DOCUMENT RESUME

ED 052 621 24 EM 009 081

AUTHOR Sass, Richard E.
TITLE A Computer-Based Instructional Management Program for Classroom Use.
INSTITUTION Pittsburgh Univ., Pa. Learning Research and Development Center.
SPONS AGENCY Office of Education (DHEW), Washington, D.C.
REPORT NO R-13
BUREAU NO BR-5-0253
PUB DATE May 71
CONTRACT OEC-4-10-158(010)
NOTE 76p.; Thesis submitted to the School of Education of Pittsburgh University

EDRS PRICE EDRS Price MF-$0.65 HC-$3.29
DESCRIPTORS *Computer Assisted Instruction, Computer Oriented Programs, *Computer Programs, Course Organization, *Curriculum Design, Data Bases, Elementary Science, Individualized Instruction, *Individualized Programs, *Learning Activities, Models

ABSTRACT

An instructional management program was developed to assist students in selecting learning activities. The program was based on a general model for specifying hierarchical curriculum structure. This model was developed using the directed graph, a mathematical form of a structural model. A hierarchy could then be generated from the curriculum designer's responses about the prerequisite relationships among the lessons in his curriculum. Using a student mastery data base, programs were designed to input data on a student's mastery of lessons, list data, and putput options for learning activities for students. The program to provide options eliminated lessons which students had already mastered and printed out an option only if all its prerequisites had been mastered. Also, feedback about activities chosen was returned to the designer. Flowcharts for these programs are included. Field tests of the programs in a first-grade science class led to the conclusion that its future application requires mature students, a relatively free instructional setting, a quick and reliable computer system, outside financing, close ties to classroom management, and a structured curriculum based on well defined objectives. (Author/JK)

UNIVERSITY OF PITTSBURGH - LEARNING R&D CENTER

1971/13

A COMPUTER - BASED INSTRUCTIONAL MANAGEMENT PROGRAM FOR CLASSROOM USE

RICHARD E. SASS

A COMPUTER-BASED INSTRUCTIONAL MANAGEMENT PROGRAM FOR CLASSROOM USE

Richard E. Sass

Learning Research and Development Center
University of Pittsburgh

May, 1971

The research reported herein was supported by the National Science Foundation and by the Learning Research and Development Center supported in part as a research and development center by funds from the United States Office of Education, Department of Health, Education and Welfare. The opinions expressed in this publication do not necessarily reflect the position or policy of the National Science Foundation or the Office of Education and no official endorsement should be inferred.

ABSTRACT

The purpose of the investigation was to develop an instructional management program based on a general model of curriculum structure that, when implemented, would promote self-direction by assisting the student in the selection of learning activities. A general model for specifying hierarchical curriculum structure was developed using the directed graph, a mathematical form of a structural model.

Based on the digraph model, a computer-based instructional management program was developed using the University of Pittsburgh's Time-Sharing System (PTSS). Using an interactive computer program, a structural hierarchy could be generated as a result of the curriculum designer's responses regarding the prerequisite relationships among the lessons of his curriculum. An analysis of his responses ruled out such structural ambiguities as circular and redundant prerequisite relationships among lessons. The student mastery data base was set up in the form of a linked list in order to facilitate accessibility to each student's record of mastery. Interactive computer programs which used the mastery data base included a program to input mastery data, a program to list the data, and the program used by the students to receive options of learning activities. The program to provide options used the information from the mastery data file to eliminate mastered lessons from a student's list of options, and permitted an option to be presented only if all its prerequisites had been mastered. The use of the digraph model for structure insured that changes in the curriculum required regenerating only the stored structural information. The program designed to provide feedback information to the curriculum designer provided for an after-the-fact examination of the initial structure based on the use of the curriculum materials. Feedback information included the percentage of students who mastered each lesson, the percentage of cases for which mastery of a prerequisite lesson was followed by mastery of the next higher-ordered lesson, and the percentage of cases for which an implied order of mastery was violated for two related lessons.

The utility of the management program was demonstrated by implementing it in a school setting which employed an individualized curriculum. It was applied to the first grade Individually Prescribed Instruction (IPI) Science classes at Oakleaf Elementary School in suburban Pittsburgh, Pennsylvania. Students were allowed to interact directly with Teletype terminals to obtain a choice of learning activities when it came time for them to receive a new prescription. At the end of the school year, the mastery information was used to verify the curriculum structure.

Observations of the operation of the management program in the school led to the conclusion that its future application requires mature students, a relatively free instructional setting, a quick and reliable computer system, outside financing, close ties to classroom management, and a structured curriculum based on well-defined objectives of instruction.

FOREWORD

The author would like to thank the members of his doctoral committee, Drs. Robert Baldwin, James Carlson, William Cooley, Richard Cox, and Leopold Klopfer, for their helpful suggestions, criticisms, and comments. He extends appreciation to Dr. Cooley, his faculty advisor, for encouragement and support throughout his doctoral study. For help during the field test of the management program, he would like to thank Mrs. Maria Clark of the Learning Research and Development Center, and Mrs. Peggy Gump and Mr. John Kirk of Oakleaf School. Credit goes to Mrs. Terri Komar for a typing job well done.

This report is based on the author's doctoral dissertation submitted to the graduate faculty in the School of Education in the fall of 1970.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. THE PROBLEM

The purpose of this investigation was to examine certain problems associated with the development of a computer-based management program for use with adaptive instructional systems. Adaptive systems, in which some aspect of the instruction is tailored to the individual differences of students, form the basis of what is commonly called "individualized instruction." The primary objective of the investigation was to develop a management program based on a general model of curriculum structure that, when implemented, would promote self-direction by assisting the student in the selection of learning activities. In addition, the utility of the program was to be demonstrated by implementing it in a school setting employing individualized instruction, and the problem of using the data generated to verify the curriculum structure was to be examined. The investigation consisted of the following three steps:

1. Use of a general model to specify curriculum structure.
2. Development of a computer-based management program compatible with that model.
3. Application of the management program to a portion of an individualized curriculum.

An early attempt to individualize instruction was the Winnetka Plan, described by Washburne (1922), which was based on the idea that time should be a variable unit within the curriculum, whereas achievement should be a constant unit. In describing the effect of this idea on the use of classroom time and evaluation procedures, he

pointed out that previous curricula were based on fixed blocks of time. Within the time units, pupil achievement varied with the student's ability. In contrast, the Winnetka Plan varied time to fit the students' capacities within fixed units of achievement. Each child was required to master certain essential skills, but at his own rate of progress. To develop units of achievement it was necessary to define which goals must be mastered and at what level, to prepare tests which covered the subject matter of each unit and diagnosed difficulties, and to prepare remedial practice materials to enable a student to make up deficiencies shown by the tests. Washburne asserted that only when achievement replaced time as the constant factor, could instruction be individualized to meet the needs of the child. More recent attempts to individualize instruction have centered around the concept of mastery learning, which is based on the idea of keeping achievement constant. Carroll (1963) and Bloom (1968) discuss the model, ideas, and strategy of the concept of mastery learning as it is generally used today.

Several programs involved in adapting the operation of schools to individual differences include project PLAN, a guidance management system for the classroom (Flanagan, 1969), the Individually Prescribed Instruction project (Glaser, 1968) for elementary grades, and the Primary Education Project (Resnick, 1967) for preschool and kindergarten students. In practice, the units of achievement that are to be mastered in these programs are under constant study. The material and methods of instruction are modified so that

they permit mastery of the objectives for each student in the shortest possible time.

The computer has been used in education to analyze data used in studies, to directly provide instruction to students (computer-assisted instruction), to test students (computer-assisted testing), to keep track of test results and other data for curriculum evaluation or teacher feedback (computer-managed instruction), and to perform operations involving more than one of these uses. Suppes (1966, p. 208) stated that the computer can aid in the individualization of instruction because it can be programmed to follow each student's progress and to use the information as a basis for selecting the new concepts to which he should be exposed next.

One of the reasons why computer systems have been less than a complete success in their use in individualization is that they typically require a thorough and complete specification of the curriculum to which they are to be applied. Once implemented, the systems tend to resist changes to the curriculum that would upset their smooth operation. Therefore, a computer system capable of aiding an individualized program should be adaptable in the face of change, even after it has been implemented. The demands of the computer system on the classroom personnel should be kept as light as possible, even though the system must be responsive to their needs. A further consideration shaping the form of the program that was developed was to avoid the spectre of computer control of the educational environment. Whatever its specific function, the

computer should be perceived by the student as a means of helping him to more effective learning. Ideally, a computer-based management program would, in the words of Cooley (1970), "serve as a device for helping the student to explore and control his environment rather than a device for controlling him in that environment [p. 25]."

To be useful, the management program was envisioned as a system that would be adaptable to changing curricula, that would be as simple as possible for the classroom personnel to maintain, and that would require storing the minimum amount of information about the curriculum and the students and still be effective. The program should be easily usable by either teacher or student, and it would not unnecessarily restrict either. The management program that was developed also allowed the student to determine his own instructional sequence, was generally applicable to any set of instructional materials based on the mastery of specified objectives, and provided for the examination of the curriculum in light of results of pupil experience within that curriculum.

The property of an individualized curriculum which suggested the development of a management program to provide a choice of instructional sequence was the "learning hierarchy" concept described by Gagné (1962). If a curriculum can be described by a set of hierarchically related learning tasks, it is possible to unambiguously specify a structure for the set of instructional materials corresponding to those tasks. That one might describe the structure of an

adaptive curriculum by using the learning hierarchy technique suggested the development of a general management program which would not be bound to one particular program of instruction. In order to be able to describe the structure of any curriculum, it was first necessary to develop a general structural model which was amenable to computer-based procedures. The management program was then designed to be compatible with the structural model that was used to specify the structure of the curriculum.

Since one objective of the present investigation was to design a management program which is generally applicable to various instructional schemes, it was important that the development of that program not be dependent upon a single idea of a learning hierarchy. Therefore, in referring to the body of the instructional program to which the management program was applied, the word curriculum was used. In describing how the parts of the instructional program related to one another, instead of using the phrase "learning hierarchy," the more general word structure was used. The instructional sequence is the order in which the parts of the instructional program were taken by the student. The structure of a given curriculum was specified by the curriculum designer as part of the input to the management program; the instructional sequence was determined by the student after interacting with the management program.

An actual curriculum structure was not specified until the management program was utilized for a specific application. The structure, specified in terms of the general structural model, was

generated as a result of implementing the management program. Similarly, the student data was generated as a result of using the instructional materials in the classroom. The management program can continue to be used even if the curriculum changes, whereupon the program can provide the vehicle for implementing that change.

The program chosen to demonstrate the applicability of the management program was the Individually Prescribed Instruction (IPI) project (Bolvin & Glaser, 1968). The curriculum chosen was one part of the Science program in the form that it was being used at the IPI laboratory school in the Baldwin-Whitehall School District in suburban Pittsburgh, Pennsylvania. Heimer (1969, p. 498) noted that the development efforts of the IPI program have been heavily influenced by the idea of learning hierarchies of instructional or behavioral objectives. Bolvin (1968) stated that "Precisely stated objectives permit the analysis of the behaviors required as prerequisites to a given objective. This analysis serves as a guide to the curriculum designer in sequencing and ordering the objectives [p. 239]." The role of the management program was to keep track of each student's achievement and to allow him to progress within the structural framework of the curriculum. The program was capable of gaining access to each student's mastery record, updating that record as additional material was mastered, and presenting the student with a list of instructional materials from which to choose suitable to his mastery record.

As summarized in brief form by Lindvall, Cox, and Bolvin (1970), the overall goal of IPI is "to develop an educational program which is maximally adaptive to the requirements of the individual learner [p. 34]." They assume that attention to the individual differences of the students can result in more effective learning on the part of all pupils. As part of the overall goal, six subgoals were listed as features of IPI designed to achieve individualization:

I. Every pupil makes regular progress toward mastery of instructional content.

II. Every pupil proceeds to mastery of instructional content at an optimal rate.

III. Every pupil is engaged in the learning process through active involvement.

IV. The pupil is involved in learning activities that are wholly or partially self-directed and self-selected.

V. The pupil plays a major role in evaluating the quality, extent, and rapidity of his progress towards mastery of successive areas of the learning continuum.

VI. Different pupils work with different learning materials and techniques of instruction adapted to individual needs and learning styles [pp. 30-34].

The six subgoals were synthesized from a survey of earlier writings (Bolvin, 1968; Glaser, 1966, 1968; Bolvin & Glaser, 1968; Lindvall & Bolvin, 1966), which reflected various statements of goals of the development effort of IPI. Since the computer system was developed to be applied to an individualized program, the aims of that program were examined to determine whether they were compatible. Goals I and II are concerned with mastery at a student's own rate, which is

characteristic of most individualized programs. Goals III and IV are concerned with involvement in the learning process and self-direction, respectively. The computer-based management program could contribute to these goals if it allowed the student to become involved in determining his own prescription for learning through use of the computer. Goal V is concerned with self-evaluation and goal VI is concerned with the technique of instruction. These could be furthered by the management program if it allowed the student to determine his own sequence of instruction and to choose specific learning materials which reflect his interests. If the management program provided the opportunity for self-direction and a sense of active involvement in the choice of learning materials, it would be compatible with these goals.

The spirit of a program of individualization is to allow a greater measure of freedom in the learning process. A computer system developed to be applied to such a program should contribute to that spirit. This consideration was a major factor in determining the form of the management program that was developed as part of this investigation.

