DOCUMENT RESUME

ED 137 089 SE 022 182

AUTHOR	Polin, Glenn M.
TITLE	MACSYS: An Automated Curriculum System for Elementary Mathematics.
INSTITUTION	Illinois Univ., Urbana. Computer-Based Education Lab.
SPONS AGENCY	National Science Foundation, Washington, D.C.
REPORT NO	CERL-R-X-48
PUB DATE	Aug 76
GRANT	USNSF-C-723
NOTE	46p.; Contains occasional light type
EDRS PRICE	MF-$0.83 HC-$2.06 Plus Postage.
DESCRIPTORS	*Computer Assisted Instruction; *Curriculum; *Elementary School Mathematics; Elementary Secondary Education; Instruction; Mathematics Education; *Program Des riptions
IDENTIFIERS	*PLATO

ABSTRACT

Details are given of the Elementary Mathematics Automated Curriculum System (MACSYS) used with fourth, fifth, and sixth graders at six elementary schools. The four decision-making components of MACSYS are discussed, the structure of the sessions in which the students interact with the computer is described, and the structure of the curriculum is explained. Seven different attributes of a lesson are analyzed, and the process for determining eligible lessons is described. (DT)

CERL

Computer-based Education Research Laboratory

PLATO

University of Illinois

Urbana Illinois

MACSYS: AN AUTOMATED CURRICULUM SYSTEM FOR ELEMENTARY MATHEMATICS

GLENN M. POLIN

CERL Report X-48

AUGUST 1976

MACSYS: An Automated Curriculum System for Elementary Mathematics

by

Glenn M. Polin

August 1976

Computer-based Education Research Laboratory
University of Illinois at Urbana-Champaign

This manuscript was prepared with partial support from the National Science Foundation (USNSF C-723) and the University of Illinois at Urbana-Champaign.

ACKNOWLEDGEMENTS

Many of the ideas used in the design of MACSYS were originally developed by Robert B. Davis, Project Director; I extend my thanks to him.

Each member of our three curriculum teams:

Graphing:	Donald Cohen Gerald Glynn
Whole Numbers:	Bonnie Anderson Seiler Charles Weaver
Fractions:	Sharon Dugdale David Kibbey

has made substantial contributions to the design of the final product and the many preliminary versions.

Additional thanks go to Bonnie Anderson Seiler, Allen Avner, Elaine Avner, William Golden, Elisabeth Lyman, and Bruce Sherwood, whose comments on earlier drafts of this paper have improved it far beyond what I could have done on my own.

For their assistance in the preparation of this paper, I owe many thanks to Sibyl Pellum, typist, and Wayne Wilson, graphic artist. Thanks are also due to Sheila Knisley and Roy Lipschutz for their assistance in the preparation of the final publication.

Finally, I owe a special thanks to Keith Slaughter, who worked with me on MACSYS from June 1974 until March 1976. Without his help, I might never have found the time to write this paper.

TABLE OF CONTENTS

I. INTRODUCTION

The developmental work of Risken and Webber (1974), begun in 1971, and the subsequent introduction of router lessons to the PLATO system in December 1973 have greatly enhanced the PLATO system's capability of delivering a fully automated, fully exportable curriculum -- one which does not require human intervention on a day-to-day, or even week-to-week basis.

The PLATO Elementary Mathematics Group, one of several curriculum projects at the Computer-based Education Research Laboratory at the University of Illinois where the PLATO system was developed, has been at work since 1973 developing an Elementary Mathematics Automated Curriculum System (MACSYS), designed for the unique participants and environment of the teaching of elementary mathematics.

In the 1975-76 school year, as part of a National Science Foundation contract, MACSYS was in operation in thirteen classrooms in six elementary schools in the Champaign-Urbana area. The typical classroom had four PLATO terminals, usually located to one side of the room; the grades involved were fourth, fifth, and sixth. Over 15,000 hours of session time were recorded during that year. On an average school day, 250 students each received a 30-minute session of mathematics instruction.

Robert B. Davis (1974) has described the scope of the two PLATO elementary curriculum projects; it is my intention here to describe the specific design of MACSYS, focusing on those aspects which differ most radically from the rest of the curriculum projects on the college-oriented PLATO system.

Some Introductory Concepts

As it is commonly used on PLATO, the word "session" means the time the student is on the computer system. If he is on for ten minutes, he had a ten minute session; if he is on for an hour, he had an hour session. Most college students work for fifty-minute sessions, the length of a class period. Generally this fifty minutes is spent in no more than two or three lessons, with the student finishing each lesson before going on to the next.

A MACSYS session may be defined as "*a structured sequence of lesson segments lasting about thirty minutes*." Each element of that definition is explained below.

It is the session selector[1], a principal program of MACSYS, which controls each student's session and determines when it is over -- not the classroom teacher, or a proctor, or the student. A student may sign in and out of MACSYS several times during the course of one session; each time he returns, he continues working wherever he left off in his current session[2]. This is important because the *sequence* of lessons provided the student is *structured*, i.e. different kinds of lessons occur at different times during the thirty minutes. An example of this is that students may only play games during the last ten minutes of a session.

A *lesson segment* is a piece or portion of a lesson. Lessons in MACSYS are designed to be studied a piece at a time, not all at once. On the average, a student sees seven lesson segments in a session; but sessions containing from three to thirteen lesson segments are not unusual.

Further, most lessons do not have a prespecified number of portions or segments. They rely instead on the student to decide when he has had enough of this particular lesson for the time being. By pressing a designated key, the student may leave the lesson he is in and choose a different lesson to study. Thus the student defines the portions of the lesson for himself. When a student chooses to do another segment of a lesson he has started previously, MACSYS takes the responsibility for remembering where the student was and restarting him appropriately. Note that while a student may decide he has temporarily had enough of a lesson, he does not decide whether he is permanently finished with the lesson; this decision belongs to another MACSYS program, the curriculum interface.

Finally, I say that the session is *about thirty minutes long.* Obviously, this is of deliberate design. Students are never arbitrarily interrupted while studying a lesson -- they either choose to leave, or reach the end of the lesson or one of its segments. The lack of a more precise session length[3] seems of minor concern compared to the anguish generated in a student who has been unexpectedly interrupted in the middle of a lesson.

We will now look at the components of MACSYS and see the way in which they interact to determine each student session.

