinary television sets with special adapters and keyboards for CAI use. The system will be used initially in selected Junior Colleges; four courses are being developed in basic mathematics and English with this setting in mind.

The TICCIT system is still in the process of development. Some of the elements attributed to TICCIT in this paper will undoubtedly be modified before the project is completed. The references to TICCIT in this paper are intended as an aid to understanding of Learner Control commands, and not as a description of the TICCIT project itself.

III. Course-Independent Learner Controls

This section describes a number of Learner Control commands which might be implemented across courses in a CAI system. This is not to say that a given command would operate in the same manner at all times, but rather that its general form and purposes are not affected by or dependent on the way a single course is structured. What follows is a description of the purpose of each command, and the actions it initiates.

STOP (or SIGN OFF or LOGOUT)

The purpose of this command is to permit the student to terminate a session at the terminal. A course author would prefer that the student stop only at "logical" points in the course; however, the individualized nature of CAI ensures that different students will see different sequences of material and complete them at different rates. Thus, a STOP request must be possible at any time.

Of course, the real difficulty is not in stopping, but in resuming the instruction at a later time. This requires that the system be able to store information about the student's current status and position in the course for use when the student begins work again. Many CAI systems simplify the problems involved by requiring the author to specify certain "restart points" at which specified data about the student is stored. Regardless of the point at which the student ends a session, he can resume work only at the most recent restart point encountered. The unfortunate student is thus forced to repeat work already completed satisfactorily.

A better solution is to handle the problem in much the same way that a timesharing system swaps a program out of core. All information is retained as is, and can be restored when the student resumes work exactly as it was when he left off. This procedure is quite easy for a CAI system to implement--but very difficult for an author not working within such a system.

The STOP routine performs certain additional functions. While restart may be possible at any point, it might not be desirable. At certain points, the author may specify that a student asking to STOP be told something like: "If you stop now, you must restart at the beginning of this section" or "Since you are in the middle of a test, if you stop now, you will be scored as having failed the test." Should the student still wish to quit, the STOP routine must carry out the necessary changes in the student's records before signing him off. In addition, there may be certain statistics kept on each session which must be stored. Finally, the author may specify that a homework assignment be given when the student signs off. The decision table structure described earlier can easily handle all of these options.

BACK and FORWARD

These commands are quite important for a student using a Cathode Ray Tube(CRT) terminal. Unlike a teletype user, the student at the CRT has no automatic hard-copy record of his past work. Once a given display is erased, he must have some way of seeing it again. The BACK command provides this mechanism. It is intended as a reminder or review mechanism by which the student is permitted to look again at earlier displays in a

static fashion, without being able to change the responses he had made.

The implementation of this command will depend on the system and course structure. The procedure to be used in the TICCIT system is described here. The system maintains a pushdown stack of display labels of fixed size. A new label is added whenever a new display is presented to the student, causing the oldest label to be lost. Successive BACK commands cause the student to move deeper into the stack, seeing one display per command. A generated display may not necessarily use the same values as those originally seen by the student; also, student responses are not restored. The student can move up the stack in an analogous manner, one display at a time, using the FORWARD or SKIP command. The ONWARD command moves him directly back to the top of the stack.

COMMENT and CUSS

Even the most flexible and carefully designed CAI course cannot hope to handle all situations properly. Some questions will prove ambiguous, some answers will be misinterpreted by the system, some reasonable actions the student would like to take will prove impossible. The COMMENT command provides a means by which the student can make known his feelings about course contents, and a way for the course author to discover and correct such problems. The COMMENT facility provides a direct communication between student and author in L^3, and so cannot affect the immediate sequence of instruction. The system can only record the comment itself and information about the precise context in which it is made, for later use by the author.

The TICCIT system includes an additional complaint mechanism requiring less effort by the student to record his complaint. This is the CUSS command, a special button which, when pressed, causes a multi-colored series of odd punctuation marks to be displayed, along with "beeps" from the audio unit. CUSS commands are recorded in the same manner as COMMENTs.

CALC

Execution of this command makes available to the student (if the author permits) a desk calculator program, or simple programming language. Examples of such programs include APL, FRED, and BASIC. This is one way in which the power of the computer can be made available to a student in a CAI course. This is a particularly useful tool when teaching subjects like physics or mathematics. However, the enthusiasm of many students for using such programs makes them a valuable motivational technique for any course.

The CALC command is intended as a means for the student to get access to such programs on his own initiative. The availability of such facilities makes it possible for the author to prescribe programming exercises directly. This can be quite a valuable tool in many areas (see, for example, Feurzeig & Papert, 1968).

SYSTEM

This command is of most use when the CAI system is imbedded in a general computer utility. In this case, the SYSTEM command is a request by the student to let him leave the course temporarily and do some other work within the computer system.

WAIT

Many interactive systems will issue a query if no input is received within a certain time period, and then log the user out automatically if no input is received. A CAI course may also set a specific time limit for responding to a given question. The WAIT command permits the student to request more time in which to respond. The course author may refuse to permit such an extension or may specify that some other action be taken when such a request is made.

COMMANDS

Execution of this command causes a display of all the available Learner Control commands, with an explanation of the use of each. This list may be dynamic, since a course author may restrict the LC's available at a given time.