II. DEVELOPMENT OF THE MANAGEMENT PROGRAM

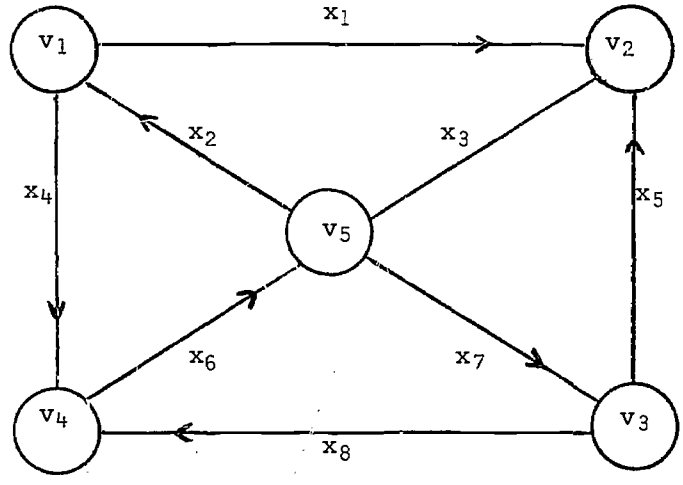
A. Model for the Structure

As a first step in developing a computer-based management program, a method of unambiguously specifying a general hierarchical structure was required. This led to an investigation of the general notion of structure. The problem of the description of structural properties in the natural sciences, social sciences, and behavioral sciences suggests applying certain branches of mathematics which deal with the abstract notion of structure. The work of Euler (1707-83) led to the fields subsequently known as topology and graph theory. Cayley (1821-95) advanced the field by his interest in structural problems in chemistry, as did Kirchhoff (1824-87) with his formulation of the laws of electrical network theory. The abstract notion of structure is treated in the mathematical theory of directed graphs, or digraphs as they are sometimes called. This theory is concerned with relationships among pairs of abstract elements. To the empirical scientist, the digraph can serve as a mathematical model of the structural properties of any empirical system consisting of relationships among pairs of elements. Applications range from communications systems in engineering to personality structures in psychology (Harary, Norman, & Cartwright, 1965, pp. 1-2).

The directed graph, or digraph, can serve as a structural model for any system which can be thought of in terms of directed relationships among pairs of elements. The digraph can serve as a model to describe curriculum structure if the instructional materials

or learning tasks are considered to be the elements, and the hierarchical relations of those tasks are considered to be the relationships among the elements. The type of relationship required by hierarchical relations can be represented by a special class of digraphs which contain no cycles, called acyclic digraphs. Digraphs which represent hierarchies are acyclic because the nature of a hierarchical relationship is such that all relations are directed in only one way: from the subordinates to the superiors. These ideas may be illustrated by considering two elementary examples.

A digraph can be represented in general by the set $\underline{V}$ of elements called "points" and the set $\underline{X}$ of elements called "lines." Each element x_i in $\underline{X}$ then corresponds to an ordered pair of elements (v_i, v_j) in $\underline{V}$, representing a line going from point v_i to point v_j. Consider the following representation of a digraph consisting of the set of points $\underline{V} = \{v_1\ v_2\ v_3\ v_4\ v_5\}$ and the set of lines $\underline{X} = \{x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7\ x_8\}$:



The lines of $\underline{X}$ can be represented in terms of ordered pairs of points of set $\underline{V}$ to indicate relationship between pairs of those points:

$$x_1 = (v_1, v_2)$$
$$x_2 = (v_5, v_1)$$
$$x_3 = (v_2, v_5)$$
$$x_4 = (v_1, v_4)$$
$$x_5 = (v_3, v_2)$$
$$x_6 = (v_4, v_5)$$
$$x_7 = (v_5, v_3)$$
$$x_8 = (v_3, v_4)$$

Let the ordered pair (v_i, v_j) be represented by the simplified notation $v_i \rightarrow v_j$. Then the lines of $\underline{X}$ can be represented:

$$x_1 = v_1 \rightarrow v_2$$
$$x_2 = v_5 \rightarrow v_1$$
$$x_3 = v_2 \rightarrow v_5$$
$$\vdots \qquad \vdots$$
$$x_8 = v_3 \rightarrow v_4$$

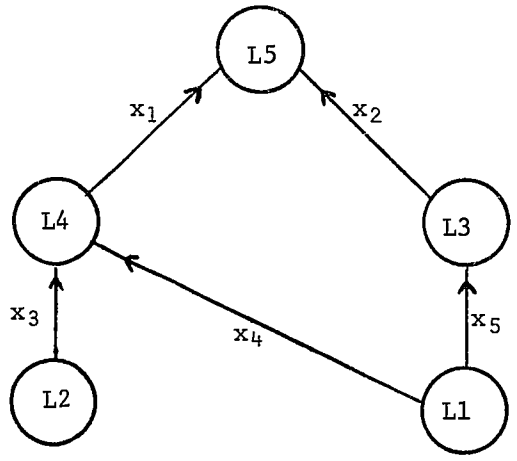
Note that this digraph contains four cycles that are represented by the following four sets of three lines:

$$\{x_1\ x_3\ x_2\},\ \{x_2\ x_4\ x_6\},\ \{x_3\ x_7\ x_5\},\ \text{and}\ \{x_6\ x_7\ x_8\}$$

Alternatively, the cycles can be represented in terms of the points by using the simplified notation. For example, the cycle $\{x_1\ x_3\ x_2\}$ can be represented by $v_1 \rightarrow v_2 \rightarrow v_5 \rightarrow v_1$.

As a second example, consider a simple hierarchy relating a set of five lessons, where lesson L5 is the terminal lesson of the set. Two lessons, L3 and L4, are both necessary for the mastery of

L5. Lesson L2 is necessary for the mastery of lesson L4, and lesson L1 is necessary for the mastery of both L3 and L4. In this example, the hierarchy may be represented by a digraph in which the lessons are represented by the set of points $\underline{V}$ = {L1 L2 L3 L4 L5} and the relationship "is necessary for the mastery of" is represented by the set of lines $\underline{X} = \{x_1\ x_2\ x_3\ x_4\ x_5\}$. The digraph is represented graphically as:



Note that each line of the set $\underline{X}$ corresponds to a relation consisting of an ordered pair of points in $\underline{V}$, or in simplified notation:

x_1 = L4 → L5
x_2 = L3 → L5
x_3 = L2 → L4
x_4 = L1 → L4
x_5 = L1 → L3

Note further that this digraph contains no cycles, which means that there is no point in the set $\underline{V}$ for which there exists some (non-trivial) sequence of lines whereby it is possible to travel in such a way as to return to that same point. It is obvious in this example

that if one starts from any of the five points one must always end at point L5, since there are no lines out from point L5. In general, the digraph of a hierarchical structure will contain no two mutually reachable points, i.e., there will exist no sequence of lines whereby one may return to a starting point, which is the definition of an acyclic digraph.

In order to specify the structure of a curriculum in terms of a digraph model, two things are needed: (1) the set of points $\underline{V}$ that define the curriculum, and (2) the relations which correspond to the set of lines $\underline{X}$ that define the hierarchical structure. In the second example, this meant identifying five lessons, L1 through L5, and the five relations which are represented by the lines. In addition, a digraph formed in this manner must possess the following properties if it is to represent a hierarchical structure. First, it must possess at least one point with no lines leading out from it. Such a point, called a terminal node, corresponds to the terminal skill, behavior, or lesson in the curriculum. It must also possess at least one point with no lines leading into it, corresponding to a basic or entering skill, behavior, or lesson. Next, the digraph must be acyclic, which in turn implies that it is possible to generate some kind of level structure by means of a suitable algorithm. A level structure based on distance up the structural hierarchy is a logical first step toward the graphical representation of the curriculum structure. Finally, in any hierarchical structure, the relations between pairs of points are necessarily

transitive in the set theory sense. This means that, in the second example, since L1 was necessary for L3, and L3 was in turn necessary for L5, the point L1 was necessary for L5, even though it was not explicitly stated. If it were explicitly stated in specifying the relational properties of the curriculum, it would have been redundant, and its inclusion would not have been necessary in representing the hierarchy by the digraph model.

B. The Processor to Specify Structure

The task of specifying the structure of a curriculum can be divided into two main areas: specifying the curriculum, or the points on the digraph model; and specifying the hierarchical relations, or the lines on the digraph model. An interactive processor program was developed to allow the curriculum designer to specify the structure while working at a remote computer terminal. This processor program was implemented using the University of Pittsburgh's Time-Sharing System. The hardware consists of IBM's System 360/50 configuration with remote IBM 2741 terminals located throughout the University. The programming language used for this program was CATALYST/PIL (Dwyer, 1969), which is supported by the time-sharing system, was developed to handle interactive programming tasks, and has been interfaced with an interpretive language. This language was chosen because of its ability to handle large blocks of text while using a relatively small amount of core storage, and because of the relative ease by which file manipulation is possible.

The two files created by the processor program, the lesson identification file and the structure file, were designed to be compatible with the rest of the management program. In the time-sharing system, each file that is saved is referred to as a dataset, which is identified by a dataset name followed by a three-character attribute. In this application, the attribute stood for the interpretive language by means of which the datasets were created, namely "PIL." The two output files from this program were stored on disk as datasets named LESSON:PIL and STRUCTURE:PIL, corresponding to the lesson identification file and the structure file, respectively.

A basic outline of the processor to specify structure is represented by the flow chart in figure 1. The first section of the program allows the user to identify the points of the curriculum, which are called lessons for the purpose of the management system. There are three parts to the identification of the lessons. The first part is the lesson code, which consists of from one to four characters that serve to identify that lesson throughout all phases of the management system. The code was necessary because of its conciseness; it must be entered often by means of the remote terminals. The second part of the identification is the lesson name, by which the lesson is recognized by the curriculum designer and by the classroom personnel. A third part is called the description of the lesson. It consists of information for the benefit of the student using the management program to help him make a choice among lessons. The name and description for each lesson are limited to about 115 characters,

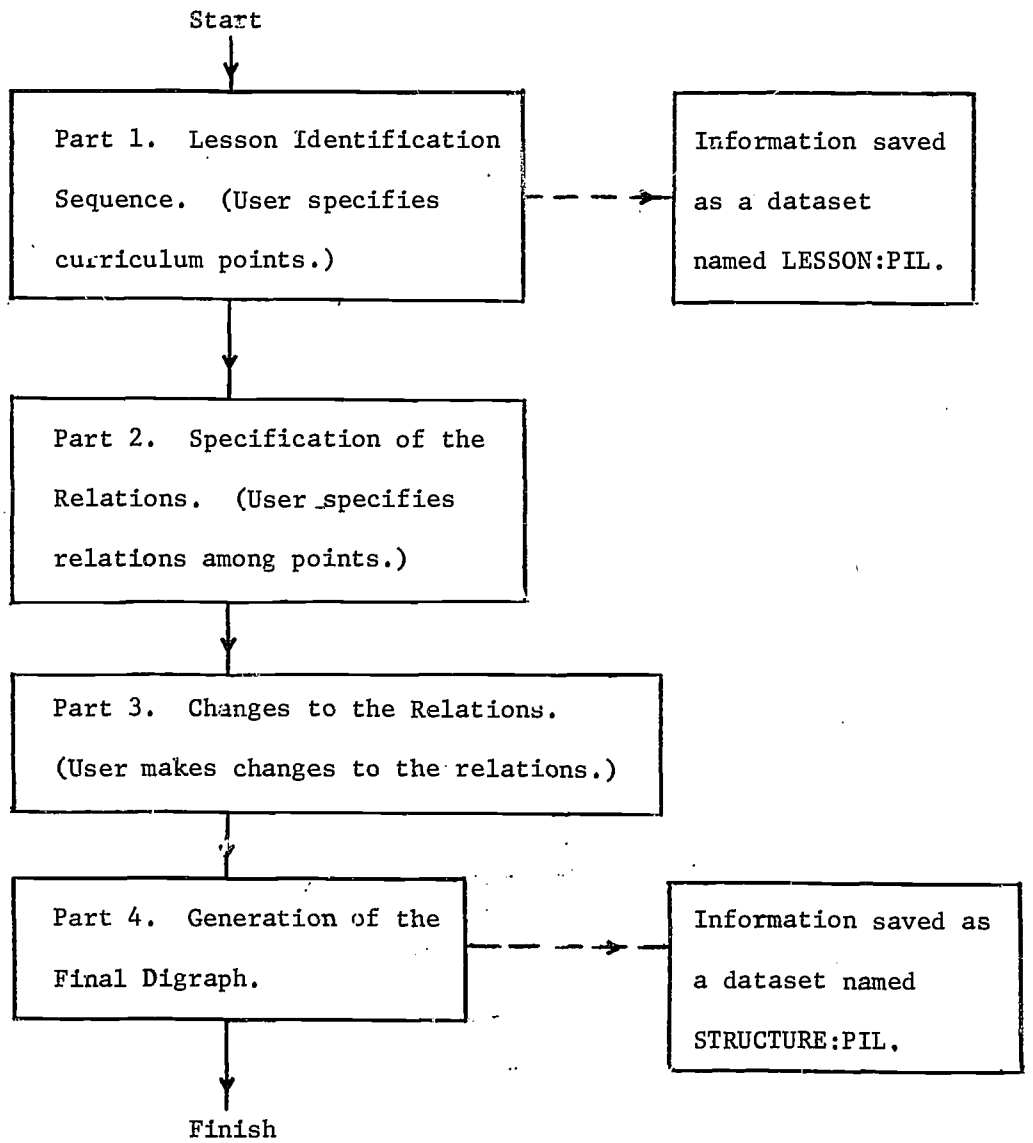


Figure 1. Processor to Specify Structure

including blanks. Because the IBM 2741 terminals include both upper- and lower-case letters, the characters used for the lesson codes are forced to upper case so that the management system will be compatible with the use of remote terminals which do not include lower-case letters. The lesson identification sequence is described in detail in the flow chart in figure 2.

The second section of the program allows the user to specify the prerequisite relationships among the points of the curriculum, i.e., to specify its structure. The program determines structural relationships by presenting one lesson and then asking which of the remaining lessons are necessary for its mastery. As a starting point for this process, the user is asked to identify one or more terminal lessons, ones which are not necessary for the mastery of any of the other lessons. A terminal lesson corresponds to a terminal node of a digraph: a point with no lines leading out from it. The procedure of starting with the terminal lessons is similar to that used in task analysis; i.e., starting with desired outcomes and working downward to more basic skills.