II. A COMPONENT DESCRIPTION OF MACSYS

There are four decision making components of MACSYS:

1. Curricula
2. Session Selector
3. Teachers
4. Students

Components one and two make up the programmed, "automatic" part of the system; components three and four are its human elements. Table 1 summarizes the type of decisions each element is responsible for.

Table 1. Decision Making in MACSYS

Components	Type of Decisions
Curricula	1. Data bases contain invariant decisions, e.g., Lesson A must come before Lesson B. 2. Interfaces make student specific decisions, e.g., This student needs remedial Lesson C.
Session Selector	1. Determines the lessons the student is eligible to see.
Teachers	1. Assign students to curricula. 2. May assign individual lessons to students. 3. May allow students to skip chapters or take an advanced route through chapters.
Students	1. Choose the lessons they see from a list prepared by the session selector. 2. May decide how much of each lesson they will study at one time.

The Curricula

There are currently three curricula available as part of MACSYS: Graphing (Cohen and Glynn, 1974), Whole Numbers (Seiler and Weaver, 1976), and Fractions (Dugdale and Kibbey, 1975). Of the over 15,000 hours of session time recorded last school year, 12% of the time was spent in Graphing lessons, 41% in Whole Number lessons, and 47% in Fractions lessons.

As seen from the outside, by the teacher, a curriculum is divided into chapters. At the present time, the Graphing Curriculum has five chapters, the Whole Numbers Curriculum has five chapters, and the Fractions Curriculum has nine chapters. The chapters of a curriculum are arranged as a linear list, much the same as chapters in a book. Each chapter covers a specific topic in the curriculum and thus consists primarily of lessons on that topic; however, other lessons are included in the chapter for readiness building, review, and general enrichment.

Internally, each curriculum in MACSYS is seen to consist of the following components:

1. A Data Base
2. Many Lessons
3. A Curriculum Interface

The Curriculum Data Base

The curriculum data base contains two kinds of information: information about lessons in the curriculum and information about the chapters of the curriculum. The lesson information includes the information needed to access a lesson, the title of the lesson, and its data requirements. The chapter information includes the names of the lessons to be included in each chapter, the attributes assigned to them, and their ordering in the chapter's curriculum trees. The ideas of attributes and trees will be discussed in full in later sections. All of the invariant curriculum decisions are contained in the data base. An example of an invariant decision is "Lesson A must come before Lesson B."

The Lessons

Lessons provide instruction for the student and measure the performance of the student within the range of that instruction. A lesson which only measures performance is called a "checkup." Lessons do not make curriculum decisions on the basis of student performance; that is the job of the curriculum interface.

A lesson in MACSYS is defined by a PLATO file name and a part number, which distinguishes different lessons in the same file. Most often, lessons within the same file are closely related and are sometimes just slight variations of one another, e.g., the same lesson form with a different set of problems. They are considered distinct lessons so that each may be assigned its own attributes and its own place in the trees.

The Curriculum Interface

The curriculum interface is a program whose primary function is to make the student specific curriculum decisions, those which can not be made in advance. Each curriculum has a curriculum interface, written as part of the job of designing the curriculum.

Looking at Figure 1, we see that after each lesson, the student stops at the interface before returning to the session selector. During each student's brief (and invisible) stay in the interface, the interface interprets the unique performance measure of the immediately previous lesson and other past lessons, and sends its decisions to the session selector. An example of such a decision is "This student needs Lesson C," where Lesson C may be a remedial lesson designed to counter a problem the student is having.

Because the interfac? is not an obvious component of a curriculum, one might well ask whether the lessons or the session selector could not do the same job as well. The answer is that either could do the job, but neither as well.

While we have not yet discussed the session selector's many functions, the fact that there are three independent curricula being run by a single session selector should make most readers realize why these decisions could not be put into the session selector. Each curriculum has changed and been improved many times since its first trial. We clearly could not have three curriculum groups all trying to alter the same program.

In early implementations of MACSYS, the lessons performed the role the interface now performs. Code was attached to the end of each lesson to implement whatever student specific decisions needed to be made after that lesson.

There were several disadvantages to this mode of operation. The primary one was that the student specific decision making apparatus was spread out in all the lessons of the curriculum; this compared unfavorably with the ready accessibility of the invariant curriculum decisions in the data base. It is quite simple to understand the decision making in a chapter when it is all in one place and quite difficult when it is spread out in perhaps forty lessons.

A secondary consideration was that we did not wish the lessons to change. In many cases, lessons were written by authors now long departed and no current member of the staff was familiar with them. Changing any part of a lesson, even code merely "attached" to the end of the lesson always carries a significant risk. By using an interface, curriculum decision making can often change without changing a single lesson.