GLOSSARY or DEFINE x

A student who encounters an unfamiliar term, or who has forgotten an earlier definition can use this command to obtain a definition of the desired word. Clearly, the author for each course must specify the glossary for his course. However, the retrieval routine for the glossary can be defined by the system. The routine may also record information about student use of the glossary for each course. Another type of glossary routine will be described later.

Control of special devices

A given system may make certain special devices available to the student. In the TICCIT system, for example, the student may request a hard-copy of the current display by using the COPY command. Each lesson also includes a videotaped introduction, which the student can see using the TAPE command. A given system might also include slide projectors, movie projectors, and tape recorders for which commands would be available.

IV. Course-Dependent Learner Controls

The Learner Control commands described in this section are, in general, more powerful and more interesting than those already discussed. Their power derives from the fact that they take into account the structure and content of the course in which they are used. As a result, they also tend to require a greater expenditure of effort on the part of the course author.

The dependence of these commands upon specific course structure makes it necessary to specify the context in which each command is to be applied. This is somewhat difficult, given the wide variety of CAI types possible. The three continua defined in Section II may be of some use here. This discussion will assume a position as far to the right (see Figure 1) along the Control and Individualization continua as is possible. Any point to the left of these may then be obtained by restricting the use of the Learner Controls and individualizing mechanisms specified here. Such restrictions might be imposed by an author in order to satisfy the demands of a particular educational philosophy or instructional theory. The policy described here of maximized individualization and Learner Control does not constitute a recommendation on the part of this writer; rather it is a device to permit a complete and general description of LC commands.

Unfortunately, it is not possible to select a single point on the "frame specification" continuum at which to describe the LC commands. Instead, four different points will be selected, representing different ways in which CAI frames can be prepared. The four points chosen are not

mutually exclusive, but each has a different effect on Learner Control commands. The four types are:

1) Pre-programmed, complex branching CAI. This is by far the most common type of CAI, and the most general in its possible applications. For this reason, the description of LC commands will be based primarily on this type of CAI. The other types will be mentioned only when the use of LC's differs from this.

2) Inquiry or Information-retrieval based CAI.

3) Natural language CAI

4) Generative CAI

The discussion which follows is divided according to the different sorts of activity which take place within any CAI course, regardless of type.

A. Expository material

This mode of CAI performs a function analogous to the classroom lecture. Here the computer is presenting the student with facts, figures, definitions, descriptions and examples of concepts, rules, and so forth. It is the most difficult type of material for the computer to manipulate--and few CAI programs have attempted to function as more than "page turners" when presenting such material. Complete individualization of such material will require further advances in the processing of natural language text and in graphics generation; some more limited procedures are described in this section.

1. Selection and sequencing of topics

In almost any field of knowledge, there is a traditional order of instruction which the instructors claim is the most "logical" order. Mager(1961) points out that this order is logical only to the instructor--the student's needs are quite different. In fact, some studies have shown that when the normal order of instruction is completely scrambled, student performance is not affected significantly (see discussion in Oliver, 1971). This is not to say that instructional sequence is of no importance; rather, it is an attempt to suggest that the student can safely be permitted to modify that sequence, within reason. The development of learning hierarchies showing the interrelations among the various parts of the subject matter(Gagne, 1968) can provide a useful basis for determining what limitations, if any, must be imposed on the student.

SCHEDULE - Upon execution of this command, the student is shown his schedule for the rest of the course. He is also given a description of the ways in which he may modify this schedule. The extent to which modification is permitted must be decided by each course author. If the scheduling is to be left up to the student, the system should at least be ready with suggestions to assist in that decision.

In the TICCIT system, the original schedule is based on a learning hierarchy developed by the course authors. The only schedule modification permitted the student is to reorder the lessons appearing at a single level of the hierarchy. In a fairly non-hierarchical subject such as English composition, this provides significant power. In a more highly

structured field, such as mathematics, it is of less value. When a student fails a given lesson, it must generally be repeated later, and the student is also permitted to modify this scheduling.

SURVEY - By executing this command, the student can look at any lesson which he has not yet taken. This capability is particularly important if the student is to be able to make meaningful instructional decisions for himself. It can provide information useful in determining a schedule, and in understanding the way elements of the course fit together. It must be noted that the requirements for a survey mode are different from the normal instructional pass over the same material. The student should see only a part of the material, providing an overview of the rest. He cannot see the mastery test for the lesson, nor should his work be scored as if he were really taking the lesson. If a very large burden is not to be put on the author, all of these things must be handled by the system. In the TICCIT system, the student's decision table is adjusted to reflect the scoring changes, and certain "MENU" items are presented to form the survey. The MENU is described in a later section.

REVIEW - This command permits the student to look back at any lesson which has been completed. The student must be able to look at any part of the completed lesson--just as the student in the classroom always has access to his lecture notes and textbooks. Again, the requirements are different from the normal presentation; the student needs to be able to "skim" the material, looking at whatever he wants, and answering questions only when he wishes. Normal scoring procedures are eliminated.