The structural relationships, or relations, between pairs of lessons are determined by the following procedure. Starting with the terminal lessons and continuing downward until all lessons are covered, the interactive program asks the user, "Which lessons are necessary for the mastery of lesson X?" The user enters lesson codes for all lessons directly prerequisite to the lesson in question. Only those lessons immediately "below" the lesson in question need

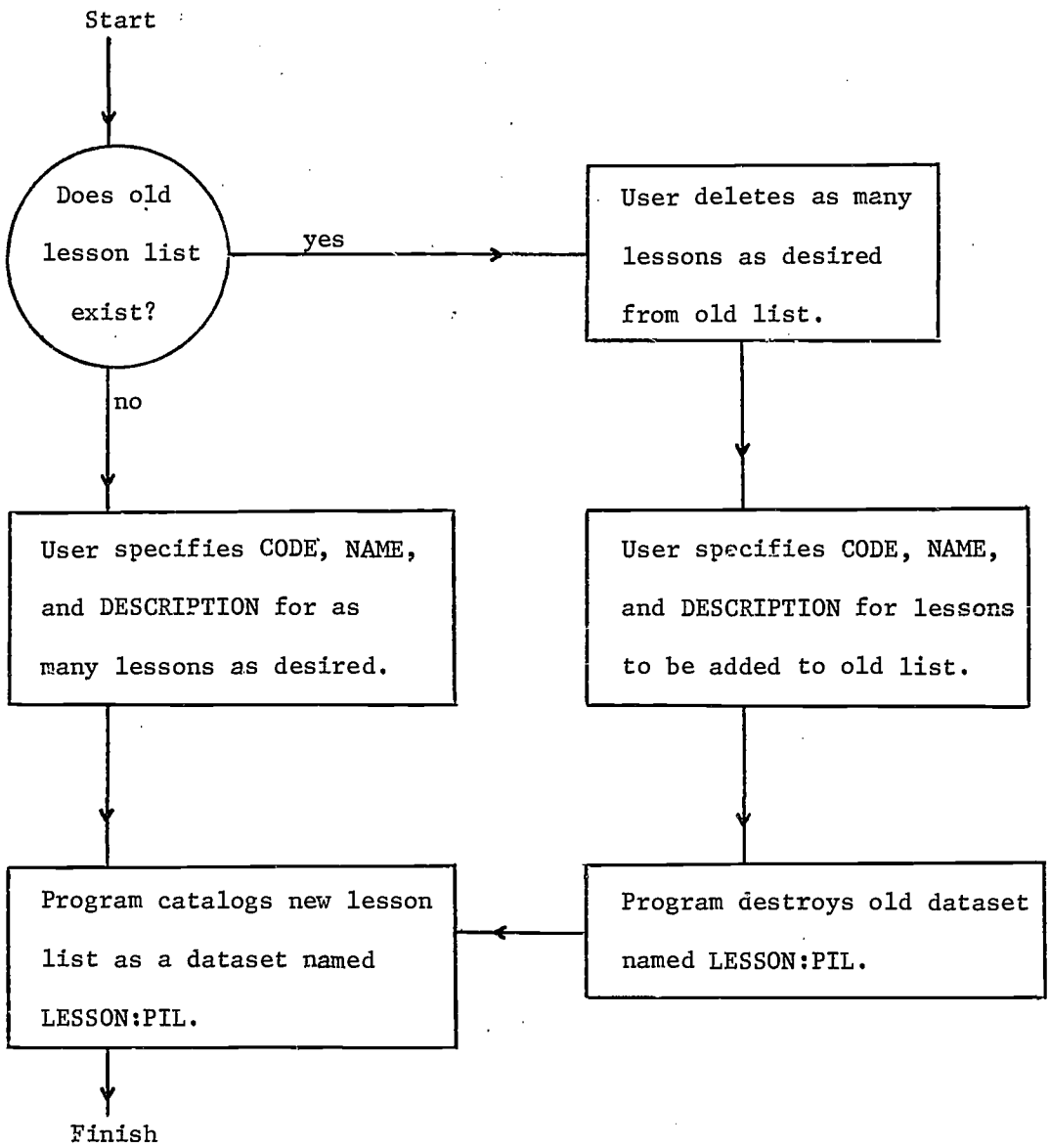


Figure 2. Lesson Identification Sequence

be entered. The specification of the relations is represented by the flow chart in figure 3. The question regarding necessary lessons is asked just once for each lesson which was specified in the first section. The result of this procedure is that a table is set up that allows each lesson to be linked with as many other lessons as were specified. The use of the interpretive language is advantageous because it allocates storage for each link only as it is specified, thereby eliminating the need for the dimensioned array of all possible lesson links.

The third section of the program allows the user to reflect on the structure he has specified and to modify it by adding more relations to it or by deleting some of the relations that were specified. In addition to adding or deleting relations, he may also obtain a list of all the relations so far specified, or a list of the relations involving just a single lesson. In this way he may check to see that what he has so far specified is indeed what he wants. This part of the program is represented by the flow chart in figure 4. The user may obtain a list of the relations he has specified and go back and change them as many times as he wants. Once he has signified that there are no more changes, however, the structure he has created is analyzed and if no errors are found, the information is saved as a dataset. To make any changes past this point, he would have to rerun the processor program.

In the fourth section of the program, the structure that the user has specified is fitted to the acyclic digraph model around

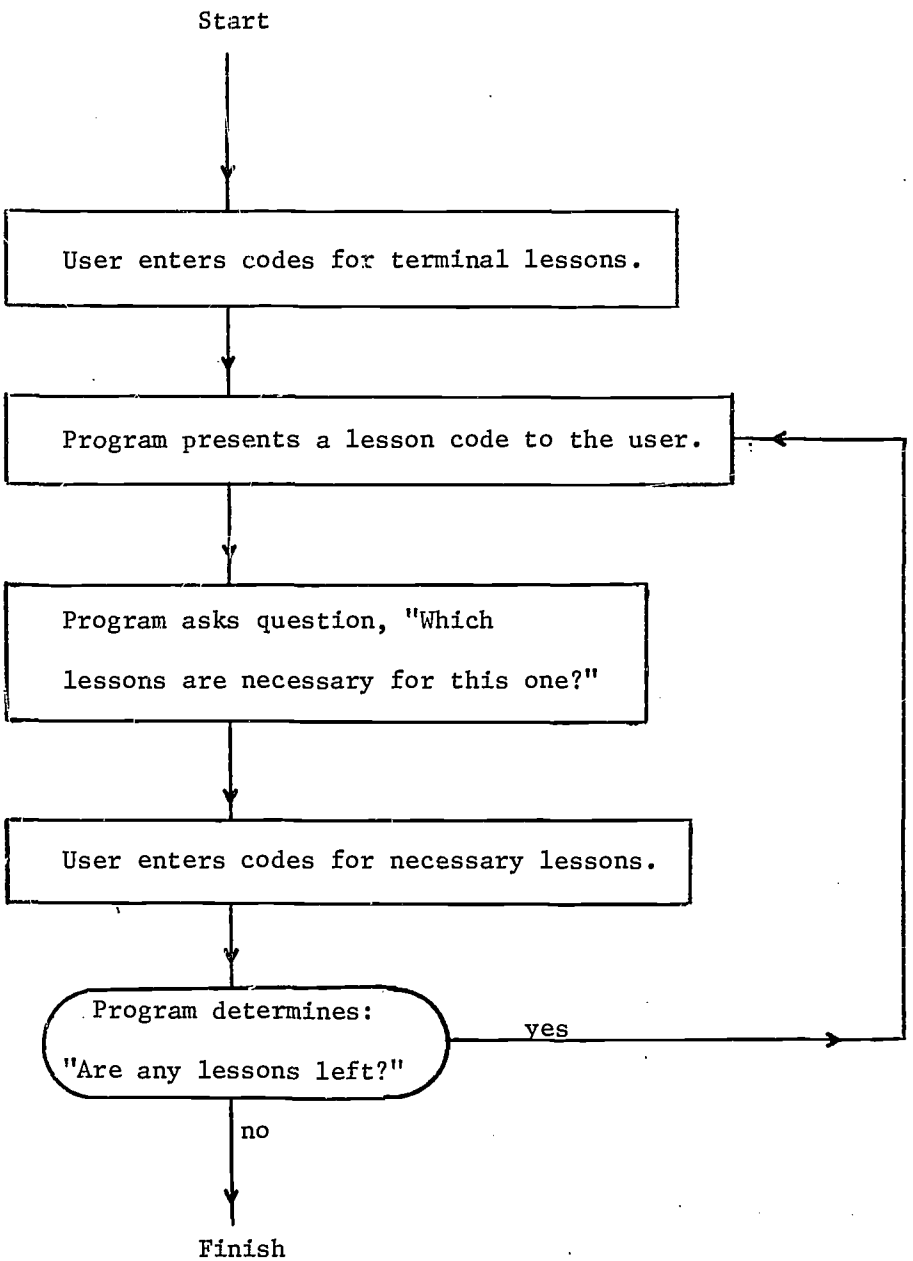


Figure 3. Specification of the Relations

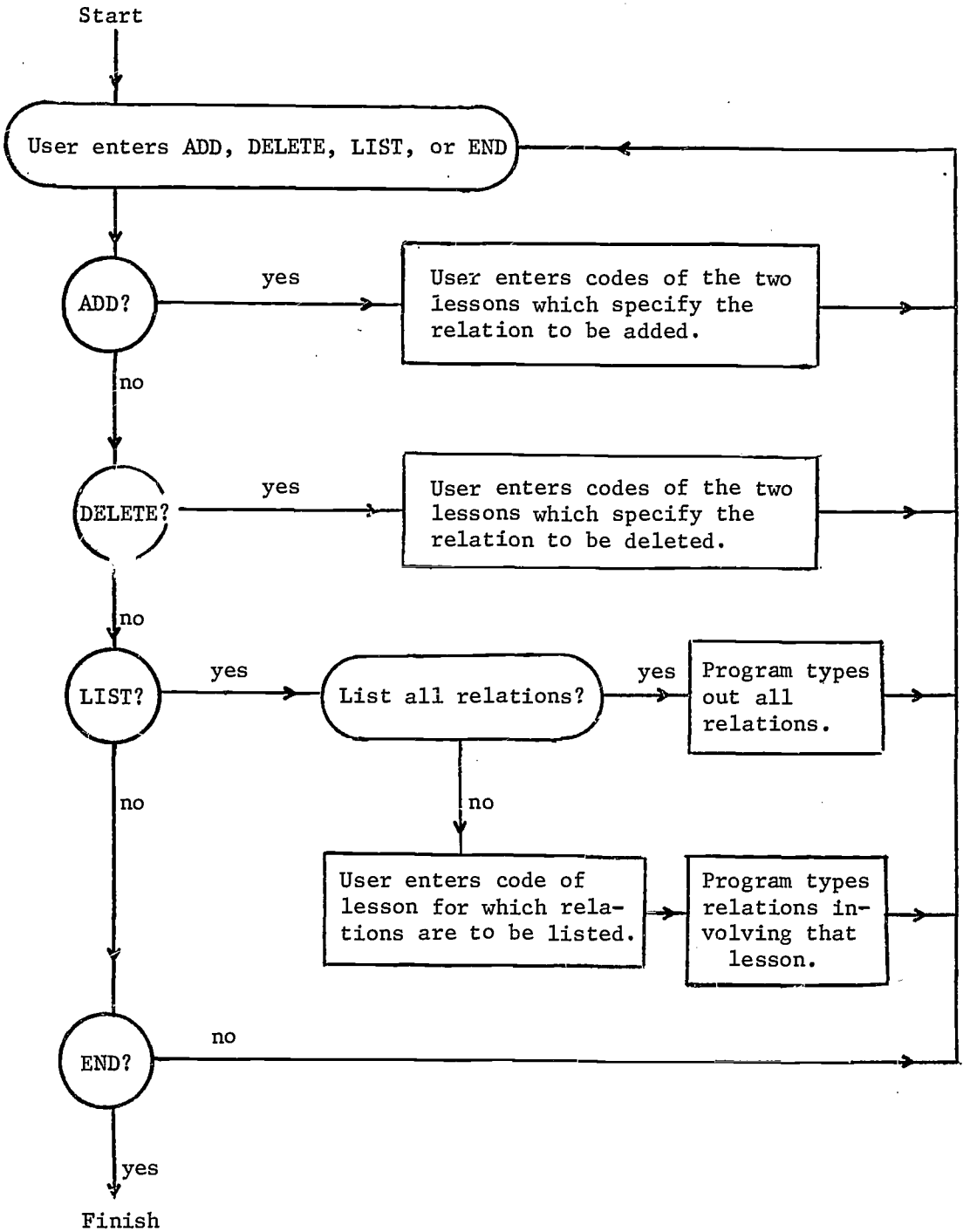


Figure 4. Changes to the Relations

which the management program is based. The first step in this analysis is to generate a level structure, a procedure which is possible for any acyclic digraph. If the digraph is not acyclic, i.e., it contains a cycle, then it is not possible to generate a unique level structure. In this case, the program branches to a routine which identifies the cycle and returns the user to the third part of the program where he may delete one or more of the relations in the offending cycle. After a level structure is successfully generated, it is typed for the user so that he may use the information if he cares to make a diagram of the structure which he has specified. The lessons that were assigned to the lowest level become entry points to the curriculum when it is served by the management program. After the levels are typed, further analysis is undertaken to simplify the resulting digraph model. Since all the relations specified are assumed to be transitive, any explicitly stated transitive relations are unnecessary and are then removed (see page 13). An example of a redundant transitive relation is the case in which a relation was explicitly specified for a lesson that was only indirectly necessary for the mastery of another. Finally, all relations remaining in the acyclic digraph model are typed out by level to complete the analysis. The structure data, consisting of lesson codes, their order within the level structure, and the relations connecting them, are saved as a dataset named STRUCTURE:PIL. The steps taken to form the digraph model in the fourth part of the program are summarized in the flow chart in figure 5. Details of the analysis of part four of the program are given in the form of