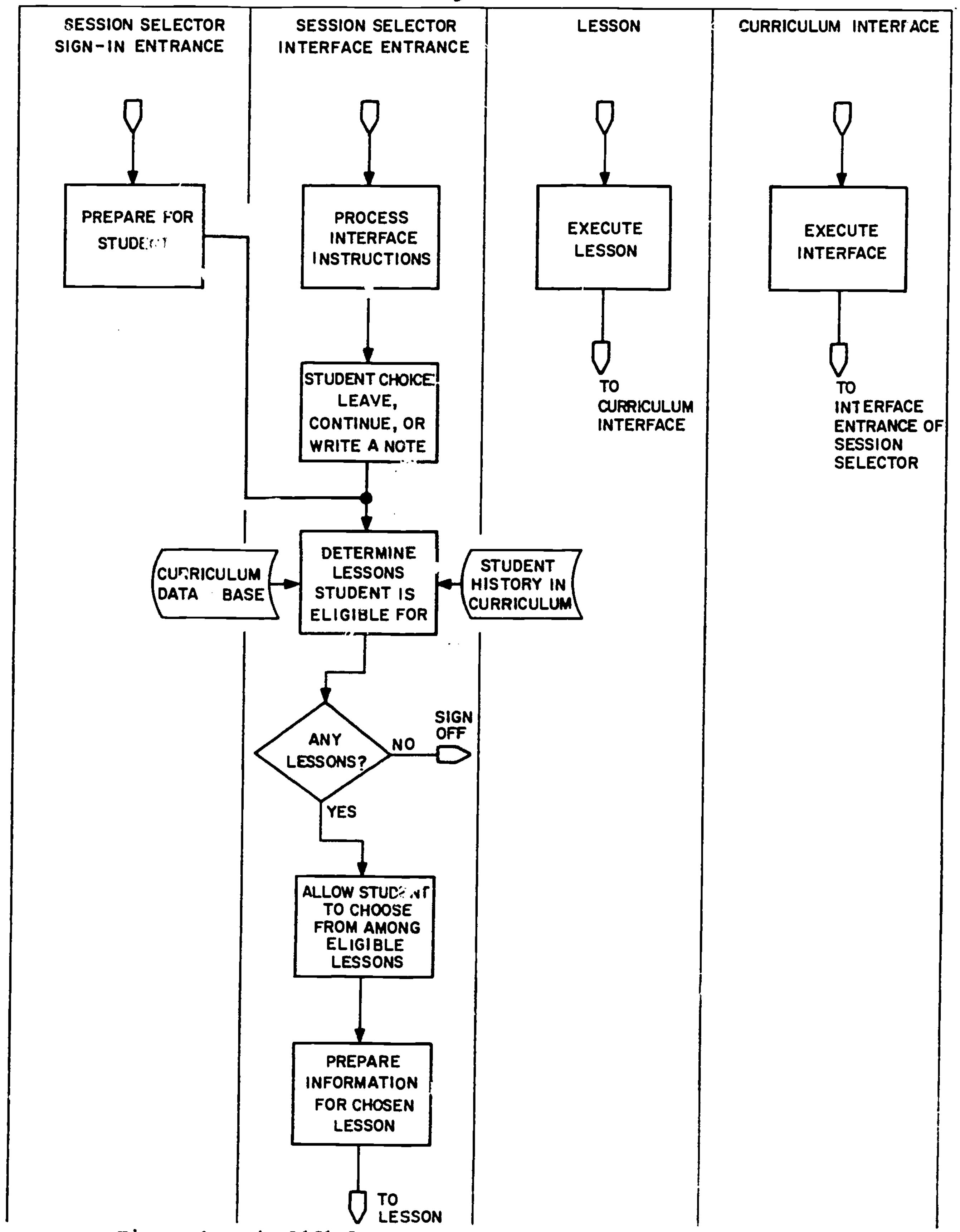

SESSION SELECTOR SIGN-IN ENTRANCE
SESSION SELECTOR INTERFACE ENTRANCE
LESSON
CURRICULUM INTERFACE
PREPARE FOR STUDENT
PROCESS INTERFACE INSTRUCTIONS
EXECUTE LESSON
EXECUTE INTERFACE
TO CURRICULUM INTERFACE
TO INTERFACE ENTRANCE OF SESSION SELECTOR
STUDENT CHOICE: LEAVE, CONTINUE, OR WRITE A NOTE
CURRICULUM DATA BASE
DETERMINE LESSONS STUDENT IS ELIGIBLE FOR
STUDENT HISTORY IN CURRICULUM
ANY LESSONS?
NO
SIGN OFF
YES
ALLOW STUDENT TO CHOOSE FROM AMONG ELIGIBLE LESSONS
PREPARE INFORMATION FOR CHOSEN LESSON
TO LESSON

Figure 1. Simplified System Flow Chart for MACSYS Students

The Session Selector

The primary responsibility of the session selector can be seen clearly in Figure 1: to determine what lessons the student is eligible for. This is a complex determination involving the student's personal history, information from the curriculum data base, commands from the curriculum interface, and decisions made by the teacher. This determination will be the subject of much of the rest of the paper; still, some other session selector tasks are worthy of note.

To free the classroom teacher from the necessity of policing the terminals, the session selector allows each student only one session during school hours, thus preventing the more aggressive children from monopolizing the terminals. The teacher can, however, give any or all students extra sessions; additionally, the teacher has the power to temporarily restrict individuals or the entire class from using PLATO.

Since students generally do not finish lessons in one sitting, some data must be saved to restart the student appropriately. The session selector collects this data when a student returns from a lesson, stores it while that student does other lessons, and returns the data to the lesson when the student next studies it. Much of the data collected on a student is collected by the session selector.

Finally, the session selector includes an error accounting system, used to detect and report errors in the lessons, the interfaces, and the session selector itself.

The Teacher

Each classroom teacher must decide which curriculum or curricula to assign to which students on which days of the week. This may be a single decision, such as assigning all students to Fractions Monday through Friday, or a series of individual decisions. An example of such individual decisions might be to assign John to the Graphing Curriculum on Monday, Wednesday, and Friday, to the Whole Numbers Curriculum on Tuesday, and to the Fractions Curriculum on Thursday; and to assign Julie to Whole Numbers Monday through Thursday and Fractions on Friday. Each student in the class may be given a unique assignment, if desired.

Once the teacher has made the initial assignment of students to curricula, the teacher is not required to interact further. The elementary school teacher is a busy person, one whose free moments are noticeably rare. The collective experience of our group suggested that our system would be widely used only if it did not require teacher interaction on more than an occasional basis.

While our system does not <u>require</u> further teacher interaction, it does encourage it. The initial curriculum/student assignment may be changed at any time, and extensive reporting to teachers on the subject of student performance gives them a basis to make their decisions. Teachers may also allow students to skip chapters of a curriculum, or take the alternate (advanced) path through a chapter. In addition, teachers may assign individual lessons to students where they see a need which the regular curriculum is not fulfilling.

<u>The Student</u>

Wherever possible in MACSYS, students are given choices. They may choose what name PLATO will call them; they choose which symbol will represent them in a game; they often are allowed to choose the level of difficulty of the problems they see, and sometimes the problems themselves; and as stated previously, there are many opportunities for students to put off studying one lesson and choose another.

The student chooses the lessons he sees; this is his primary influence on the specific content of his session. Looking at Figure 1, we note again that each time the student returns to the session selector, he is presented with a list of lessons to choose from; sometimes as few as one or as many as eighteen, but most often six or eight. It is the procedure by which the session selector determines the contents of that list that we will now be concerned with, and we shall begin by defining the precise structure of the student session.

III. STRUCTURE OF THE STUDENT SESSION

Previous work by Robert B. Davis (1973) indicated that a mathematics session for elementary school students should have a definite structure. The lesson material offered during the session should not get continually harder, building on itself, as a college lecture does. If the student loses the thread of the material offered during such a session, he will be lost from that point on, with resulting bad feelings about mathematics and PLATO.

Davis' work indicated that the session should start with something familiar. The middle of the session should be concerned with the ongoing new material, and the end of the session should return to familiar things, usually games. It is these ideas about session structure for classroom mathematics which we have attempted to build into our session selector.

Within the thirty minutes of the session, the following divisions exist.