ABORT - This command permits the student to stop work on a given lesson and go on to the next item in his schedule. Clearly, if this command is to be a privilege and not a punishment, it must be possible for him to return to the lesson later and resume at the point where he left off. This is much the same problem as the "restart" problem discussed earlier, and could be handled similarly. The ABORT command is likely to be quite costly in storage space, which may make it necessary to restrict its use, and to limit the amount of information which can be retained. This might force the student to repeat some work--which could be an advantage in this case, since it provides a review mechanism.

INDEX - This routine performs much the same function as a textbook index, i.e. a way for the student to discover where in the course some needed specific information might lie. He might then be able to look at the appropriate lesson using one of the commands already described.

The discussion in this section so far has been based on the branching model described earlier. In an inquiry system, the scheduling problem is left primarily for the individual student to solve. However, if such a system is to be used or instruction, it should have at least the possibility of an author-determined schedule. Many students would prefer to let the system choose the topics to be discussed, and the system should at least supply advice about possible topics and schedules. Otherwise, it is more of a reference work than a course of instruction.

Recognizing this problem, two programs have sought to build a CAI

system combining both inquiry and author-directed modes of instruction. One was developed by Carbonell(1970), the other by Wexler(1970). The general approach of the two systems is similar. Each uses a large data base, organized in semantic net form, as the central element of the system. It is interesting to note--and it might imply something about the generality of the approach--that both systems selected geography as the subject matter for the data base. However, both Carbonell and Wexler claim that by changing the contents of the data base and making minor changes to the retrieval and CAI programs, their systems could handle other subject areas as well.

In the inquiry mode, each of the two systems functions much like any information-retrieval system. In the tutorial mode, ordinary CAI frames are generated from the data base. More will be said about this generation in later sections. In the inquiry mode, the sequencing is clearly done by the student, while in the tutorial mode the problems are much the same as those described for normal branching programs. However, the question of control is not as important, since the student can always take control by reverting to the inquiry mode.

Generation and natural language processing are not major factors in the scheduling problem. Individualized schedules may be "generated" based on pretests showing the student's previous knowledge of the subject matter. Natural language would permit a freer discussion with the student of his scheduling options, which is a useful, but not vital addition.

2. Presentation style and lesson structure

A good human tutor tailors his presentation to his student almost automatically: the choice of words and examples, the relative emphasis placed on theory or practice, the amount of descriptive material needed, all can be varied with ease. The human tutor is not necessarily the best model for CAI; however, pending further research, it seems reasonable to assume that a good computer tutor should vary its presentations in ways at least as complex as those mentioned above. Due to the large number of man-hours or man-years required to program even a single presentation of course material, it is quite important to develop procedures which can generate a variety of different presentations automatically. A full capability for generation would require an ability to manipulate natural language beyond anything likely to be possible for many years. Even if this capability were available, it would still be necessary to develop decision algorithms which could select the proper type of presentation for each student. This would require major developments in instructional theory and research.

While these problems cannot be solved completely today, more limited solutions are possible. The need for a decision algorithm can be avoided by leaving the decision to the student. A variety of presentation styles can be achieved by chopping the information into small categories and blocks, from which a variety of presentations can be built. Two such schemes will be discussed here. The first is the "Information Mapping" procedure developed by Horn(Horn, et. al., 1971). Using this scheme, each part of the course material is classified according to the

type of learning involved. Each learning-type calls upon a different "information map." Each map includes a variety of "information blocks" which are used to explain the course material in appropriate ways. A chart of different information maps and blocks is given in Figure 4.

The TICCIT system uses a similar procedure, based on a taxonomy of instruction and learning paradigms developed by Merrill(Merrill & Boutman, in press). After the course is divided into lessons and units based on its learning hierarchy, each lesson is analyzed in detail. Groups of frames are prepared dealing with each aspect of the lesson: definitions, concepts, examples, rules, etc. Certain standard items are always included: introduction, "minilesson" (this is an abbreviated version of the lesson useful for review, survey, and advanced students), objectives, summary, "So what?" (reasons for studying this topic), mastery test, "fun options" (games and other "fun" material using the information in the lesson), etc. A MENU is then prepared showing all of the available items for the lesson. Figure 5 gives an example of a lesson menu.

Once an information map or a lesson MENU has been prepared, a wide variety of presentation strategies can be developed to fit the needs of different student. Selection of specific strategies can be accomplished under program control, student control, or a combination of the two. The TICCIT system utilization of these structures should suffice to show their power; of course, it is only an example of the ways in which they may be used.

When a student begins a lesson for the first time, the MENU for that lesson is automatically displayed, and he is asked to select the items

Types of Maps	Description	Information Blocks
Concepts	A concept may be a: . technical term . generalization sentence . property sentence . rule sentence . relationship sentence.	. name of the concept . definition or description . theorem (or generalization) . formula . use . rule . example . non-example . introduction . synonym . notation . diagram . comment . analogy . related maps
Structures	A structure is: . a physical thing, or . something which can be divided into parts which have boundaries.	. name of structure . meaning . function . /all of the concept blocks/ . parts and subparts
Processes	A process is some structure changing through time. The description of a process involves writing about what happens during successive stages of time.	. name of process . /all of concept blocks/ . /all of structure blocks/ . stage table . parts-function table . cycle chart . input-output table . cause-effect table . block diagram . PERT chart . WHIF chart . state table . condition . cause . effect
Procedures	A procedure is a set of steps performed to obtain some specified outcome.	. name of procedure . /all of concept blocks/ . procedure table . flowchart . occasion for starting . when to stop . decision table . check list . work sheet
Classifi-cations	Classification is the sorting of things by concepts into categories by the use of one or more sorting factors (criteria).	. classification table . classification sheet . classification list . outline
Facts	Facts are sentences about things done, things that are or were in existence, events, conditions, and so on, and are presented without supporting evidence.	. statement of fact
Proofs	Proofs are generally used in mathematical subjects for more difficult theorems.	. name of proof . assumptions . to prove . statement . reason . example