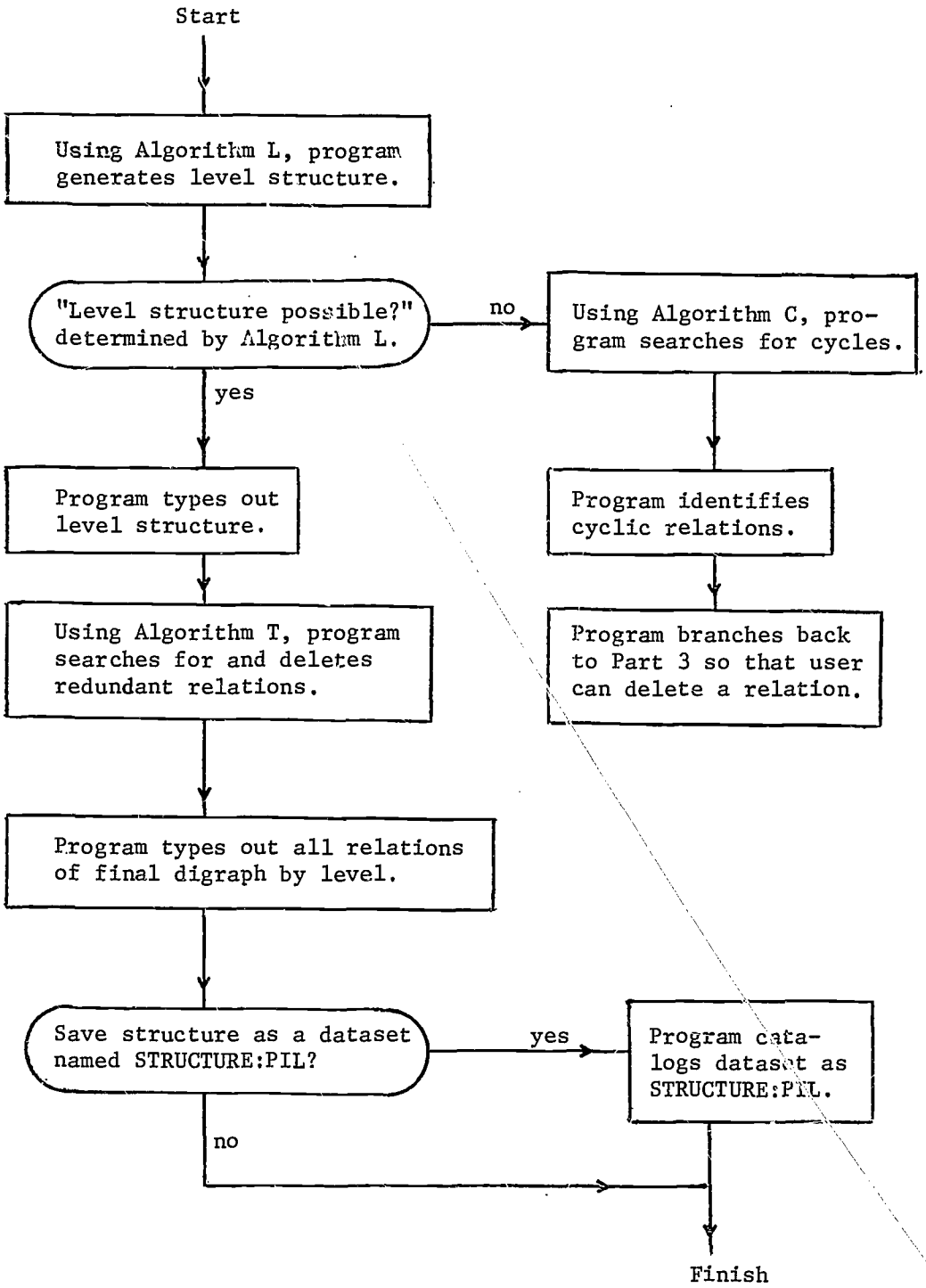


Figure 5. Generation of the Final Digraph

algorithms for each major step of the analysis. The conventions used for describing the algorithms are those used by Knuth (1968, pp. 2-4).

The first task in the analysis is to generate a level structure. Defining a (directed) path as a series of connected lines going in the same direction, it can be shown that for an acyclic digraph, if n is the length of the longest path, then $n + 1$ is the smallest number of levels in a level assignment of the digraph. This follows from the consideration that for a given path, the levels of all points in that path must be distinct. A path of length n must contain $n + 1$ points if the digraph is acyclic; thus $n + 1$ levels are necessary (Harary et al., 1965, pp. 269-270). The algorithm used to generate the level structure follows:

Algorithm L. (Level structure). Given the points on a digraph D and the lines joining them, generate an ascending level structure using the smallest possible number of levels.

L1. [Initialize] Start a new level.

L2. [Next point] Choose the next point on D.

L3. [Transmitter?] Determine the number of lines directed into the point. If zero, enter the point in the current level.

L4. [More points?] If all points have not been chosen, go to L2.

L5. [Level empty?] If the current level has no entries, the algorithm terminates: the digraph was not acyclic.

L6. [Reduce digraph] Remove those lines directed out from each of the points in the current level. Remove these points. If no points remain, the algorithm terminates; otherwise the remaining points and lines make up D. Go to L1.

In the event that it is not possible to generate a level structure, a search for the cyclic relations is undertaken. The first step in this search is a reachability calculation. Point v is reachable from point u in a digraph if and only if there is a sequence (or path) from u to v. After reachability has been calculated for all pairs of points, a cycle can easily be identified from the points which are mutually reachable, i.e., reachable from each other. For purposes of calculation, a digraph can be represented in matrix form. The points and lines of the digraph are represented by an adjacency matrix in which the rows and columns correspond to points on the digraph, and the matrix entry a_{ij} is non-zero only if there is a line from v_i to v_j in the digraph. In the reachability matrix the presence of a non-zero element r_{ij} indicates that a point v_j is reachable from a point v_i (Harary et al., 1965, p. 110). Since a point is defined to be reachable from itself, the diagonal entries on the reachability matrix are unity. The algorithm for finding the cyclic relations follows.

Algorithm C. (Cycle identification). Given n points on a digraph D and the lines joining them, identify the cycles by determining which points are mutually reachable.

C1. [Initialize] Set $r_{ij} \leftarrow a_{ij}$, $r_{ii} \leftarrow 1$, $I \leftarrow 1$, $J \leftarrow 1$. (Set up matrix from adjacency matrix A, set diagonals to 1, initialize indices)

C2. [New row?] If $J > n$, set $J \leftarrow 1$, $I \leftarrow 1$.

C3. [Last row?] If $I > n$, set $I \leftarrow 1$, $J \leftarrow 1$. Go to C8.

C4. [Element non-zero?] If $r[I,J] \neq 0$, set $J \leftarrow J + 1$. Go to C2.

C5. [Product non-zero?] Take product of corresponding elements of row I, column J. If one is non-zero, go to C7.

C6. [Next element] Set J ← J + 1. Go to C2.

C7. [J reachable from I] Set r[I,J] ← 1, J ← J + 1. Set flag. Go to C2.

C8. [Is reachability complete?] If flag is set, initialize it and go back to C2.

C9. [More elements?] If J > I, set J ← 1, I ← I + 1.

C10. [More rows?] If I > n, the algorithm terminates.

C11. [Transpose elements unequal?] If r[I,J] ≠ r[J,I], set r[I,J] ← 0, r[J,I] ← 0.

C12. [Next element] Set J ← J + 1. Go to C9.

The points on the digraph which are mutually reachable, and thus make up the cycle, are those which correspond to any row with at least one non-zero off-diagonal element in the resulting r matrix of Algorithm C. The user is then branched to the section of the program where he may delete one or more relations.

Following the successful completion of the level structure, the search for explicitly stated transitive relations is undertaken. This also involves a reachability calculation, but it can be shortened somewhat by first rearranging the matrix representation of the digraph in upper triangular form, a form that is always possible when representing an acyclic digraph (Harary et al., 1965, pp. 268-269). The upper triangular form is easily obtained by ordering the points corresponding to the rows and columns of the matrix in the same order as that which resulted from the generation of the level structure. The algorithm for identifying and removing explicitly stated transitive relations follows.

Algorithm T. (Transitive relation identification). Given n points on an acyclic digraph D represented by the upper-triangular adjacency matrix A, identify and remove those lines representing transitive relations.

T1. [Initialize] A_{ij} is upper triangular. Set I ← 1, J ← I + 2

T2. [New row?] If J > n, set I ← I + 1, J ← I + 2

T3. [Last row?] If I > n, set I ← 1, J ← I = 2. Go to T9.

T4. [Element negative?] If a[I,J] < 0, set J ← J + 1. Go to T2.

T5. [Product non-zero?] Take product of corresponding upper-triangular elements of row I, column J. If one is non-zero, go to T7.

T6. [Next element] Set J ← J + 1. Go to T2.

T7. [Element positive?] If a[I,J] > 0, type out message that the line from point I to J was redundant. (Point J was reachable from point I by some combination of lines; its explicit specification was unnecessary.)

T8. [J reachable from I] Set a[I,J] ← -1, J ← J + 1. Set flag. Go to T2.

T9. [Is reachability complete?] If flag is set, initialize it and go back to T2. Otherwise the algorithm terminates.

This algorithm calculates reachability and superimposes elements representing paths of length two or more over the adjacency matrix by assigning them values of -1. If the calculation yields a path of length two or more between two points for which a line is present on the digraph, that line is transitive and therefore redundant. Its corresponding element is changed on the matrix from +1 to -1, and when the algorithm terminates the positive elements represent the final adjacency matrix.

Finally, the relations are typed out in the upper-triangular order, and the final structure is saved as a dataset named STRUCTURE:PIL

at the user's option. The user may then draw a graphical representation of the digraph that has resulted from the use of the program.

C. The Mastery Data Base

There were two considerations in the development of the data base for the management program: what information should be saved and in what form should that information be stored? Both of these considerations depended upon the function of the management program. Since one objective was to keep the system as simple as possible to use and maintain, only the information essential to the functioning of the management program was included in the data base. The information concerning the curriculum, including the lesson identification data and the digraph representation of the structure, was saved as datasets named LESSON:PIL and STRUCTURE:PIL. The remaining information consisted of the mastery data of the individual students involved in the curriculum. This data base contained the following information: (1) the student number, (2) the name of the student, (3) the lesson identification code for as many points of the curriculum as were mastered, (4) the date on which mastery was certified for each, and (5) further information regarding the conditions of mastery.

The student mastery data base was set up in the form of a linked list, an information structure in which each element of a list points to other related elements, in order to facilitate accessibility to the mastery data by the management program. The linked list improves accessibility over a sequential format by allowing a search procedure to examine relatively fewer entries of

the list to find the correct entry. The entries of the data base each contained a left link, pointing to the next-entered lower-ordered entry, and a right link, pointing to the next-entered high-ordered entry. The ordering was based on the student number of each entry. The data base is started when the first mastery record is added. As each new record of mastery is added to the data file, it is simply attached to the end of the file, and its order within the file is represented by adding a new link to one of the already-existing records of the file. This type of ordering scheme, in which new records are added to the file in any order, is due to the efforts of Booth and Colin (1960), and is used to best advantage when the entries are randomly ordered. The algorithm used to add a new record of mastery data to the file follows.

Algorithm A. (Add mastery records). Given the student number NUM of the student for whom the mastery data are to be added and the number of mastery records NR currently on the file, add mastery records at the end of the file and specify their sequence within the file by altering appropriate links of existing records.

A1. [Read record] Read next record from the mastery file. Set L ← its left link, R ← its right link, and S ← its student number.

A2. [Mismatch?] If NUM ≠ S, go to A5.

A3. [Name?] If the variable NAME is defined, go to A5.

A4. [Define name] Set NAME ← name field of last record read.

A5. [Compare numbers] If NUM < S, go to A8.

A6. [Right link positive?] If R > 0, position mastery file at record R. Go to A1.

A7. [Link to last record] Set R ← NR + 1. Go to A10.

A8. [Left link positive?] If L > 0, position mastery file at record L. Go to A1.

A9. [Link to last record] Set L ← NR + 1.

A10. [Rewrite] Position mastery file back one record. Rewrite last record onto mastery file with new values of L and R. Position mastery file after last record. Set L ← 0.

A11. [Name?] If the variable NAME is defined, go to A13.

A12. [Get NAME] Read NAME from user terminal. (New Student)

A13. [Get mastery data] Read mastery data from user terminal.

A14. [One record?] If mastery data will all fit onto one new record, set R ← 0; otherwise set R ← NR + 2.

A15. [Write record] Write new mastery record onto mastery file with values of L, R, NUM and NAME. Set NR ← NR + 1.

A16. [More data?] If there are additional mastery data which did not fit onto the last record, go to A14. Otherwise the algorithm terminates.

In order to determine which lessons have been mastered by a particular student, a search routine similar to the routine to add a record was employed. The algorithm for this search follows.

Algorithm S. (Search mastery file). Given the student number NUM of the student for whom the mastery record is searched and the number of mastery records NR currently of the file, search the file for all records containing information for that student.

S1. [Read record] Read next record from the mastery file. Set L ← its left link, R ← its right link, and S ← its student number.

S2. [Mismatch?] If NUM ≠ S, go to S6.

S3. [Name?] If the variable NAME is defined, go to S5.

S4. [Define NAME] Set NAME ← name field of last record read.

S5. [Mastery data] Store mastery information from last record read.

S6. [Compare numbers] If NUM < S, go to S8.

S7. [Right link positive?] If R > 0, position mastery file at record R. Go to S1. Otherwise, go to S9.

S8. [Left link positive?] If L > 0, position mastery data file at record L. Go to S1.

S9. [Name?] If the variable NAME is defined, the stored mastery information is complete; the algorithm terminates. Otherwise, the input value NUM has no entries on the mastery file; branch to error message; the algorithm terminates.

If the mastery data file had been organized in the form of a sequential list instead of a linked list, a search routine would have to read every record in the mastery file, compare its number field with the number searched for, and, if they matched, store the data. Algorithm S requires that many fewer records be read and compared, and increases the efficiency over the sequential search as the number of records in the file increases.

D. Utility Programs to Maintain the Data Base

To start and maintain the mastery data base so that it was current for any student at the time he wished to use the system, it was necessary to add the mastery data to it in the instructional setting. Two functions were deemed necessary: the ability to add mastery data to the data base and the ability to see what has been entered for a given student. To achieve this, two interactive programs were implemented. They were both designed from the viewpoint of being easy to use in the instructional setting and occupying the smallest possible computer storage. The programming language used for each was the interpretive language PIL (University of

Pittsburgh Computer Center, 1969) that is supported by the Time-Sharing System at the University of Pittsburgh. The interactive processor CATALYST/PIL was not used because the amount of interaction required by these programs was small, whereas the functions they performed were done more efficiently by the interpretive language alone.

The program to input mastery data was designed to be used in or near the classroom with a remote terminal. Mastery data consisting of the student's name, his student number, the lessons mastered, and their respective dates of mastery are entered from a data sheet prepared by a classroom aide. From this sheet the user in the instructional setting can enter the mastery data by means of the input program named INPUT:PIL. The input program, the first time it is used, creates the mastery file, which is then saved as a dataset named MASTERY:PIL. Subsequent uses of the program add data to this mastery file.

The input program first stores the lesson codes from the structure data file so that a typing error by the user does not result in erroneous mastery data. The user may also specify other lesson codes that he may input to the mastery file which would not otherwise be accepted. Then the user enters the student number of each student for whom mastery data are to be entered, along with the data for the lessons mastered. Algorithm A is carried out, which adds the appropriate mastery data to the mastery file. The lesson codes entered must correspond to one of the

codes from the structure data file unless exceptions were specified. The dates should be five-digit integers. A sixth character may be suffixed to the data entered as a condition on the mastery. For instance, if desired, a sixth character can indicate that a lesson should be considered mastered from the point of view of the student's progress within the curriculum, but that the lesson was not, in fact, mastered according to criteria set by the curriculum designer. The program to input mastery data is represented by the flow chart in figure 6.