I. Notes from the classroom teacher or PLATO staff to the student

II. One lesson prescribed by the classroom teacher

III. Instruction in the assigned curriculum and chapter for for that day (20 minutes)

IV. Student choice of games (8 minutes)

V. Optional note from the student about his PLATO session (2 minutes)

Each classroom teacher is responsible for the lessons his students see in Sections II and III. Section II consists either of a single lesson, or none at all. Section III may contain any number of lessons, as it is possible for the student to spend his entire time in one lesson, just as he has the opportunitv to return to the session selector every few minutes and choose a different lesson.

The PLATO staff assigns the lesson material in Section IV, which is known as "Game Slot"[4] to the students. Each student's game list is individualized, containing within it a subset of games, common to all students, and a few games specifically appropriate to whichever chapters of the three curricula the student is currently working in. No restrictions are placed on which lessons the student chooses in Game Slot, or how much or little time he spends in each one.

In any particular session, Sections I, II, and V may or may not be present. In general, the time remaining from one section (positive or negative) is subtracted from the time allotted to the next section.

Within Section III, there are two slots. While it is not a hard and fast rule, the first slot usually contains review and is short (five minutes or less). The second slot, called "Main Slot," contains the material new to the students.

Looking now at Davis' model, we see that the teacher prescription and the review constitute the beginning, familiar material. The main slot of Section III makes up the ongoing curriculum, and the session ends on a familiar note with eight minutes of games.

The boundaries of Sections III and IV, and in fact of all the sections, are not rigid. Section III may end early or late -- early if there are no further lessons for the student to do (unlikely, but possible), or late if the student progresses more slowly than expected or chooses to spend more time than usual in a lesson. In addition, a student who has reached "Game Slot" may continue working on his "Main Slot" instead, if he so chooses.

Thus while some sessions may deviate in particulars from the general model, this deviation is almost always the result of student choice.

IV. STRUCTURE OF THE CURRICULA

Within each thirty minute session, ten minutes are devoted to games assigned by the PLATO staff, a lesson assigned by the teacher, and notes written by the student; it is the twenty minutes of Section III which are devoted to the assigned curriculum. We may now begin to discuss the structure of that curriculum and the way in which the student, session by session, progresses through it.

A curriculum, to review briefly, is seen as consisting of chapters. The student's classroom teacher assigns him to one or more curricula; he works on each curriculum one chapter at a time. When he is finished with one chapter of a curriculum, he starts the next chapter of that curriculum. Chapters within a curriculum are arranged in linear order, somewhat like chapters in a book.

Internally, a chapter is made up of one or more curriculum trees. By convention, lessons in the first or primary tree of a chapter are the lessons that <u>all</u> students who enter the chapter will see. The lessons in the secondary trees are those which only <u>some</u> students will see; it is the responsibility of the appropriate curriculum interface to activate these lessons for a particular student if and when he needs them, as determined by some data collected from a lesson.

Figure 2 shows a sample curriculum tree. Each node or circle in the tree in Figure 2 represents a lesson. The relationship between lessons in the tree is expressed as that of predecessors and successors. In verbal terms, a successor of a lesson is one which comes after the lesson; a predecessor obviously must precede the lesson. In graphical terms, any

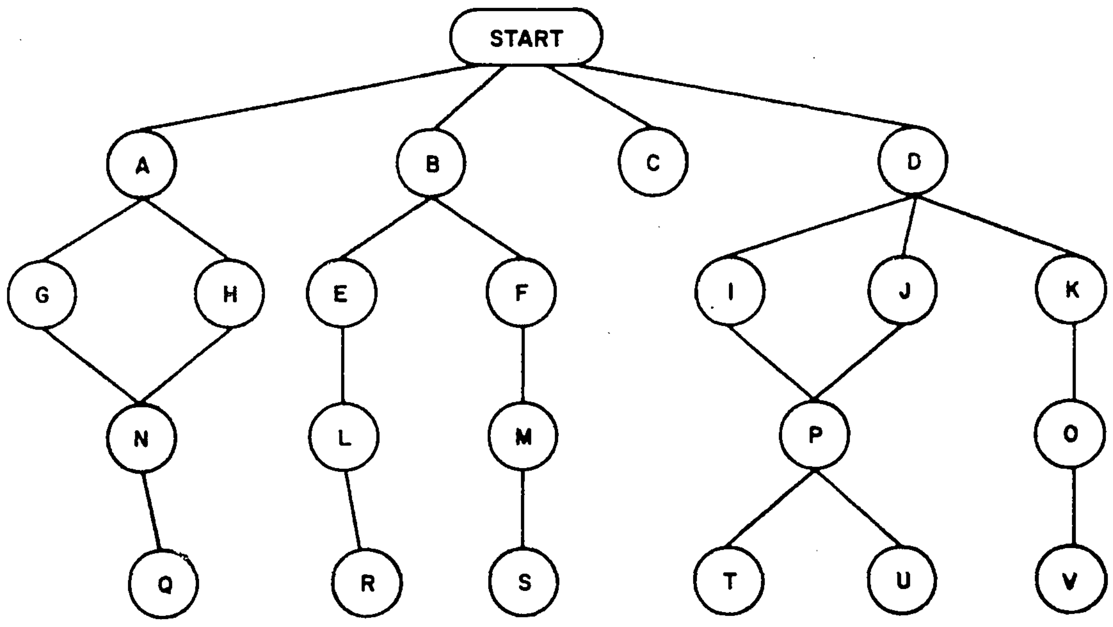


Figure 2. A Curriculum Tree

two lessons in a chapter connected by a line with no other lessons between them are in successor/predecessor relationship. The one on top is a predecessor of the one below, and the one below is a successor of the one on top. For example, in Figure 2, we say that Lesson F is the predecessor of Lesson M, and that Lesson M is the successor of Lesson F. Lessons in a curriculum tree may have more than one successor and/or predecessor, e.g., Lesson P in Figure 2 which has predecessors I and J, and successors T and U.

The Student's Position in a Chapter

How do we define the position of a student in a chapter? First, each lesson in a chapter is always considered to be in one of two complementary states for a given student at a given moment -- available or unavailable.

When deciding what lessons the student may choose from, the session selector begins the process with the set of available lessons; the student is never offered an unavailable lesson.