Figure 4: Information Map Classification Chart(Horn, et. al., 1971, p. 151)

Lesson Menu

Second degree polynomial functions

1. Videotape
2. Objectives
3. Review tips
4. So What?
5. Mini-lesson
6. Definitions
7. Instruction: zeroes, extreme point, y intercept, graphing
8. Instruction: algorithms
9. Mastery test
10. "Fun options"

Figure 5: Sample lesson menu(adapted from Bunderson, 1972)

he wishes to see from that MENU. The Advisor program built into the system may offer specific suggestions to guide this choice, based on the student's past performance; however, the decision is up to the student. He may add to, or change his selections at any time, as described below. The only restriction is that the student must take and pass the mastery test before moving on to the next lesson.

The MENU may be modified by the system in any way desired. In SURVEY mode the student is only permitted to see certain MENU items (such as the minilesson). A student who has failed the mastery test might be forced to look at certain items before retaking the test. The system might then require the student to review certain items in an earlier lesson.

The following are some LC commands relating to this area:

MENU - This command causes the MENU for the current lesson to be displayed, along with the list of items the student has selected from it. The student may modify his selections if desired. After the student has seen the MENU and made any desired changes, he may continue from the point where he left off.

ONWARD - This command permits the student to terminate work on a given MENU item, even though it is not yet completed. This mechanism permits each student to terminate a given sequence as soon as he feels he has mastered the material. Upon execution of the ONWARD command, the student is branched to the next item in the list he has selected; if no

items remain, he will be branched to the MENU. In either case, the point at which he left off is retained, so that he may resume work on the same MENU item at a later time, if desired.

QUIZ - The student uses this command when he is uncertain as to whether he understands the material he has been going over, and he wishes to be quizzed on it. The effect of the command is to present to the student one or more practice questions covering the material--differing from the mastery test in that the questioning is for the student's own use and "does not count."

MORE - This command is used by the student to indicate an interest in the subject being discussed, and a desire to know more about it. The author programs additional material on each subject, which the student can access with this command. The material displayed might be a note on the history of the subject, its applications, or some other supplementary information which might interest the student. This information might be displayed automatically to advanced students as enrichment material.

In an inquiry mode of CAI, the type of presentation made will be chosen by the student. However, the problem of implementing a variety of presentations remains much the same as in more structured forms of CAI. Again, the primary variation possible is in the selection of specific bits of information to present on a given subject. If the data base is thoroughly classified and structured down to the lowest levels, then it is easy for a student to elicit any specific bit of information desired. The problem arises when the student makes a more general inquiry. If a

student says "Tell me about Brazil," how much does he really want to know about Brazil? In what order should the information be presented? In Wexler's and Carbonell's systems, this problem is handled by attaching to each category of information a weight statistic, indicating its relative importance. In this way, the more "important" information can be presented first. However, this weight should really be a variable quantity, based on the individual's background and interests (for example, a geology student might be more interested in soil type than names of major cities). It should be possible to develop such variable weights, at least within the limited contexts implied by a CAI course.

Application of generative techniques to expository material is still quite limited. It is sometimes possible to generate example files, particularly in mathematical areas. Generation of textual material is another matter. The procedures discussed in this section indicate the maximum that is currently possible in this area.

The availability of effective natural language processing techniques would be a major aid to the development of generative procedures for expository material. As already noted, such techniques are a prerequisite to the complete individualization of such material. They could reduce the task of the CAI author considerably, making it possible to develop CAI materials from a data base or from standard textbooks(see Uhr, 1964, 1967). High quality materials can be developed in this way only if procedures are also implemented to ensure that the materials developed are instructionally effective. In any case, such methods remain far beyond current capabilities.

B. Inquisitory material

This section is concerned with the CAI mode in which the student must answer questions or solve problems. The purpose of this is to practice the concepts and rules he has learned; questions asked for purposes of testing will be considered in a later section. Since the questions are intended as a learning aid for the student, it is reasonable to permit him to modify the question process with LC commands.

1. Amount of practice

Varying the number of questions asked of different students is clearly very easy to do. Deciding when the individual student is ready to move on, is a more difficult proposition. Normal testing procedures are of little help--they are designed to tell the instructor that the student has already learned the material, not to pinpoint the moment at which the learning is completed. One question may be all some students really need, but statistical decision algorithms can rarely make useful predictions based on only one question. This seems to be another case where the decision is best left to the student, at least initially; if he has not really learned the material, he will be caught by the mastery test later.

Some restrictions are necessary. The author may wish each student to be exposed to each problem type. Also, many students would rather let the program decide when to move on, or at least provide them with advice about their possible actions. The TICCIT system is designed to meet these requirements. The student can move on when desired using the ONWARD

command already described. Procedures are implemented for the system to "guess" the amount of practice a given student needs based on his past performance, and his degree of success on the current problem set. This information is used to advise the student at appropriate points, or when asked.