The program to retrieve data for a given student or for the entire student mastery file was also designed for use in the classroom with a remote terminal. If the student number for a single student is input, the program searches for and types out the lesson codes mastered with their dates of mastery. Or, if a complete listing is wanted, mastery information for the entire file is typed. This program, named LIST:PIL, uses only the mastery data from MASTERY:PIL. To find the data for a single student, the search procedure of Algorithm S is used. To output the entire file, a different procedure is used, which is described in the algorithm which follows.

Algorithm P. (Postorder traversal of mastery file). Given the mastery file formed using Algorithm A and the use of an auxiliary stack, traverse the file outputting mastery data for all students in order by their student number.

P1. [Initialize] Set J ← 1, set up empty stack.

P2. [J zero?] If J = 0, go to P5.

INPUT:PIL

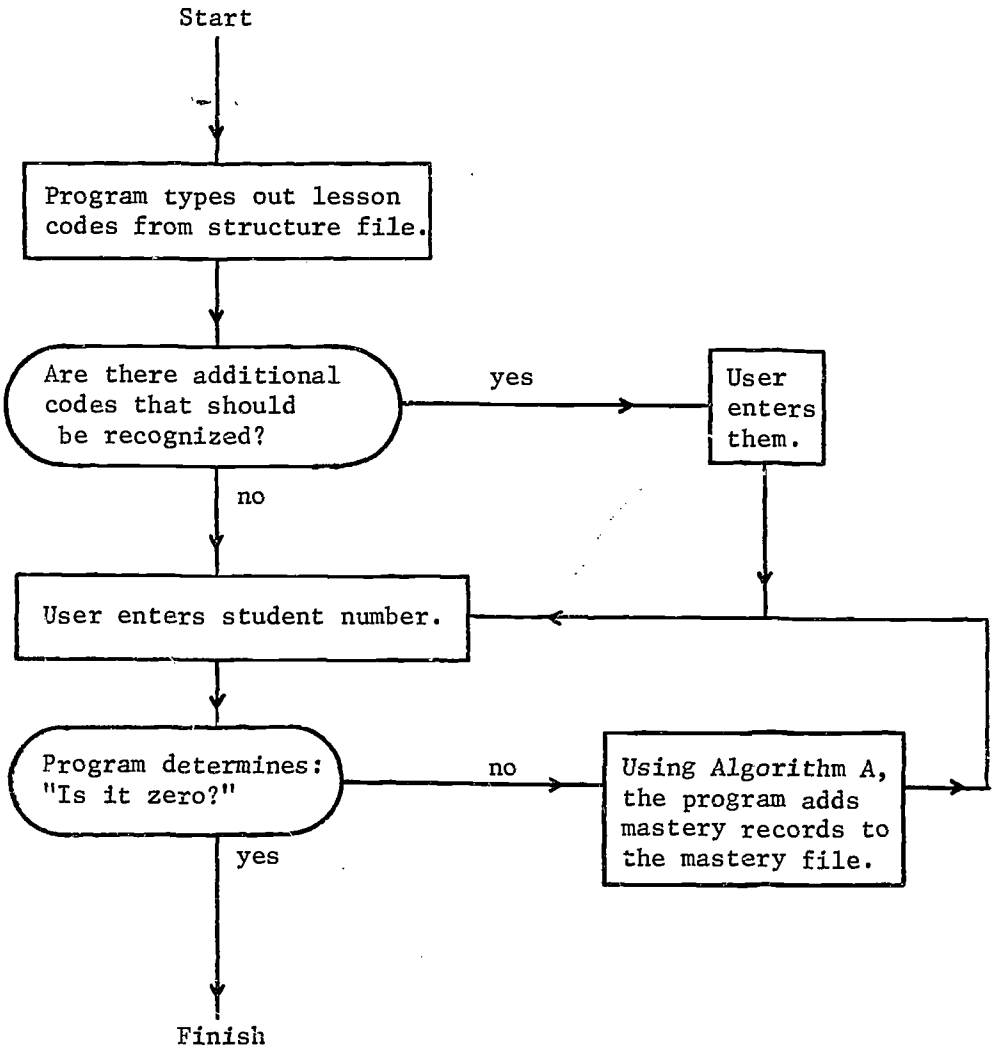


Figure 6. Program to Input Mastery Data

P3. [Stack ⇐ J] Push the value of J onto the stack.

P4. [Read record] Position mastery file forward to record J. Read that record. Set J ← its left link. Go to P2.

P5. [Stack empty?] If stack is empty, go to P13.

P6. [J ⇐ Stack] Pop a value from the stack; set it equal to J.

P7. [Read record] Position mastery file back to record J. Read that record. Set J ← its right link.

P8. [Next student?] If the record read is the first record of another student, go to P10.

P9. [Mastery data] Store mastery information from last record read. Go to P2.

P10. [First student?] If this is the first time through, go to P12. (No stored information yet.)

P11. [Output data] Type out the mastery information which was stored from previous student: NAME, NUM, lesson codes and dates.

P12. [Next student] Set NAME ← name field of last record. Set NUM ← its student number. Go to P9.

P13. [Last student] Type out the mastery information which was stored from last student. The algorithm terminates.

An outline of the program to type out the mastery data is represented by the flow chart in figure 7.

E. The Program to Provide Options

The processor to specify curriculum structure generates two output files, the lesson identification file (LESSON:PIL) and the structure file (STRUCTURE:PIL). The program to input mastery data generates a mastery data file (MASTERY:PIL). The program which was developed to provide options to the student uses all three of these files as input data. The option program, named OPTION:PIL and

LIST;PIL

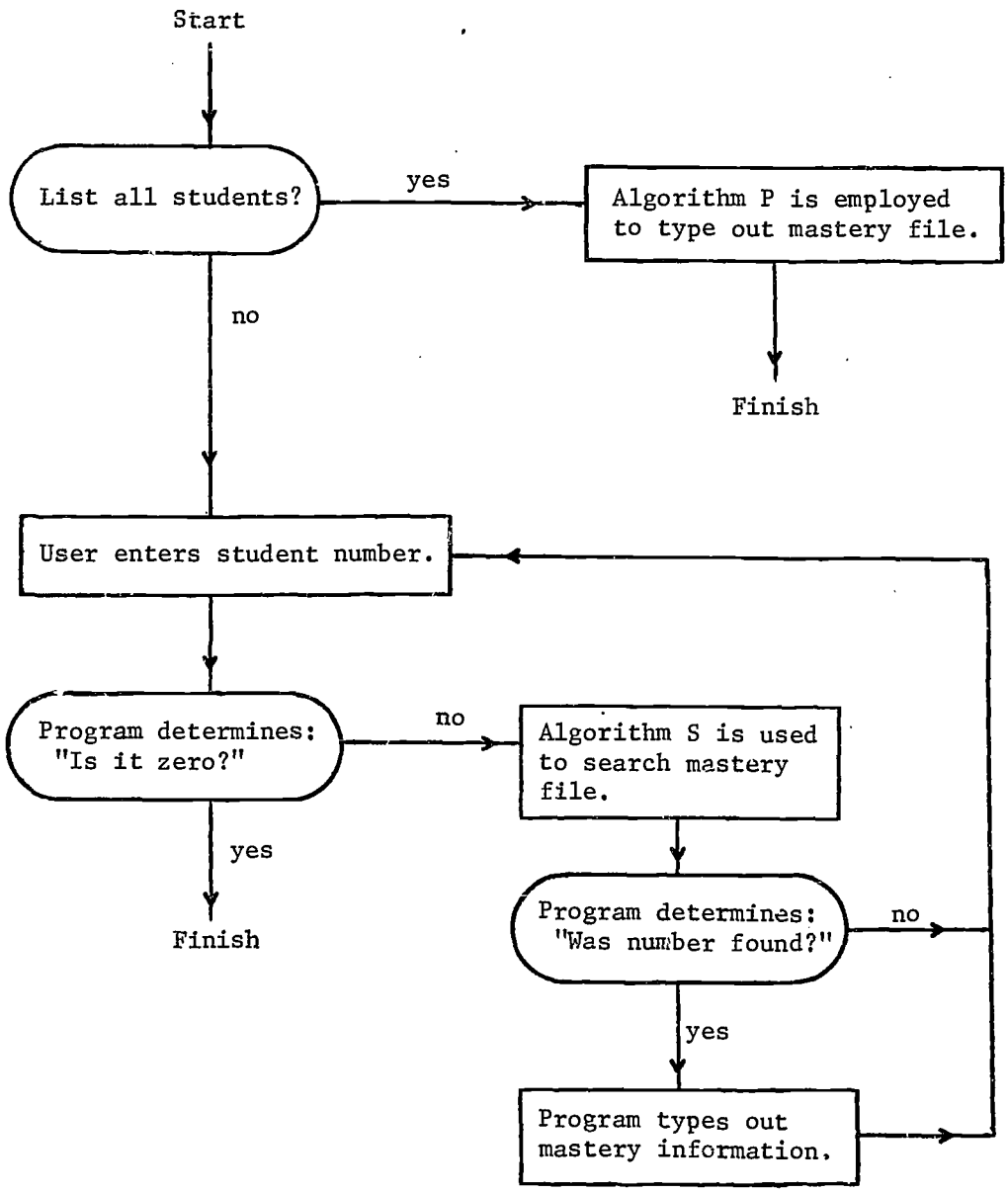


Figure 7. Program to List Mastery Data

written in the interpretive language, is the one program in the management system that the student can use himself. It is an interactive program designed to be run on a remote terminal in or near the classroom. Output from the option program consists of a list of the lessons in the curriculum that the student, by virtue of his previous mastery of lessons, may select. The choice of the lesson, in fact, is not recorded at the terminal, since the lesson chosen may depend on the student's later investigation of the lesson materials, the availability of those materials, and the judgment of the teacher. The option program was designed to be used by the student or teacher as a classroom aid, was not to be restrictive or binding on the progress of the student, and was to be, in fact, strictly optional itself.

The option program determines which lessons from the curriculum have already been mastered by the student in question and superimposes this information over the digraph model of curriculum structure in order to determine the lessons to which the student may proceed. Then it outputs each lesson choice along with the description of the student's information. It accepts any combination of lessons mastered regardless of the curriculum structure at the time the program is used. Without this feature, the management program could not adapt to variations in the curriculum or its structure. It was designed to be simple to use, require minimal input information via the terminal, and yield short and concise output information.

The option program first accesses the dataset STRUCTURE:PIL and stores the structural information regarding the lessons. It then requires only that the student using it enter his student identification number. The use of an identification number avoids the possibility of spelling errors or the need to store abbreviated forms of names that the student must remember. The search routine (Algorithm S) is employed to gather information about the student from the dataset MASTERY:PIL. When his records are found, he is presented with the name corresponding to the identification number he has typed. A "yes" or "no" response is all that is required for verification of the name. Next, the lessons that have been mastered previously are removed from the structure model of the curriculum along with all relations involving them. The examination of the resulting structure made up of the remaining lessons determines which of them require no prerequisites. The appropriate lesson codes are then paired with the lesson descriptions provided on the dataset LESSON:PIL, and this information is presented to the student at the terminal. If the list of options contains one or more lessons that have been mastered or that the teacher feels can be skipped or are inappropriate, the lessons in question can be removed from the list of choices, and a new list of options is typed out. After an appropriate list of options is typed, the program is complete, after which another student may use the remote terminal. Since the system is completely time-shared, as many students may use the program simultaneously as there are terminals available, up to the limitations of the system.

The algorithm used in the option program to modify the digraph in terms of the lessons mastered follows. Its function is to remove the points that correspond to lessons mastered from the upper-triangular matrix representation of the digraph model of structure, and to output those remaining that require no prerequisites.

Algorithm R. (Remove mastered points). Given n points on acyclic digraph D represented by the upper-triangular adjacency matrix A, remove a subset of the n points, corresponding to lessons mastered, and output those remaining with no prerequisites.

R1. [Initialize] Let I be row counter, J be column counter. Set $I \leftarrow 1$, $J \leftarrow 1$.

R2. [Check rows] If lesson of row I is not mastered, go to R4.

R3. [Mark lines] Mark all lines coming out from point I for removal. (All non-zero entries in row I.)

R4. [Next row] Set $I \leftarrow I + 1$. If $I \leq n$, go to R2.

R5. [Check columns] If lesson of column J is not mastered, go to R7.

R6. [Add lines] Add a line to digraph D from points going into point J (i.e., entries in column J that were not marked for removal in R3) to all points coming out from point J (i.e., entries in row J including any marked for removal).

R7. [Next column] Set $J \leftarrow J + 1$. If $J \leq n$, go to R5. Otherwise, set $I \leftarrow 1$.

R8. [Check rows] If lesson of row I is mastered, go to R10.

R9. [Store option] If point I has no lines coming out from it (i.e., no entries in row I), store it as an option. Output the description corresponding to that option.

R10. [Next row] Set $I \leftarrow I + 1$. If $I \leq n$, go to R8. Otherwise the algorithm terminates.

A description of the entire option program is detailed in the flow chart of that program given in figure 8.