What then determines whether a lesson is available or unavailable? When a student first enters a chapter, the successors of the start node of the primary tree are made available to him (lessons A, B, C, and D in Figure 2, if we assume that it is the primary tree of some chapter). All other lessons are considered unavailable. *A lesson further down the curriculum tree becomes available only when each one of its predecessors has either been satisfied or finished.* Thus in Figure 2, lessons G and H change from unavailable to available when Lesson A is either satisfied or finished. Similarly, Lesson P changes from unavailable to available when each one of lessons I and J is either satisfied or finished.

Satisfied and finished are two completion stages in the history of a student's experience with a lesson. *A lesson is satisfied when the student has attained the minimum requirement for proceeding past that lesson in the curriculum.* An example of a minimum requirement we might use is "the student has answered 70% of the lesson's problems correctly." If this were the requirement for Lesson A in Figure 2 to be satisfied, then when the student reached this level of proficiency in the lesson, the curriculum interface would signal the session selector that Lesson A was satisfied. The session selector would then note that all of the predecessors of lessons G and H were either satisfied or finished, and it would therefore make lessons G and H available.

When a student finishes a lesson, it means that not only is the minimum curriculum requirement satisfied for that lesson but, in addition, the student has completed the lesson, and it should no longer be available *to him.* An example of a finished criterion we might use is "90% of the lesson's problems must be answered correctly." When a student achieves this performance level, the curriculum interface would notify the session selector that Lesson A was finished. The session selector would then not only make lessons G and H available, but would also make Lesson A unavailable. Once a lesson is available to a student, it will become unavailable only after it is finished.

In my example, the criterion for satisfied and finished are different, with the satisfied criterion being less demanding. This is the most common case, assuring that a student will make progress in the curriculum even though he has not mastered a particular lesson. If the lesson has the same criterion for satisfied and finished (which is equivalent to the lack of a satisfied criterion), then the student will proceed in the curriculum only when he has reached the finished criterion level. A lesson may also be designed with a satisfied criterion and no finished criterion, letting the student proceed in the curriculum while the lesson remains available until the chapter ends.

To summarize, the history of a student with respect to any lesson in the primary tree of a chapter (e.g., Lesson P in Figure 2) is as follows: first, the student must satisfy the predecessors of Lesson P (lessons I and J); at that point, Lesson P becomes available to the student, and may begin to appear on his list of lesson choices. When the curriculum interface detects that the student has met the requirements for finishing or satisfying the lesson, it will inform the session selector; when the session selector marks Lesson P as satisfied or finished, the successors of Lesson P, lessons T and U, will become available. If the student has finished Lesson P, it will become unavailable, and the student will no longer see it on his choice list.

We can therefore define a student's position in a chapter by stating what lessons are available to him; this is determined by knowing which lessons the student has satisfied and which lessons he has finished.

Role of the Curriculum Interface

It is now possible to state more precisely what the role of the curriculum interface is. There are five commands a curriculum interface may send the session selector:

1. Declare any lesson in the chapter finished.
2. Declare any lesson in the chapter satisfied.
3. Make any secondary lesson in the chapter available.
4. Start any secondary tree of lessons in the chapter.
5. End the chapter.

Thus we may say that while the curriculum tree is concerned with the order of the lessons in the chapter for all students, the curriculum interface is concerned with the speed of each student's progress through those lessons, and with providing any extra lessons a particular student might need.

Why Use a Curriculum Tree

The quite natural question arises, "Why use trees to represent curriculum?" The answer is that the tree is the natural generalization of some commonly used ways of designing a curriculum.

One such common way is the straight line or linear curriculum; when one lesson is finished, the student starts the next lesson. Another common approach for college students is the choice list, where the student may choose from all the lessons in the chapter, thus allowing him to decide the order of lessons for himself. Both of these approaches are easily implemented with a tree structure. Also, in either approach, lessons may be left available for review by the student if they are declared to be satisfied instead of finished whenever the student completes them.

The curriculum tree, then, provides a very general way of defining in what order the lessons of a chapter must be done. If very little order is required, as in a choice list, the curriculum tree can express that. If very strict order is required, the curriculum tree can also express that. The curriculum tree is given additional flexibility by the notions of satisfied and finished, allowing us to define one condition for going on in the curriculum, and another condition for finishing a lesson.

The Curriculum Tree Is Not Enough

The curriculum tree, by itself, is not enough; it does not provide enough data to properly control the student's progress through the curriculum. A tree, for instance, does not contain any information about the relative importance of lessons; one available lesson may be a checkup and the other a review lesson, but using only the information in the curriculum tree, they are indistinguishable.

There are other types of desirable curriculum control other than that offered by the tree. One might want to allow a student to choose the same lesson he just returned from or one might want to restrict students from seeing a particular lesson more than once a session. A curriculum designer might wish to specify that a particular sequence of lessons within a tree should be continuous, i.e., uninterrupted by other lessons. Neither of these are possible using only a tree.

To gain further control in specifying curriculum, each lesson must carry some information about itself beyond the names of its predecessors and successors. We call this information the attributes of a lesson. Each lesson in a chapter is assigned values for seven attributes; these attributes give the session selector the information it needs to implement the additional kinds of control that the curriculum designer requires.

V. ATTRIBUTES

Seven attributes of a lesson are listed and grouped in Table 2.

Table 2. Attributes of a Lesson

Group I	Group II
Attribute: Slot Range : 1-2 Units : none	Attribute: Time Range : 0-15 Units : minutes
Group III	**Group IV**
Attributes: Rank Priority Range : 0-15 Units : none	Attributes: Frequency Delay Forced Range : 0-15 Units : sessions

<u>Group I</u>

<u>Slot</u>. Within the twenty minutes of Section III, there are two internal time divisions called <u>slots</u>. Each of these slots is assigned a portion of the twenty minutes, as part of the specification of the module. Each module lesson is then assigned to one of these two slots and can only be offered to the student within the assigned slot.

In this way, two groups of materials within the same module may be isolated from each other, and delivered each within its own time slot during the session. As was stated previously, our general usage of the slots is for Slot I to offer review or repetitive practice lessons, and for Slot II to consist of lessons which are new to the student.

When the term *available* is used, it should be taken for granted that the lesson is only available or unavailable within its own slot. A lesson is never considered outside its slot.