FASTER and SLOWER - When the amount of practice decision is made under program control, and sequencing is handled by the system, these commands can be used to give the student a limited voice in the decision. Essentially, the author is permitting the student to modify the parameters in the decision algorithm, within prescribed limits. A FASTER command might reduce the number of problems seen by the student, while a SLOWER command would increase that number. Data gathered on use of these commands across students should help the author to revise the decision parameters.

2. Problem type and difficulty level

There are many ways to characterize the dimensions along which questions and problems on a given subject may differ, often based on complex instructional or psychological theories. Since the concern here is with Learner Control, it seems most useful to select a characterization which is meaningful to the student. From the student's point of view, two main dimensions emerge.

The first is fairly well understood and deals with the form of the question and answer; common types include multiple choice, true-false, constructed response, and essay questions. Educational psychologists have developed a variety of theories as to which type is best for any given

purpose. These theories need not be discussed here. Regardless of the relative educational merits involved, it is certainly true that a CAI program will have much better success at interpreting the responses to a multiple choice question than an essay question.

An author could clearly include questions of all types in the practice material for a given lesson, although it would involve a large amount of effort. Each student would probably see some questions of each type, with some algorithm for selecting them included in the system. A QTYPE command would permit the student to increase the proportion of whatever question type he preferred.

It also seems possible to implement generative procedures in this area; that is, an author should not have to program all types of questions. It should be possible to develop, for example, a number of true-false or fill-in questions from a single multiple choice question written by the author. It is likely that this process would occasionally go wrong, but as long as the questions are used for practice and not testing, this should not matter. No such generative procedures have actually been implemented, but it should not prove too difficult.

CAI questions can generally be programmed in one of three ways. The simplest (and most time-consuming) is to program the exact form and content of each question explicitly. The second is to develop a single format or paradigm from which a number of questions can be built. The variable elements in the paradigm are filled in from tables of values prepared by the author(Boessenroth, Smith & Gregory, 1970). The third method is similar to the second; however; the values are generated by some

algorithm, rather than taken from a table(see, for example, Peplinski, 1968; Uhr, 1970). The Carbonell and Wexler systems go even a step further; complete questions are generated directly from the data base. However, it is not clear that the overall quality of these questions is sufficient for actual student use; the development of non-ambiguous, non-trivial questions is not an easy task.

The second dimension along which questions may vary is that of "difficulty." This dimension is not as easily analyzed. Certainly, some questions are "harder" than others, and the most difficult question for one student will not be the most difficult for another. Educational psychologists have a variety of explanations for this which might supply a good guess as to the difficulty of a given item--with a high probability that the guess will be wrong much of the time. Still, that guess, combined with experience with actual students answering the questions, can be of use, as described below.

Regardless of the method used to create the questions, each question must have associated with it a "difficulty weight." For the first two methods discussed above, this weight can be derived using any combination of two procedures: First, the author may assign a weight based on his own analysis of the difficulty of the question. This might be based on instructional theory or the author's subjective impressions of its difficulty. Second, data can be collected by trying the question out on many students, and a weight assigned based on that experience. This weight can even be updated dynamically as students go through the course. In the case of

generated values, the algorithm used for the generation can be designed to specify a difficulty weight for different types of values.

Once these weights have been assigned, the system can easily take them into consideration in selecting items for each student. The basic procedure is simple enough: If the student gets a problem right, he is given a harder one next, otherwise he gets an easier one. The procedure can be expanded to ensure that each student sees both easy and difficult problems, and to take into account the magnitude of change in difficulty weights desired in any given circumstance.

HARDER and EASIER - These commands permit the student to manipulate the parameters of the algorithm just as described. The author can, of course, limit the student's ability to do so. However, these commands, along with the FASTER and SLOWER commands, give the student a valuable tool in expressing his own needs and wishes which the author is probably well advised to take into account.

3. Feedback

In the preceding sections the concern was with the questions being asked; here it will be with the answers and the program's reaction to those answers. One of the major advantages generally cited for the use of CAI is its ability to provide immediate feedback to the student when he answers a question. A major topic for research has been the relative merits of different types and patterns of feedback. The options available in CAI include a large array of possibilities. A few samples:

When the student responds correctly -

- provide no feedback--just ask another question
- acknowledge that the answer is correct
- congratulate the student

When the student responds incorrectly -

- tell him his answer is wrong, and to try again
- tell him his answer is wrong, and supply the correct answer
- just supply the correct answer
- explain why his answer is wrong
- give a hint, and ask the student to reply again

Any of these possibilities can be handled fairly easily by a CAI system. The actual type of feedback given might vary at different points in the course and from student to student. Below are some LC commands which might permit the student to affect the feedback choices.

FEEDBACK - Upon executing this option, the student might be given certain choices of ways in which he is permitted to modify his feedback conditions, and allowed to select from among them. No special author effort is required to permit this--all the necessary information must be input anyway for the program to be able to judge the student's response.

WHY - The student using this command is asking for an explanation of why his answer is wrong, or another one was correct. For this command to be implemented, the author must supply such explanations to go with each question. In the case of a generated question, it may well be possible to generate the WHY explanation as well as the basic question and answer.