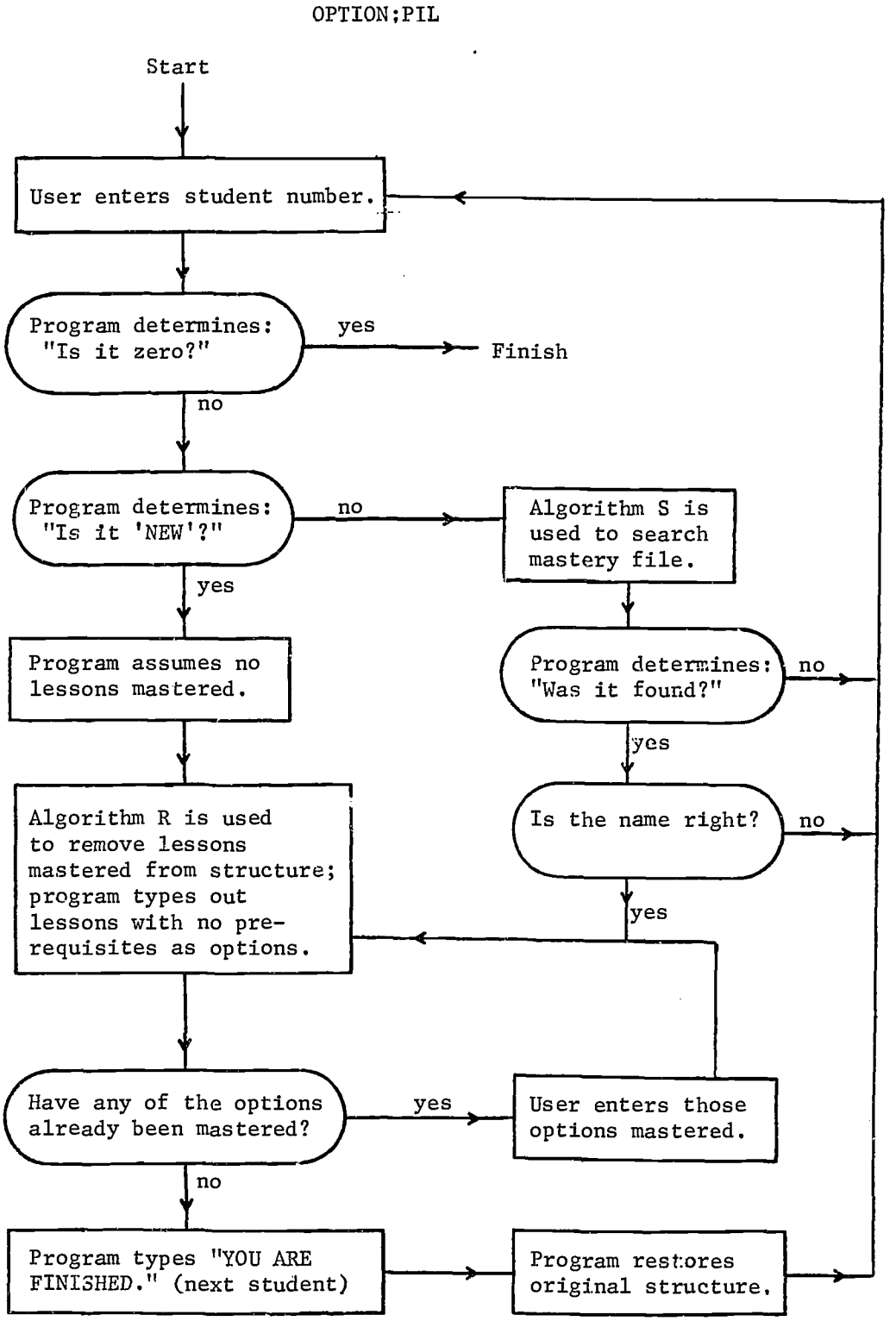


Figure 8. Program to Provide Options

F. Feedback for the Curriculum Designer

The problem of what feedback information to provide for the curriculum designer depends upon how he plans to use the management program. Since the mastery data for all the students are stored on the mastery file (MASTERY:PIL) whether or not the option program is used to assign their lessons, these data are available for an after-the-fact examination of the structure which was initially assumed. Whether that examination should take the form of an analysis or just act as an indicator depends upon the criteria used in specifying the structure. It is unrealistic to assume that the structure specified will be the result of detailed psychological transfer studies or subjected to scalogram analysis for most of the uses that are to be made of the management program. In addition, the management program was not designed to be used in a controlled study, but as a free-wheeling classroom aid. The structure specified need not consist entirely of instructional materials. As an example, certain non-instructional "lesson" points can be included in the structure as common entry and exit points to different parts of the curriculum, or intermediate tests can be included as points to be "mastered."

For the mastery data that are stored as a result of using the management program, several interesting indicators might be of value to the curriculum designer. First the percentage of mastery for each lesson at any point in time might be revealing. If this summary revealed that a certain lesson was mastered by fewer students than lessons of corresponding difficulty, it could indicate trouble

with the lesson, with the process used to certify mastery, with the desirability of the lesson to the pupil, or with the availability of the lesson in the school. The cause may or may not be obvious depending on the situation, but the phenomenon could otherwise go unnoticed in the extra-instructional setting. Next, if one looks at the pairs of lessons for which a structural relation was said to exist, each relation implies an order of mastery. If the order of mastery for all such pairs across all students is examined, violations of the implied order of mastery might indicate that certain relations specified should be re-examined. The date of mastery was included in the student mastery data file so that the order of mastery could be retained as part of this summary. Violations of the order of mastery implied by a relation between two lessons include the second lesson's mastery before the first, and the second lesson's mastery in the absence of the first. Finally, if the first lesson in a relation is mastered, a look at whether its mastery is followed up by mastery of the second might be revealing in some cases. Given the mastery of the first lesson, the number of cases in which the second lesson is later mastered could indicate that the jump between mastering the two lessons is too severe, or that an intermediate lesson might be advantageous if a very low transfer rate exists.

The summary program was designed to provide the information which might prove useful for the curriculum designer in improving his curriculum. The program, named SUMMARY:PIL, uses the datasets STRUCTURE:PIL and MASTERY:PIL. It carries out a postorder search

(Algorithm P) of the mastery file to record the following information: (1) the number of students mastering each lesson; and, for each relation, (2) the number of students mastering the first lesson before the second one, (3) the second lesson before the first one, (4) the first lesson only, and (5) the second lesson only. This information is provided only for the lessons and relations specified in the structure dataset. In case there are points of the structure that should not be included in a mastery summary, the user can cause these points to be deleted from the analysis by entering their codes as points to be ignored. In case there were characters used in specifying mastery that should not be interpreted as such for purposes of the summary, these characters can be entered as symbols for non-mastery. A mastery summary is typed out in a grid format, in which rows are students and columns are lessons. An outline of the summary program appears in the flow chart in figure 9.

G. The Management Program

In this section the components of the management program have been described in detail. Taken together, they form the computer-based management program which was developed for use with adaptive instructional systems. A schematic representation of the management program is presented in the diagram in figure 10. The curriculum designer specifies the curriculum structure using the MODEL program. The classroom teacher or aide builds up the student mastery data base by using the utility program to input mastery data, INPUT:PIL. Either one can use the utility program LIST:PIL

SUMMARY:PIL

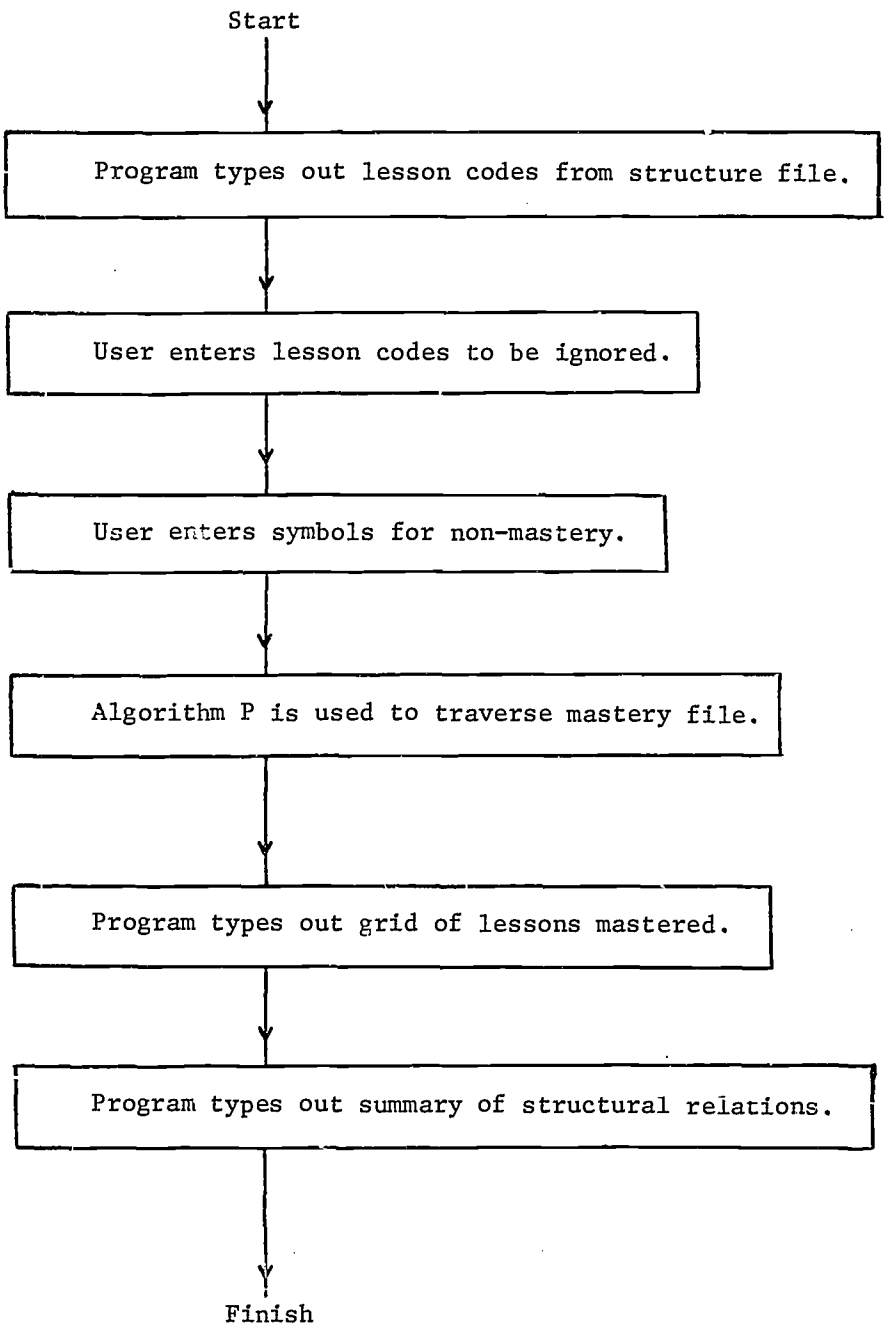


Figure 9. Mastery Data Summary Program

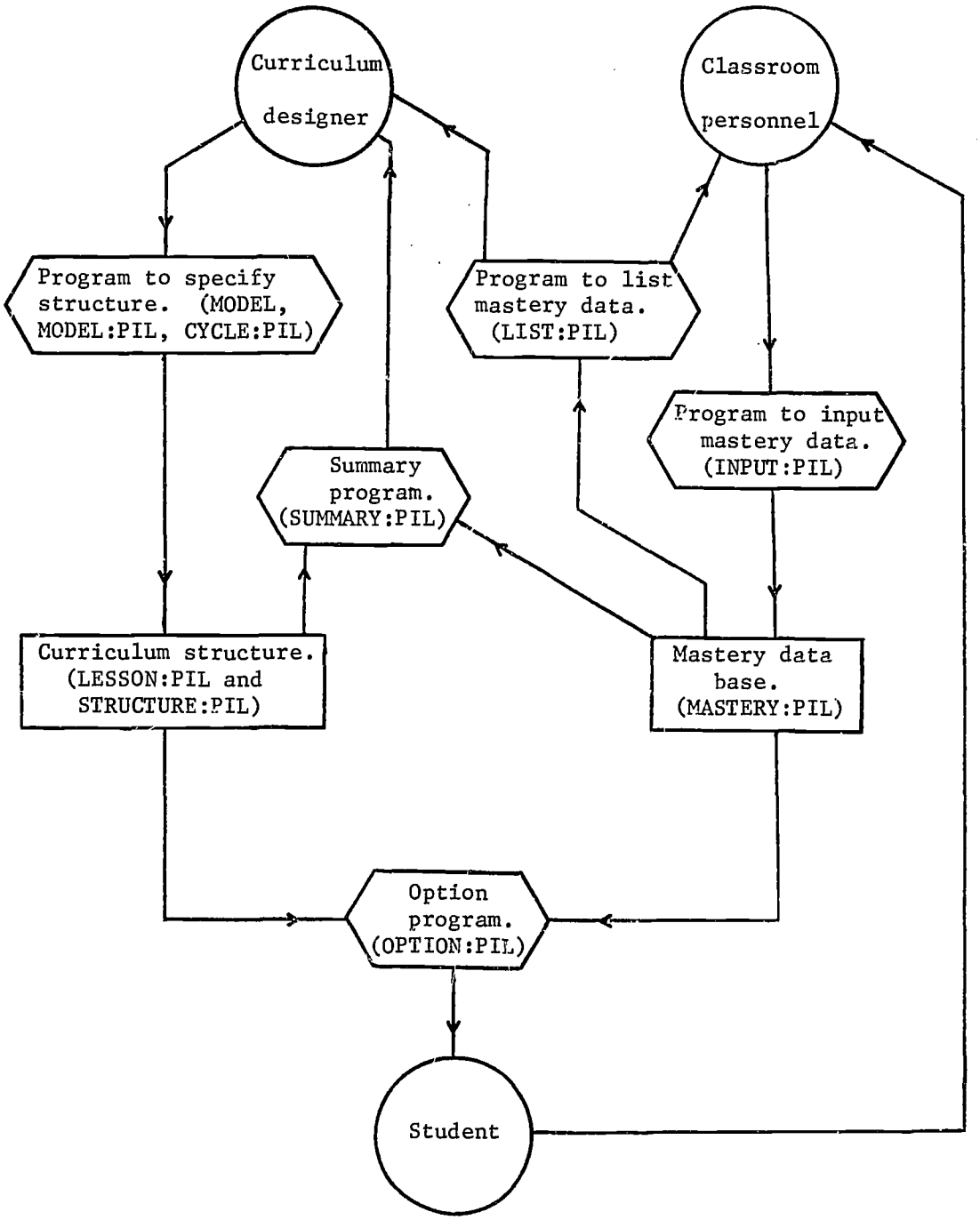


Figure 10. Schematic Representation of the Management Program

designed to list the mastery data file in order to examine this file. The mastery data file, MASTERY:PIL, and the structure file, STRUCTURE:PIL, are used as inputs for the program to provide options, OPTION:PIL, with which the student interacts. These files are also used as input to the summary program, SUMMARY:PIL, which feeds back information to the curriculum designer about the success of his curriculum. After the student chooses his instructional materials and masters them, this information is compiled by the classroom personnel and entered onto the mastery data file in preparation for the next time the OPTION:PIL program is used.