In one respect, Slot II is very different from Slot I. In Slot I, the session selector offers the student only lessons from today's assigned chapter in either Graphing, Whole Numbers, or Fractions. But in Slot II, the session selector will add four *alternative lesson choices* to the student's choice list from the other curricula. Bear in mind that the Slot II choice lists, even with these additions, are composed primarily of the assigned curriculum. If, for example, it is Friday and a student is assigned to the Graphing Curriculum, then his Slot I choice lists will consist entirely of Graphing lessons; however, his Slot II choice lists will include two lessons each from the Fractions and Whole Numbers curricula, in addition to whichever lessons would normally occur in the Slot II Graphing Curriculum. The alternative lesson choices from Fractions and Whole Numbers will be appropriate to whichever chapters the student is currently studying in those curricula.

As I have stated previously, the student is given a choice wherever possible in MACSYS and this is the principal design goal which suggested the device of alternative lesson choices. In this way, the teacher still assigns curricula to the student, but the student has some capacity to override that decision, by choosing the alternative lessons from the other curricula. We are hopeful that this device of alternative lesson choices contributes to skill maintenance in those curricula to which the student is assigned infrequently, i.e., one or two days a week; however, there is no empirical evidence to support that hope.

Group II

Time. The student session must last as close to thirty minutes as possible. A session that is too short or too long is disruptive to the classroom routine. However, it is also disruptive to the student to terminate his or her session abruptly.

The time attribute tells the session selector the expected time in minutes for the average student to complete one segment of that lesson. This estimate is further strengthened with the use of the student's own time factor. The time factor is a ratio of two sums -- the sum of the actual time in all activities the student has been in to the sum of the estimated time for all those activities. Thus a student who habitually works slower than our estimates develops a time factor greater than one, while a student who works faster develops a time factor smaller than one. If the expected time for a lesson is eight minutes and the student has a time factor of .75, the session selector may offer that lesson to that student only if the student has at least six minutes left in the current slot or in the session.

This method of prediction is most useful where the segments of a lesson are fixed and not subject to modification by the student or the lesson environment. While this was true of lessons written early in the term of the project, lessons have become increasingly less rigid, allowing students to exit at many points and making their own time checks to assure that the student does not run over the end of the session or a slot. For these lessons, the time attribute is a minimum time, as opposed to an expected time. Since the portion of these lessons the student studies is no longer a fixed quantity against which we can measure our expectations, the student's time factor for these lessons is defined as one.

Group III

Rank. The rank attribute allows the curriculum author to create varying ranks or levels of importance within each chapter. The session selector uses the following rule as part of its determination of which lessons the student is eligible for:

> *If "n" is the lowest non-zero rank of all lessons available in this chapter, then only lessons of rank "n" and rank zero may be offered to the student. (Rank zero lessons are always treated as if they were of rank "n," the current lowest rank, whatever that may be.)*

The rule prevents the session selector, for example, from offering the student a rank seven lesson while a rank five lesson is available; if "n" equals five, then the student may only be offered lessons of rank five and rank zero. Thus the more important lesson, the rank five lesson, will be done before the less important rank seven lesson.

Priority. After the lowest non-zero rank is determined, the priority values of the available lessons at that rank are considered. The routing procedure algorithm will attempt to offer the lessons whose priority values are lowest, as it does with rank, but these lessons must also pass some situational tests. For example, the adjusted time attribute for each lesson to be offered must be less than or equal to the remaining slot and session time. Thus if the lowest priority value is "p" but no lessons at priority value "p" pass the situational tests, then the routing procedure will examine the lessons at priority "p+1." This process terminates when a priority value is found at which one or more lessons pass the above tests. All lessons found at this priority are included in the student's choice list.

Thus the priority attribute provides a suggestion of order preference to the session selector, but not a command, as does rank. A priority seven lesson (of rank "n") may be offered before a priority five lesson (of rank "n") if the situational factors prevent the priority five lesson from being offered. But while a rank seven and a rank five lesson are both available to a student, the rank seven lesson will never be offered.

Priority values zero and one have special meaning. When the student's choice list consists of priority one lessons, the session selector does not append the alternative lesson choices from the other two curricula to the choice list. If the list has priority zero, the student is given no choice at all. The requirement that the lesson fit into the remaining slot time is waived and the student is sent directly to the lesson.

Examples Using Rank and Priority

Figure 3 shows the same curriculum tree as pictured in Figure 2 with the rank of each lesson shown in parentheses underneath the lesson. The student who begins a chapter with this tree will start with Lesson C, because it has rank one and the other available lessons have rank ten. (One could imagine that Lesson C is a pretest for the chapter and thus needs to be done first.)

When Lesson C is finished, lessons A, B, and D will be offered to the student. The session selector does not distinguish among them because all have rank ten. The student will continue to be offered lessons A, B, and D until he finishes one of them, say B. When B is finished, lessons E and F become available, and the choice list offered to the student will now only contain E and F, since they are of rank five, the current lowest rank among the available lessons.

When both E and F are finished, then lessons L and M will be offered
ıe student. Note that while the student will finish one of the two
ons first, say E, the student may not begin Lesson L, Lesson E's
essor, because it has rank six, which is greater than the rank five
ıe still available Lesson F. Only when both of the rank five lessons
finished will the rank six lessons be offered the student. By the same
ning, only when both rank six lessons (L and M) are finished will the
n choice list contain lessons R and S, as only then will rank seven
ıe lowest rank of the available lessons.