ANSWER - This command would most commonly be used when the student is asked to try again after supplying the wrong answer. He is essentially saying, "I give up, tell me." It is ordinarily scored as a wrong answer when choosing the next question. The command might also be permitted at other times, such as in REVIEW or SURVEY mode, or when the student simply doesn't want to bother supplying an answer, but would like to know what that answer is.

SKIP - This command is used by the student to avoid the current question entirely; he is not interested in the answer or the scoring--he simply wants to move on. Ordinarily, another question at the same level would be selected, and no penalty would be assessed. The author would have the option of scoring it as a wrong answer, if desired. Some restrictions might well be placed on the frequency with which the student may use this command.

RIGHT and WRONG - These commands would probably be available only in REVIEW or SURVEY modes, and would also be used by the author for debugging, and for demonstrations. When one of these commands is used, the system displays a correct or incorrect answer (depending on the command) and then treats it as if the student had typed it in. This permits the student or author to control the feedback and selection of questions in a convenient way

Something needs to be said at this point about the use of generative CAI in this area, particularly when the generator is a complex

program of the sort referred to by Siklossy as a "knowledgeable tutor" (1970, 1971). In such a case, the feedback provided can be quite complex--far beyond what is practical without generative techniques. For example, the computer can lead the student through its solution of the problem, step by step. Clearly, this will be helpful only if the procedure used by the computer is one which is also convenient and meaningful for the student. This can be a very useful tool, particularly in the teaching of complex problem-solving tasks.

C. Testing

Testing in CAI performs two functions. First, it satisfies the demand of our educational system that all students be graded in some way. Second, it permits the CAI program to analyze the student's progress with a fair degree of confidence in order to determine his future lesson plan. In a system with Learner Control, it acts as the primary check on that control. For that reason, the availability of LC commands is severely limited while testing is taking place. Since the tests usually follow a "tailored testing" model(see chapters by Lord and Green in Holtzman, 1970), which itself ensures an individualized testing procedure, the lack of LC's is not critical. There are some LC's which are possible in this area, and these are described below.

ITEM N - In certain cases where a group of test items is to be administered, the student may be permitted to "look over the test" as

he would in a classroom, before answering all the questions. Furthermore, while working within one section of the test, he may be permitted to change earlier responses in that section. In such cases, each test item is numbered visibly, and the student can branch to that item by using the ITEM command with the desired number after it.

SKIP - Uses of this command in other areas have already been specified. During testing, the command permits the student to avoid answering a test item. If the ITEM command is available, he will be able to return to the item later. If not, he is warned that the item will be scored as a wrong answer.

CONTRACT - In the TICCIT system, the course is graded on an A, B, Pass or Incomplete basis. "A" and "B" grades are based on additional work, demonstrating a higher level of knowledge of the material. This may involve extra MENU items, more complex practice items, and certainly an additional section to the mastery test. The student chooses the grade he wishes to work for at the start of the course. This choice may be changed at any time using the CONTRACT command. If the change is from a "Pass" to a higher grade, he is permitted to return to any lesson already seen, and take the additional material to try to raise his grade.

STATUS - Since grades are a part of the course, the student can use this command to find out his grade on each completed lesson, and his current course average. Advise about improving that grade might also be given.

A few words are necessary about generative CAI and about natural language in this area. Test items could certainly be generated in the same way that practice items are. However, this introduces an element of variation and uncertainty into the test results which is probably worth avoiding, even at the cost of additional author effort. Natural language would certainly be valuable for testing, even more than for practice items: A certain proportion of misinterpreted answers can be tolerated for drill and practice, but not for testing. Not only does it reduce the validity of the results, but it is likely to frustrate and enrage the student. Natural language would increase the ability of the CAI system to understand the student's input.

D. Learning aids

This section describes a few additional Learner Control commands which can provide the student with guidance or information upon request.

1. HELP or ADVICE

This command is one of the most difficult to implement, but it is also one of the most important to the student. Even in the most well designed system, there will be times when a student simply does not understand what is going on, or what he is supposed to do. In the classroom, he could always raise his hand and ask the teacher; a good CAI system should have a comparable mechanism.

Unfortunately, this calls for the computer to be a "knowledgeable tutor" in the fullest sense of the word, and this is far beyond the current state-of-the-art in computing. The best that can be hoped for is to program every sort of suggestion and difficulty which the author or system designer can think of. If this list is updated continually as a result of experience with students taking the course, it is likely to solve the student's problem at least a fair proportion of the time. When it does not, the student can be directed to the proctor or the instructor for human assistance. The human communication in L^3 can be a major safety valve in solving these problems of the student which are beyond the capability of the CAI program.

When HELP is requested, the student is given a choice of types of assistance available. This might include:

1) specific suggestions related to the current frame of instruction. For example, the author might supply a hint for solving a particular problem.
2) Advice on using LC commands.
3) Lessons on study skills.
4) Compendium of common student problems.