III. RESULTS OF THE FIELD TEST

A. Choice of IPI Science

After the management program had been developed, the next step was to apply it to an existing instructional setting. The IPI Science program being used at Oakleaf School was chosen to demonstrate the management program because its characteristics were such that the management program could be used to advantage. First, the Science curriculum contained a large amount of material from differing content areas, so that allowing the student a choice of instructional sequence had relevance. Second, the IPI Science program was not a stable curriculum, and its hierarchies were not well established. The flexibility and adaptability of the management program can be used to best advantage when applied to a curriculum that is subject to periodic revision and normal evolution.

The IPI Science program consists of several "levels" which correspond roughly to grade levels in the elementary school. The goal of the IPI Science program is to provide instruction within the IPI framework for kindergarten through grade nine. Instruction in effect at Oakleaf School began in grade one with level A Science. Level B Science normally began in grade two, but students might enter it in grade one after completing level A. Before beginning instruction in a certain level, a student was required to take a placement test for required entering behaviors and prior mastery of material to be presented within that level. Within each lesson

in the level was a "curriculum-embedded test" which tested the behavioral objective of that lesson and served to certify its mastery.

In practice, level A Science lessons were grouped into units representing a set of related objectives. Each time a student mastered all lessons in a unit he went to the teacher for a new "prescription," in most cases a new unit to study. The dominant mode of instruction in level A Science had been the taped lesson, consisting of a tape cassette and a box of manipulative materials, by which the student received instructions through a set of earphones connected to a playback device. There was a summary chart for each student entitled "Student Order of Program," which contained blocks for all the units of level A which were crossed out as each unit was mastered. A copy of this chart is shown in figure 11. Within each unit a prescription sheet was used to record the progress of each student in mastering the lessons. In addition, there were other instructional activities not required for the mastery of any unit, including group activities and individual student projects. To implement the management program, the level A Science units contained on the chart from which prescriptions were given (figure 11) were selected as the curriculum points. Level A included three placement tests which tested for prior mastery of the units. These tests were also considered to be curriculum points, as was an introductory unit on the use of the tape players. Further information concerning the rationale, development, and operations of IPI Science can be found in Lipson (1966), Klopfer and Weber (1969), and Learning Research and Development Center (1970).

IPI SCIENCE

STUDENT ORDER OF PROGRAM

NAME _____ ROOM _____ GRADE _____

A LEVEL

COLOR 1 2	SIZE/SPACE 1 2	SOUND 1 2	SMELLS 1 2
IRON PULLERS 1 2 3	SORTING THINGS 1 2	INCHING ALONG 1 2 3 4	SHAPES 1 2 3
FILLING THINGS 1 2 3	NEW SORTS 1 2	SINK OR SWIM	TIME 1 2 3
MEASURING TEMPERATURE 1 2 3 PT. I PT. II	FREEPLE GAME	(ALTERNATE UNIT) HOT AND COLD LIGHT 1 2	MEASURING TEMPERATURE EXPLORING UNIT 1 2 3 4

Figure 11. Student Order of Program

B. Specification of Structure

Specifying the structure to be used with the management program at the Oakleaf School involved deciding what the points of that structure were to be and determining how they were related. The units listed on the chart entitled "Student Order of Program" (figure 11) are a logical choice of the points for the curriculum structure. The points of the structure correspond to decision points in the management of instruction, since prescriptions are written by the teacher upon completion of each unit listed on the chart. In addition, three placement tests, called "mini-placement tests," were included in the structure, since a placement test must be taken before the units which it tests can be prescribed. With the initial unit, which teaches the use of the tape machines, there was a total of twenty points on the curriculum structure. To these twenty points, or "lessons," were assigned lesson codes by which they would be referenced in the various programs. After this task was completed, the "description" of each was written; i.e., the information which was to be presented to the student in choosing his option. Because of the young age of the students who were to use the program and their limited vocabulary, the temptation to describe the lessons in terms of behavioral objectives was resisted. Instead, each lesson was described by asking a rhetorical question which might generate interest in its content. At this point the program to specify curriculum structure, MODEL, was run and the lesson information that was entered was stored as a dataset named LESSON:PIL. The program was stopped short of the

point of specifying structure, however. The points of the curriculum, including their codes, names, and descriptions, that made up the lesson list that was used for the program are given in figure 12.

After the lesson list was completed, the structural relations of that group of lessons were specified by using the program to specify structure, MODEL. A person that had been instrumental in developing the level A IPI Science curriculum and responsible for supervising its management at Oakleaf School at the beginning of the school year interacted with the MODEL program. Within approximately one and one-half hours the specification of the structure was completed. This person was completely unfamiliar with either the remote terminal or the workings of the program prior to this time. The author was present to answer questions regarding the use of the terminal. The structure was altered slightly after the program had been put to use in the school because of procedures which the teacher used in prescribing three lessons. Two of these lessons were omitted from the structure because they were never prescribed in practice, and one lesson was moved to a higher level to allow for a prerequisite lesson. In this case, the author, thoroughly familiar with the program, was able to completely specify the structure in about one-half hour. The final form of the structure of the full set of twenty lessons is described by the hierarchy chart in figure 13, drawn from the structure as typed out by the MODEL program.

LESSON :PIL 13 AUG 1970

AF Audio Frame System: How do the tape machines work?
CO Color: Red, green, yellow, blue - Can you pick the right color?
P1 Mini-Placement 1: A test to see what you can do.
SI Size/Space: Will the wooden block fit into the box?
SO Sound: Can you tell what made that sound?
SM Smells: How good is your nose? What things smell the same?
IP Iron Pullers: What is a magnet? What things will it pick up?
ST Sorting Things: What is a set? Can you sort things into sets?
P2 Mini-Placement 2: A test to help you speed ahead in science.
IA Inching Along: Which one is longer? Which one is shorter? How long is it?
SH Shapes: What shape is a penny? A stick of gum? A rubber ball?
FT Filling Things: How many will fill this one? How many will fill that one?
NS New Sorts: Can you sort things in new ways? What are the new ways to sort things?
P3 Mini-Placement 3: A test to find out what you know.
SS Sink or Swim: What things float in water? What things sink?
TI Time: How long does it take for something to happen? What things happen first?
MT Measuring Temperature: Do you know how to use a thermometer?
FG Freeple Game: Would you like to play a game with make-believe people?
LI Hot and Cold Light: What is a light? Are some lights hot? Are some lights cold?
TE Temperature-Explore: Here is a problem. Can you figure out how to do it?

Figure 12. Lesson List

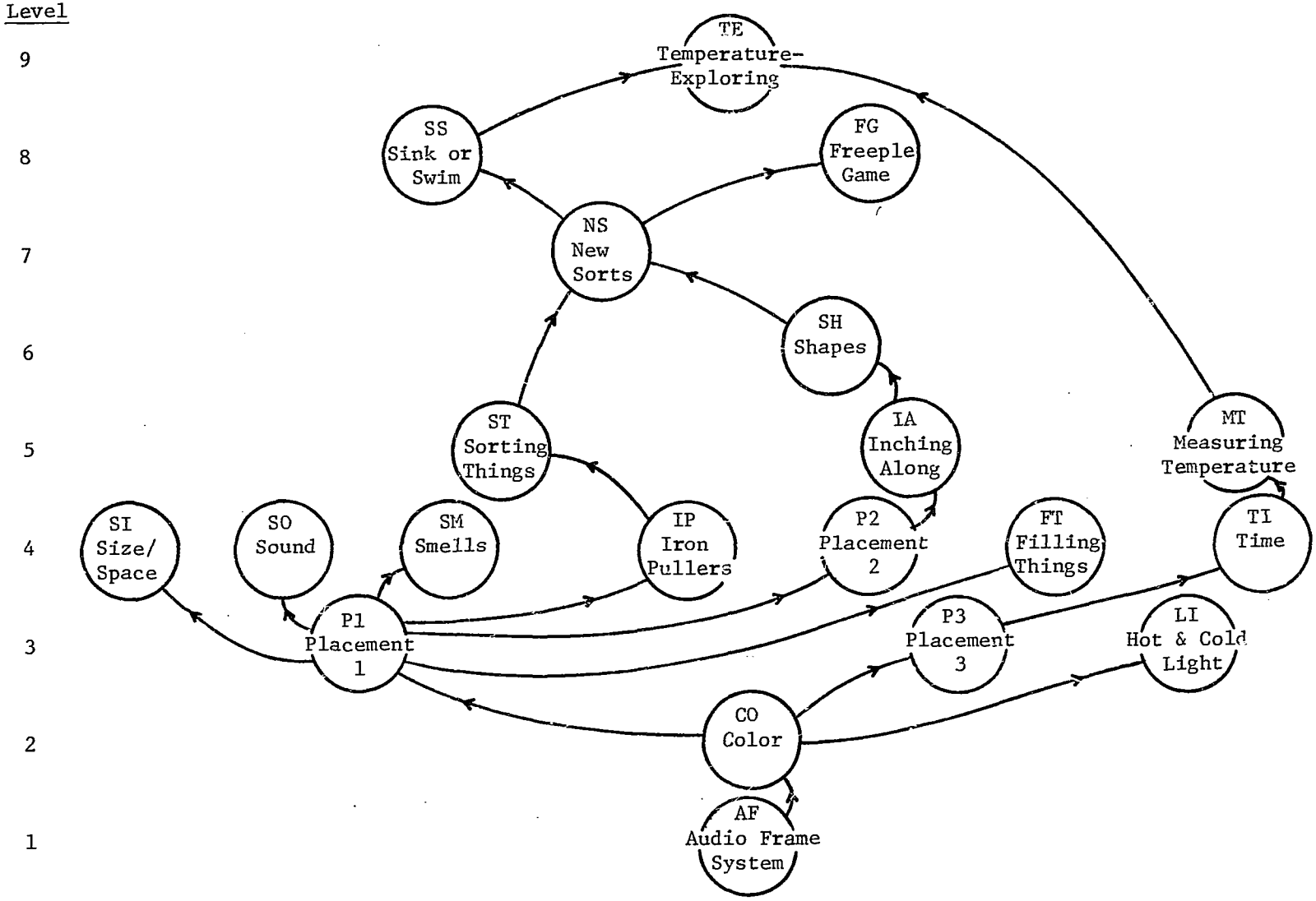


Figure 13. Hierarchical Structure of Curriculum

C. Mastery Data

Recording the student mastery data involved two tasks: getting a record of the material mastered up to the point of applying the program, and providing a mechanism for inputting the mastery of material during the time the management program was in effect. As part of normal classroom procedure, the teacher aide reviewed the records of each student at the end of each Science period. If mastery of all material included in one of the Science units was indicated for a student, the student's record was put aside for examination by the teacher. The teacher would certify mastery for the material, remove the prescription sheet for that unit, and write a new prescription. The corresponding block on the mastery chart entitled "Student Order of Program" (figure 11) was crossed out for the unit mastered. If the results of a placement test indicated prior mastery of one of the units, the corresponding block was crossed out, but the placement test score for that unit was also entered in the corner of the block.

In order to collect a record of the material already mastered, the information from the mastery charts of all the students was compiled. This information was then entered onto a mastery data file by using the program to input mastery data, INPUT:PIL. Dates that were entered on the mastery file corresponded to the date on which mastery was certified. When mastery was indicated by a placement test, the date of the placement test was entered. Dates were coded in five digits: the first two digits stood for

the year, e.g., "70" for 1970; and the last three digits stood for the day of the year, e.g., "001" for 1 January. Sometimes, even though the teacher did not certify mastery of certain materials, classroom management procedures called for action equivalent to mastery. In such a case, the teacher would write "Pull" rather than "Mastery" across the prescription sheet, and the child would be given a new prescription as if he had mastered the material. If such a situation occurred, the character "N" was added to the end of the data field on the mastery file to indicate non-mastery, even though the unit was to be considered mastered for classroom management purposes.

Another situation arose in the classroom management procedures that was different from the original design. For two of the units, any student who did not attain mastery had been given a new prescription as if the unit had been mastered, but his old prescription sheet was marked "Hold," presumably to await modification of the lessons by the curriculum staff. Since this was uniformly practiced across the first grade and neither unit was being prescribed, the points corresponding to these units were removed from the structure which was used in the school. After the management program was operational, further mastery information was entered on the mastery data sheet by the teacher aide after mastery was certified by the teacher. The new mastery data was then added to the mastery data file before the beginning of each class by using the program to input mastery data, INPUT:PIL. The mastery

data file, named MASTERY:PIL, was made up of data records which contained a left and right link, the student number, the name of the student, and room for mastery information of up to five lessons. The records were stored sequentially on the file in the order that they were entered.

D. Implementation of the Management Program

The implementation of the management program as it was applied to IPI Science began on 24 April 1970 with the specification of the points on the curriculum, the "lessons," using the MODEL program. On 28 April 1970 a member of the curriculum staff specified the structure of the lessons using the same program. Mastery data from September, 1969, to April, 1970, was compiled from the records at Oakleaf School, from which the mastery data file was created on 12 May 1970. Thereafter, the mastery data file was updated before each new day of level A Science for the remainder of the school year. Grade one Science at Oakleaf was scheduled in two sections, the first of which met Wednesday and Friday from 11:10 A.M. to 11:50 A.M., and the second on the same days from 2:30 P.M. to 3:10 P.M.

E. Summary of the Year's Mastery Data

At the close of school for the summer at Oakleaf, the IPI Science classroom data were delivered to the curriculum developers. A final check of this material was made by examining the prescription sheets for each student and checking them against the data

on the mastery data file for correct mastery information and dates. The final structure that appeared in figure 13 was used for running the mastery data summary program. It included all the lessons originally specified before the structure was modified for use in the school. Of the lessons appearing in figure 13, the three placement tests were not included in the summary since they were not considered to have been mastered in the sense of other lessons; they were included in the structure only as non-instructional points of the curriculum. These points were excluded by specifying their respective lesson codes at the time the program SUMMARY:PIL was run. In addition, the single character "N" was interpreted to signify non-mastery of a lesson when it appeared after the date in the mastery file.