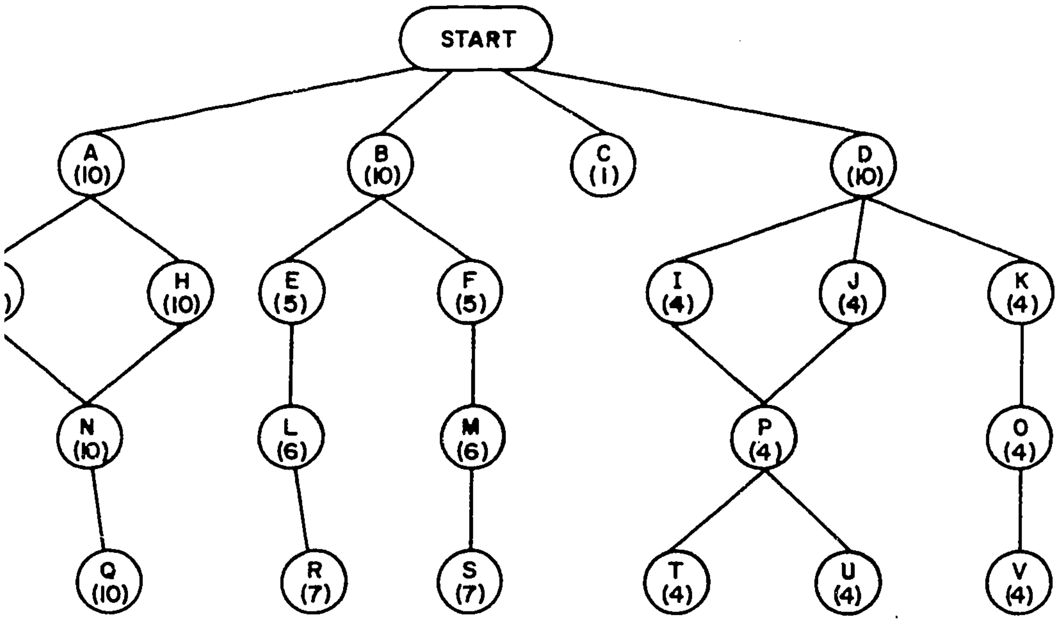


Figure 3. A Curriculum Tree with Rank Assignments

When the student finishes his rank seven lessons, his choice list
once again consist of lessons A and D, since ten is now the lowest

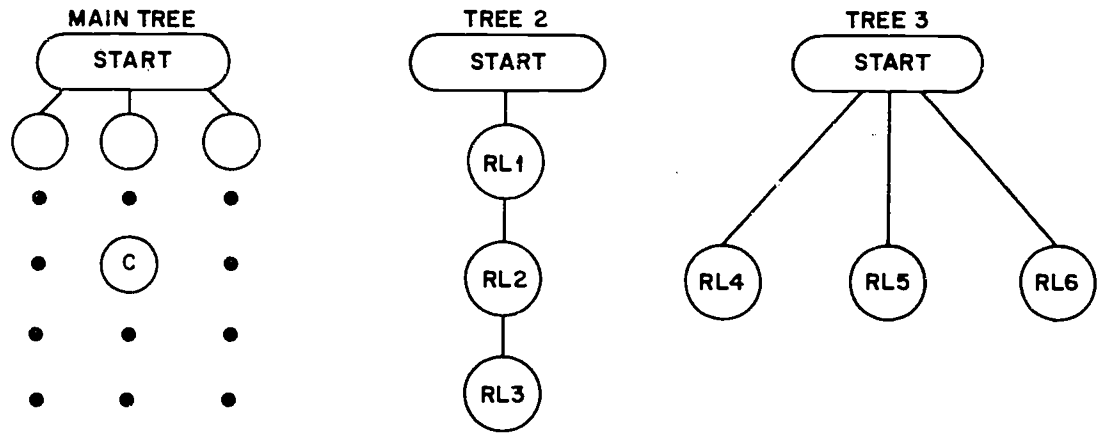
rank of any available lesson. If the student finishes Lesson A first, then the successors of A, lessons G and H, will be added to the student's choice list, which would then include lessons G, H, and D.

However, now suppose that the student finishes or satisfies Lesson D, and the successors of D, lessons I, J, and K, become available. The student's choice list will then contain only lessons I, J, and K since their rank is four. As the student finishes or satisfies lessons I, J, and K, the successors of these lessons, also rank four lessons, will be added to the choice list. Finally, when the student finishes lessons T, U, V, and all the other rank four lessons, the remaining rank ten lessons, G and H, will return to the choice list.

Some of the controls I have described above could be effected using only predecessor/successor relationships. (Lesson C, the rank one lesson, could simply be made a predecessor of lessons A, B, and D thus assuring that it would be done first.) But a little examination will show that, even within a single tree, rank gives the curriculum designer capabilities that predecessor/successor relationships alone can not provide.

The most important usage of rank and priority, however, involves control over the order of the student's access to lessons *when there is more than one tree in the chapter*. To review briefly, a chapter contains one or more curriculum trees. The first, by convention, contains the lessons which all students will see; the secondary trees, when they are present, contain the lessons that only some students will see, i.e., lessons for students who fail checkups, lessons for students for whom the regular material is too easy -- in short, lessons which are only appropriate for some students, as determined by their performance. These secondary trees are not automatically "activated" like the primary tree when a student enters a chapter; instead, they are activated only when the curriculum interface tells the session selector to activate them.

It should be clear that since each tree is unconnected to all others, predecessor/successor relationships can not be used to control the order of access to lessons in two or more active trees. However, by carefully assigning values for rank to all lessons involved, the order of access can be controlled.



ATTRIBUTES

LESSON	RANK	PRIORITY
C	2	1
RL1	1	1
RL2	1	1
RL3	1	1
RL4	1	1
RL5	1	1
RL6	1	1
ALL OTHERS	> 2	———

Figure 4. A Chapter Containing Two Remedial Trees

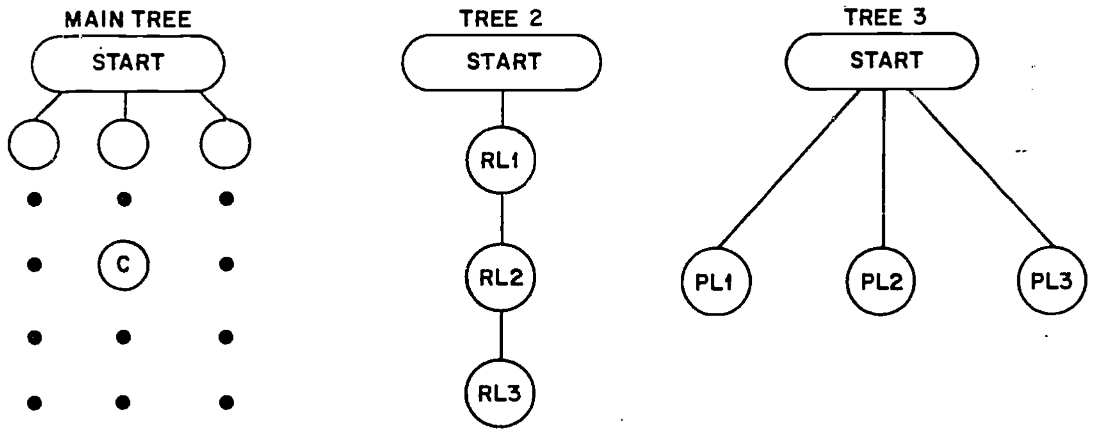
Figure 4 shows an example of a chapter containing three trees: a main tree and two trees with remedial lessons. Students begin the chapter in the main tree. At some later time, the student satisfies the predecessors of Lesson C, a checkup, and it becomes available. The next time the session selector offers lessons to the student, only Lesson C will appear on the student's choice list, since it has rank two and all the other lessons in the main tree have ranks greater than two.