2. SUMMARY

This is particularly useful for a student in an inquiry system, or for one who is jumping around from topic to topic. Its effect is to provide the student with a summary of his recent activities. This is not the same as the summary item on a lesson MENU; that is a summary of the lesson, not of the student's specific work. The SUMMARY must clearly be generated specifically for the student at the time of the call. This might be done by abstracting the relevant aspects from lesson summaries, or it might be more complex. In the case of an inquiry system, it would have to be more complex. Natural language processing capabilities would certainly be useful for this command.

3. GLOSSARY or DEFINE x

The implementation of this command using specially defined glossary items has already been discussed. However, it is also possible for this command to give the student access to frames in the regular course material which define the given concept. This requires that the course material is organized in such a way that such definitions can meaningfully be presented separately from the surrounding material. The system must also make it possible to retrieve and display this information.

4. BANK

In the TICCIT system, each student has a "bank account" which he can "spend" as desired, and "earn" additional credit towards. This is basically a mechanism for shaping the student's behavior in various ways,

and controlling his use of LC's of some kinds. The student can spend his account on "fun options" and on commands such as CALC, COPY, and SURVEY. The cost of each item might vary from case to case; a student who "fools around" a great deal will find them more expensive than one who is progressing well. In general, this is one mechanism for insuring that the student uses his freedom responsibly. The BANK command itself simply lets the student find out his current bank balance. He might also be given suggestions about maintaining or using the account.

E. Specialized Learner Controls

All of the Learner Controls discussed in this paper are, at least to some extent, general. It is expected that they can be implemented for a wide range of course types and subject matter. However, it is also clear that any given course is likely to make possible some more specific, specialized LC's designed to fit in with its particular design and purposes. Any system of Learner Controls should make provision for addition of such LC's to the basic set available with the system. Once programmed, they should be written up and made available to other authors who might find them useful in their courses as well.

V. Summary

Using the model in Figure 3, Learner Control can be defined as any method by which the student can change the flow of instruction in L^0. While student communication in L^0 or L^3 can affect instruction by causing L^2 modifications, the only direct means of control available is via communication in L^1. The Learner Control commands which have been described in this account supply a means for student communication in L^1.

The LC commands were designed to meet two primary requirements. First, it should not be necessary to implement Learner Control procedures seaparately for each CAI course, but rather to include LC commands as part of a larger CAI system. This requirement is based upon the author's belief that Learner Control is likely to be widely adopted only if it does not require a major effort on the part of the individual course author.

The second requirement is that the LC commands be as general as possible. Any modification to L^0 which the state-of-the-art in computing will permit should be made possible by some LC command. In practice, the great complexity of the instructional process makes this requirement unreasonable; the LC commands described here were selected on the basis of an analysis of the instructional process. Each type of instructional activity found in any CAI course can be modified by one or more of the LC commands described in Sections III and IV; however, these commands do not and cannot include all of the ways in which any activity can be modified.

These two requirements were imposed in order to maximize the possible uses of LC commands. Any specific CAI project could probably adapt these commands to fit its particular needs and instructional theory. The commands are intended for use in just that way; they do not comprise an integral whole.

The author hopes that Learner Control will become a regular part of all CAI courses--whether or not it is implemented in the manner described in this account. As explained in the introduction, it is the author's belief that Learner Control can improve significantly both the instructional process itself and the student's attitude toward that process. It is hoped that the techniques which have been described here will be used to further this objective.

References

Atkinson, R. C., and Wilson, H. A., eds. (1969) Computer-Assisted Instruction: A Book of Readings. New York: Academic Press.

Austwick, K., ed. (1964) Teaching Machines and Programming. New York: MacMillan Co.

Barnes, O. D. (1970) The effect of learner controlled computer-assisted instruction on performance in multiplication skills. Ph.D. dissertation, University of Southern California, Los Angeles, California.

Boessenroth, T., Smith, A., and Gregory, C. (1970) Engineering Operational CAI. Technical Memo 1, Computer-assisted Instruction Laboratory, University of Texas, Austin, Texas.

Brown, R., Hansen, D. N., Thomas, D. B., and King, A. D. (1970) Learner Control of Automated Instruction. Technical Report: NAVTRADEVCEN 68-C-0071-3, Computer-assisted Instruction Center, Florida State University, Tallahassee, Florida.

Brown, J. S., Burton, R. R., Zdybel, F. (1972) A Model Driven Question-answering System for a CAI Environment. Technical Report 13, Department of Information and Computer Sciences, University of California, Irvine, California.

Bunderson, C. V. (1970a) Current Issues in the United States Regarding CAI. Technical Memo 3, Computer-assisted Instruction Laboratory, University of Texas, Austin, Texas.

Bunderson, C. V. (1970b) Justifying CAI in Mainline Instruction. Paper presented at National Science Foundation Conference on Computers in the Undergraduate Curricula, June 17, University of Iowa, Iowa City, Iowa.

Bunderson, C. V. (1972) Mainline CAI, necessary but not oppressive. In Proceedings of the Spring Joint Computer Conference. Atlantic City, New Jersey, May 16-18.

Carbonell, J. R. (1970) AI in CAI: An artificial intelligence approach to Computer-assisted Instruction. IEEE Transactions on Man-Machine Systems, MMS-11: 190-202.

Crowder, N. A. (1964) On the differences between linear and intrinsic programming. In Educational Technology (DeCecco, ed.) New York: Holt, Rinehart, and Winston, 142-152.