The output of the mastery summary program for the level A Science data is summarized in Table I. In the first column of Table I the Science units have been arranged in the order of percentage of mastery. It is immediately evident that the last four units had very low mastery percentages, and the curriculum designer can note that the first two of the four, FT and SI, are units that were assigned early in the course of instruction. Thus he can discover what was already known at the school, i.e., that the students were seldom able to master these two units. The last two units, SS and TE, although mastered by a total of only four students, were "exploring" lessons, for which the curriculum designer can be the judge of whether they were satisfactory. Looking

TABLE I

OAKLEAF MASTERY DATA SUMMARY

N = 53

Unit Code	% Mastery	Structural Relation	% Transfer	% Inconsistencies
AF	100	AF → CO	100	0
CO	100	CO → LI	72	0
SO	100	CO → SI	4	2
IP	100	CO → SO	100	6
ST	100	CO → SM	78	13
IA	98	CO → IP	100	9
SH	92	CO → FT	18	6
NS	89	CO → TI	79	0
SM	81	CO → IA	98	0
TI	79	IP → ST	100	15
LI	72	TI → MT	77	30
MT	70	ST → NS	89	0
FG	57	IA → SH	92	9
FT	23	MT → TE	3	0
SI	6	SH → NS	86	43
SS	6	NS → SS	6	0
TE	2	NS → FG	63	2
		SS → TE	0	25

further at the mastery percentages of Table I, one sees that listing the units in order of difficulty was not inconsistent with the order implied by any of the structural relations. That is, although the order in which the unit codes were listed on the summary program output represented a valid instructional sequence, the order in which they are listed in Table I would be a perfectly valid sequence. The fact that none of the structural relations was violated after

ordering the lessons by mastery percentage is one indication of the validity of the lesson hierarchy.

In examining the structural relations that are listed in Table I, two indicators are available to the curriculum designer for evaluating each relation. First, the percentage of transfer is a measure of the number of students who, having mastered the first lesson of the relation, go on to master the second one in sequence. If the two lessons are to be closely related, a high percentage of students would be expected to go on to master the second lesson in a related pair. As Table I shows, this transfer rate was greater than 50% for every relation except those involving one of the four units: SI, FT, SS, and TE. The low rate indicated that either these units were too difficult, or that intermediate material should have been included to facilitate their mastery.

The second indicator, the percentage of inconsistencies, is a measure of the number of students who mastered the second of the lessons in the relation prior to mastering the first. Looking at the last column of Table I, one can see that two relations had a rather high measure of inconsistent order of mastery. Thirty percent of the students who mastered either TI or MT were seen to have mastered MT first. The curriculum designer would recognize that this high percentage could be accounted for by students who mastered MT upon taking the third placement test, which covered both of these units. Forty-three percent of the students who mastered either SH or NS were seen to have mastered NS first,

again mainly by virtue of a common placement test. This result would cause the curriculum designer to re-examine the two relations to determine in each case if the lower-order unit was really necessary for the mastery of the second. For the application of Oakleaf School, then, the results of the summary program indicate to the curriculum designer that he should investigate units SI, FT, SS, and TE; and that he should re-examine the prerequisite relations TI → MT and SH → NS.

The information that the curriculum designer can get from the summary program is dependent upon the particular form of the curriculum, how the structure was formed, and how the program was managed in the school. The output from the summary program is designed only to provide basic feedback, which the user can then apply to his particular situation. For the general applicability of this management program, in which the structure is specified beforehand, the feedback provided by the summary program is concentrated on the relation of actual lesson sequence to postulated curriculum structure.

IV. CONCLUSION

A general model for specifying curriculum structure was developed using the directed graph, or digraph, a mathematical form of a structural model. The use of the digraph to describe a general hierarchical learning structure proved to be a successful solution to the problem of unambiguously specifying such hierarchies. Based on the digraph model, a computer-based management program was developed to allow students using it a degree of choice in the selection of their learning activities. The development of the management program required that an interactive computer program be written that would allow the curriculum designer to specify the structure of his curriculum. It allowed the structural hierarchy to be generated as a result of the curriculum designer's responses regarding the prerequisite relations among the lessons of his curriculum. It provided for analysis of his responses to rule out such structural ambiguities as circular and redundant prerequisite relations among lessons. As a result, it yielded a structure that was a function of the prerequisite relations specified, and was unique for each set of relations so specified.

The management program required that a record of each student's mastery of the curriculum materials be available. The student mastery data base was set up in the form of a linked list, an information structure in which each element of a list points to other related elements, in order to facilitate accessibility. Interactive computer programs which used the mastery data base

included a program to input mastery data, a program to list the data, and the program used by the students to receive options of learning activities. The organization of the mastery data base gave these programs quick access to the records of an individual student without requiring a line-by-line search of the entire file, and made it possible to generate an ordered list of the entire student file without first sorting the records. The program to provide options for the students used information from the mastery data file to eliminate mastered lessons from a student's list of options, and permitted an option to be presented only if all its prerequisites had been mastered. The use of the digraph model for structure insured additionally that changes in the structure or content of a curriculum required regenerating only the stored structural information.

The program designed to provide feedback information to the curriculum designer provided for an after-the-fact examination of the initially specified structure based on student mastery data from the use of the curriculum materials. Feedback information included the percentage of students who mastered each lesson, the percentage of cases for which mastery of a prerequisite lesson was followed by mastery of the next higher-ordered lesson, and the percentage of cases for which an implied order of mastery was violated for two related lessons. The program was designed to provide feedback information regarding the mastery of curriculum objectives regardless of whether the management program was actually employed to prescribe options to the students.

The use of the management program was demonstrated by applying it to the grade one IPI Science classes at Oakleaf School. That the program was successfully applied in the IPI laboratory school does not in itself imply that it can contribute substantially to the individualization effort. Therefore, the question of whether such a program can be useful in an actual instructional situation employing an individualized format will be explored.

Looking at the specific situation in which the program was applied, the level A Science curriculum lent itself well to the hierarchical structure, but the young age of the students required that someone be present to read them their choices. The fact that aid was required to get the instructional materials for the students could have been remedied had the materials been clearly identified by means of a readable code on the boxes containing them. Although the classroom personnel were consistently helpful and cooperative throughout the field test, their enthusiasm for the use of the computer system in the school was clearly lacking after they saw it in operation. The teacher aide, in commenting on the use of the management program by the students to get their prescriptions, felt that the teacher could do the job faster. She was disturbed by the extra time the students spent waiting to use the terminal, time which could have been spent working on the next lesson. The teacher, when asked if he would use the management program the following year for level A Science if it were available to him, replied that he probably would not. The main points against it

were the undependability of the computer system and the extra demand on him and the aide when students would want to start a unit in the middle of the Science period. It would seem from these observations that major points of emphasis in applying the management program elsewhere would be (1) increasing the speed of the program to provide options, (2) improving the reliability of the time-sharing system, (3) limiting its use to older children who can read, and (4) insuring that its use will not interfere with other classroom activities.

The general problem of the usefulness of educational devices, the properties of these devices, and their relation to the processes of education were discussed in a section of an essay by Oettinger and Marks (1969, pp. 159-164). They state that:

> "Novelty and glamor are not the only properties of educational tools worthy of note or sufficient to make them valuable for teaching. What are the really important characteristics?
>
> Cost and value are of obvious importance . . .
>
> More purely technical factors must also be considered: flexibility, generality, scheduling, parallelism, amount, physical accessibility, reliability, maintenance, complexity, comfort, standardization, integration, and content [p. 160]."

Considering each of these factors in turn for the management program as it applied at Oakleaf School may provide some insight regarding future applications.

The cost of such a program is clearly a factor against it at the present time. Costs are characteristically borne by government research grants, but they may come down to school board levels

in years to come as computer costs decrease. The flexibility of the management program can be measured by its ability to meet the needs of the moment. Flexibility was one of the prime objectives in the program's development. It was successfully demonstrated in its application to IPI Science by allowing changes to occur in curriculum structure and by not requiring students to progress strictly according to that structure. Its generality was assured by developing the structural model, although certain basic structural properties were required. Scheduling the use of the program created a problem in that students were permitted to use it only during the class period, which forced many students to wait. The use of additional terminals possible with a time-shared computer and the evolution to a less rigid classroom schedule would help alleviate this difficulty. Parallelism concerns having a resource available simultaneously to several students, and is related to scheduling. If the students may use the management program only during the class period, having several terminals available simultaneously will improve the resource; the need for parallelism will increase as more students use the program. The amount of the resource available would be determined by how much of the curriculum is included in the use of the management program. The application to IPI Science involved a very small portion of that curriculum; only twenty points were included in the curriculum structure. To increase the scope of the program, faster routines must be developed or the delays resulting will be unacceptable.

The physical accessibility of the management program is important to its use. The use of silent CRT terminals will allow their installation in the instructional setting. The location of the Teletype room across the hall from the Science classroom was a major factor in the decision to use the Teletype rather than the inaccessible Sanders CRT terminal. Reliability is a key factor, and proved to be a major fault of the management program. The program failed to perform normally under certain conditions of use; specifically, heavy afternoon time-sharing load and upon depressing the "return" key out of sequence. That the time-sharing system failed to perform at all during five of the class sessions was quite embarrassing, but not abnormal for the computer service. Presumably, the reliability of computer systems will improve as they mature with the years. Related to reliability is maintenance. No breakdowns occurred in the equipment used for the management program in the school, but time was lost due to repairs which had to be made to the time-sharing system computer hardware. One advantage of using time-sharing service from a central computer is that repairs due to breakdowns are likely to be more prompt since many users are depending on the service. Complexity of the program concerns the training required to use it and the ease of continuing operation. Although the management program was designed to be as simple as possible with few demands on the user, the relative frailty of present computer systems would certainly require a person trained in computer usage to be available in case of trouble.

The comfort of the people using the management program is important to its success. The people using it should feel at ease,

although training can bring familiarity with a complex system and in turn instill comfort. The program must respond quickly and efficiently to the needs of those using it, however, in order to gain their confidence. Standardization is a consideration to which computer systems are extremely vulnerable. In the future, standard computer systems should permit truly standard programming languages to exist, but presently users are commonly limited to what has been implemented on the particular system available to them. The management program was written in a language peculiar to one system, and extensive work would be necessary to use it on another. Integration of the management program concerns its ability to tie in with the classroom management of the instructional program to which it is applied. It is particularly important that the management program be adaptable to changes in other parts of the instructional system. If the choices of lesson materials do not correspond to materials actually available, what is the use of the program? If its use interferes with the instructional process, where is its future? Finally, the content of the management program consists of the structure to which it is applied. The structural model must be applicable to the curriculum, and the structure of the curriculum must be meaningful to the actual instruction. Thus, the content is valid only insofar as the objectives of instruction are well defined and the resulting curriculum is based on those objectives.

In conclusion, the management program was successfully developed and later implemented in an individualized setting. Observations of its operation in the school, along with considerations of

the program in light of the properties of an educational device which contribute to its usefulness, indicate the following:

1. The students using the program must be mature enough to take advantage of having a choice of instructional materials presented to them.

2. The instructional setting should be free enough that using the program does not interfere with classroom activities.

3. The computer system used should be reliable enough to be available to the students at any time during the school day, and the program should provide information quickly and efficiently to the student using it.

4. The financing must continue to be borne by outside sources until costs of computer service lower substantially.

5. The use of the program should be tied closely to classroom management procedures so that it may be responsive to classroom needs.

6. The application of the program requires a structured curriculum based on well-defined objectives of instruction in order to be meaningful.

With these considerations in mind, the development of a management program similar to the one that has been described in this investigation can contribute to the success of adaptive programs of instruction.

REFERENCES

Bloom, B. S. Learning for mastery. Evaluation Comment, 1968, 1(2).

Bolvin, J. O. Implications of the individualization of instruction for curriculum and instructional design. Audiovisual Instruction, 1968, 13, 238-242.

Bolvin, J. O., & Glaser, R. Developmental aspects of individually prescribed instruction. Audiovisual Instruction, 1968, 13, 828-831.

Booth, A. D., & Colin, A. J. T. On the efficiency of a new method of dictionary construction. Information and Control, 1960, 3, 327-334.

Carroll, J. B. A model of school learning. Teachers College Record, 1963, 64, 723-733.

Cooley, W. W. Computer assistance for individualized education. Journal of Educational Data Processing, 1970, 7(1), 18-28.

Dwyer, T. A. A lesson designer's guide to CATALYST and the CATALYST/PIL interface. Pittsburgh, Pennsylvania: Department of Computer Science, University of Pittsburgh, 1969.

Flanagan, J. C. Program for learning in accordance with needs. Psychology in the Schools, 1969, 6, 133-136.

Gagné, R. M. The acquisition of knowledge. Psychological Review, 1962, 69, 355-365.

Glaser, R. The education of individuals. Pittsburgh, Pennsylvania: Learning Research and Development Center, 1966.

Glaser, R. Adapting the elementary school curriculum to individual performance. In Proceedings of the 1967 Invitational Conference on Testing Problems. Princeton, New Jersey: Educational Testing Service, 1968. pp. 3-36.

Harary, F., Norman, R. Z., & Cartwright, D. Structural models: An introduction to the theory of directed graphs. New York: Wiley, 1965.

Heimer, R. T. Conditions of learning in mathematics: Sequence theory development. Review of Educational Research, 1969, 39, 493-508.

Klopfer, L. E., & Weber, V. L., Jr. IPI science: A teaching revolution in the making. Science Activities, 1969, 1(1), 27-30.

Knuth, D. E. The art of computer programming. Vol. 1. Fundamental algorithms. Reading, Massachusetts: Addison-Wesley, 1968.

Learning Research and Development Center. Individualized science: A program with relevance for the child. Pittsburgh, Pennsylvania: Learning Research and Development Center, 1970.

Lindvall, C. M., & Bolvin, J. O. The project for individually prescribed instruction (the Oakleaf project). Pittsburgh, Pennsylvania: Learning Research and Development Center, 1966.

Lindvall, C. M., Cox, R. C., & Bolvin, J. O. Evaluation as a tool in curriculum development: The IPI evaluation program. AERA Monograph Series on Curriculum Evaluation, Chicago: Rand-McNally, 1970, No. 5.

Lipson, J. I. An individualized science laboratory. Science and Children, 1966, 4(4), 8-12.

Oettinger, A. G., & Marks, S. Run, computer, run: The mythology of educational innovation. Cambridge, Massachusetts: Harvard University Press, 1969.

Resnick, L. B. Design of an early learning curriculum. Pittsburgh, Pennsylvania: Learning Research and Development Center, 1967.

Suppes, P. The uses of computers in education. Scientific American, 1966, 215(3), 207-220.

University of Pittsburgh Computer Center. PIL/L: Pitt interpretive language for the IBM system/360 model 50. Pittsburgh, Pennsylvania: University of Pittsburgh Computer Center, 1969.

Washburne, C. W. Educational measurements as a key to individualizing instruction and promotions. Journal of Educational Research, 1922, 5, 195-206.