Dean, P. M. (1971) Learner Versus Teacher Controlled Arithmetic Practice. Paper presented at American Educational Research Association Conference, February 4.

DeCecco, J. P., ed. (1964) Educational Technology. New York: Holt, Rinehart, and Winston.

Feurzeig, W., and Papert, S. (1968) Programming languages as a conceptual framework for teaching mathematics. Proceedings of NATO Science Conference on Computers and Learning. Nice, France.

Fry, J. P. (1970) The interactive relationship between inquisitiveness and student control of instruction. Ph.D. dissertation, Michigan State University, East Lansing, Michigan.

Gagne, R. M. (1968) Learning Hierarchies. Educational Psychologist, 6(1).

Grubb, R. E. (1969) A Study of Differential Treatments in the Learning of Elementary Statistics. Paper presented at DAVI Conference, April 28, Portland, Oregon.

Holtzman, W. H., ed. (1970) Computer-Assisted Instruction, Testing, and Guidance. (Part IV: Individually tailored testing.) New York: Harper and Row.

Horn, R. E., Nicol, E. H., Roman, R., and Razar, M. (1971) Information Mapping for Computer-Based Learning and Reference. Cambridge, Massachusetts: Information Resources, Inc.

Judd, W. A., Bunderson, C. V., and Bessent, E. W. (1970) An Investigation of the Effects of Learner Control in Computer-Assisted Instruction Prerequisite Mathematics (MATHS). Technical Report 5, Computer-assisted Instruction Laboratory, University of Texas, Austin, Texas.

Mager, R. F. (1961) On the sequencing of instructional content. Psychological Reports, 9: 405-413.

Mager, R. F., and McCann, J. (1961) Learner-Controlled Instruction. Palo Alto, California: Varian Associates.

Merrill, M. D., and Boutwell, R. (in press) Instructional development, methodology, and research. Review of Research in Education, American Educational Research Association.

Newkirk, R. L. (undated) A Comparison of Learner Control and Machine Control Strategies for Computer-Assisted Instruction. Computer Science Department, University of Western Ontario, London, Ontario, Canada.

Oliver, W. P. (1971) Program Sequence by Ability Interaction in Learning a Hierarchical Task by Computer-Assisted Instruction. Technical Report 4, Computer-assisted Instruction Laboratory, University of Texas, Austin, Texas.

Pask, G. (1967) The control of learning in small subsystems of a programmed educational system. IEEE Transactions on Human Factors in Electronics, HFE-8: 88-93.

Peplinski, C. (1968) A Generative System for CAI Teaching of Simple Algebra Problems. Technical Report 24, Computer Science Department, University of Wisconsin, Madison, Wisconsin.

Siklossy, L. (1970) Computer tutors that know what they teach. Proceedings of the Fall Joint Computer Conference, 251-255.

Siklossy, L. (1971) Topics in CAI: Information Transfers and Review. Computer Sciences Department and Electronics Research Center, University of Texas, Austin, Texas.

Simmons, R. F., and Silberman, H. (1967) A Plan for Research Toward CAI with Natural English. Report TM-3623, Santa Monica, California: System Development Corporation.

Skinner, B. F. (1964) Why we need teaching machines. In Educational Technology (DeCecco, ed.) New York: Holt, Rinehart, and Winston, 92-112.

Stetten, K. J. (1971) TICCIT: A Delivery System Designed for Mass Utilization. Report M71-56, The MITRE Corporation, Washington, D. C.

Stolurow, L. (1969) Some factors in the design of systems for computer-assisted instruction. In Computer-Assisted Instruction: A Book of Readings (Atkinson and Wilson, eds.) New York: Academic Press, 65-93.

Thomas, D. B. (1970) Effects of Learner-Control and Prior Knowledge on Learning Mathematical Learning Models under Computer-Assisted Instruction. Department of Educational Research, Florida State University, Tallahassee, Florida.

Uhr, L. (1964) The compilation of natural language text into teaching machine programs. Proceedings of the Fall Joint Computer Conference, 35-44.

Uhr, L. (1967) Toward the compilation of books into teaching machine programs. IEEE Transactions on Human Factors in Electronics, HFE-8: 81-84.

Uhr, L. (1969) Teaching machine programs that generate problems as a function of interaction with students. Proceedings of the 24th National Conference of the Association for Computing Machinery, 225-234.

Wexler, J. D. (1970) Information networks in generated computer-assisted instruction. IEEE Transactions on Man-Machine Systems, MMS-11: 181-189.

Zinn, K. L. (1969) Comparative Study of Languages for Programming Interactive Uses of Computers in Instruction. Boston: EDUCOM.

Vita

Stephen Ronald Fine was born in Madison, Wisconsin on January 8, 1948, the son of Selma Fine and Isadore Victor Fine. After completing his work at West High School, Madison, Wisconsin, in 1966, he entered the University of Wisconsin at Madison, Wisconsin. During the 1968-1969 academic year, he attended the Hebrew University of Jerusalem, Israel. He received the degree of Bachelor of Arts from the University of Wisconsin in June, 1970. In September, 1970, he entered the Graduate School of the University of Texas.

Permanent address: 826 S. Midvale Blvd.
Madison, Wisconsin

This thesis was typed by Claire H. Fine.