If the student's performance in the checkup meets some standard, then the curriculum interface will tell the routing procedure that the checkup is finished. Otherwise, the curriculum interface will tell the routing procedure to activate Tree 2.

Since Lesson RL1, the first lesson in Tree 2, has rank one, it will now become the sole lesson on the student's choice list. When RL1 is finished, the student will then have only RL2 to work on, and similarly for RL3.

When RL3 (rank one) is finished, Lesson C (rank two), the checkup, will again appear on the student's choice list. Once again, the student's performance will determine whether the checkup is finished or whether further remediation is required in the form of Tree 3. If so, the student would then have the three lessons in Tree 3 (rank one) to choose among until he had finished them, at which time Lesson C (rank two) would reappear alone on the student's choice list. Obviously, this pattern could be repeated with further trees. The only limitations here are on the number of trees (nine) and the number of lessons in a chapter (45).

There are many variations possible on this basic scenario. Figure 5 shows a slightly revised module, with the lessons in Tree 3 (PL1, PL2, and PL3) assigned a rank of zero and a priority of five. The priority of all other lessons in the chapter is also specified as five. In this case, if the student still needed help after the Tree 2 remedial lessons, Lesson C could declare itself finished and call for Tree 3, containing PL1, PL2, and PL3, three practice lessons. Since these lessons have rank zero and the same priority as the rest of the lessons in the chapter, they will be included on all future choice lists, when situational factors permit. In this way, the student may attempt to maintain the skills which he has recently acquired. The student who passes the checkup the first time or with a higher grade may not need the practice, and the interface will not order the practice lessons for him.



ATTRIBUTES

LESSON	RANK	PRIORITY
C	2	1
RL1	1	1
RL2	1	1
RL3	1	1
PL1	0	5
PL2	0	5
PL3	0	5
ALL OTHERS	>2	5

Figure 5. A Chapter Containing One Remedial and One Practice Tree

Group IV

Frequency. The vast majority of lessons in our curricula are not designed to be finished by the student in one sitting. Rather, they are designed to be seen a little at a time, over several sessions. Frequency is defined as the number of sessions the student must wait between the session he chooses a lesson and the next session the session selector may offer that lesson to him. A frequency of three, for example, means that on the third session after the student last chose the lesson, he will be eligible to see it again. A frequency of zero means the lesson may appear on the student's choice list immediately after he returns from it.

Note that a frequency of "n" does not guarantee that a student will see a lesson every "n" sessions; however, it is a guarantee that the student will see the lesson no sooner than "n" sessions.

Delay. Just as one might desire a delay between repeated executions of a lesson (and use frequency for this purpose), it is possible that one might wish a minimum number of sessions to pass between the time a lesson's predecessors are satisfied and the first session that that lesson may be offered to the student. This is particularly useful where a lesson has only one predecessor, and it is desired that the student not do the predecessor and its successor in the same session (due to their similarity, perhaps). Delay is therefore defined as the number of sessions between the session a lesson's predecessors are satisfied and the first session the session selector may offer this lesson to the student. A delay of zero indicates the lesson may be offered immediately after the predecessors are satisfied. As with frequency, the number of sessions specified is a minimum number, guaranteeing that the student will see the lesson no sooner than the specified number of sessions after the lesson's predecessors have been satisfied.

If a student is ineligible for a lesson due to its frequency or delay, we say the student is *waiting* for the lesson.

Forced. Occasionally, a student will avoid choosing a lesson, even when it is offered to him session after session. The reasons for this may be justified from the student's point of view, but it is possible that this avoidance, if allowed to continue, will prevent the student from making progress in the curriculum.

The forced attribute tells the session selector if and when the lesson should be forced on the student. It is defined as the number of sessions the lesson may be offered to the student without being chosen before the session selector forces the student to go the lesson. If a lesson has a forced attribute equal to ten, then if the lesson has been offered for ten sessions and the student has not chosen it, the session selector will force the student to go to it in his next session in that chapter. A value of zero for this attribute means the lesson should never be forced.

"Forcing" a lesson is implemented by setting the lesson's priority value to one until the lesson is chosen. The effect of this is to exclude alternative lesson choices and lower priority lessons from the student's choices, thus forcing him to choose the lesson he has been avoiding. The lesson also plays a part in this "forcing" by turning off the student's normal prerogative to exit from the lesson at his discretion. Here, the lesson will decide when the student has done enough, and then allow him to leave.

A few general remarks can be made about all the Group IV attributes. We have used sessions as the unit of these attributes. For the student studying a single curriculum, the word sessions may be replaced by "days" since generally students are only allowed one session per day. This substitution may make the uses of these attributes somewhat clearer to those whose unit of teaching is a day.

The situation is not so simple for those students studying more than one curriculum. Here, the sessions referred to are sessions in the current chapter of a curriculum. Sessions which intervene in other curricula are not included in the count of sessions. Because a student in several curricula may be assigned to a curriculum from one to four days a week, the relationship between sessions and days no longer holds.

VI. THE DETERMINATION OF ELIGIBLE LESSONS

Exactly how the session selector determines which lessons the student is eligible for in Section III of the session can now be described. This process may be conceptualized as occurring in four stages, although the actual coding combines stages one and two into one pass, and stages three and four into another. Each stage is described in a decision table, Tables 3, 4, 5, and 6. The stages receive a list of lessons as input. By their decision rules, the stages eliminate some lessons from the list. The list is then passed as input to the next stage. The process terminates if the list is empty at the end of a stage. An empty list signals the session selector to begin the next slot or section of the session.

The input to the first stage is the list of all lessons in the chapter. Lessons are eliminated if they do not belong in the current slot, if their predecessors are not done, or if they have already been finished. Please refer to Table 3.

Table 3. Stage I Decision Table

Conditions / Rules	1	2	3	4
1. Does the lesson belong in the current slot?	N	Y	Y	Y
2. Are the lesson's predecessors satisfied or finished?	-	N	Y	Y
3. Is the lesson finished?	-	-	Y	N
Actions				
1. Include the lesson for next stage input.	N	N	N	Y

Note: All lessons in the chapter are input to Stage I.

The second stage finds the lowest non-zero rank in its input list. All lessons of this rank and all zero rank lessons are included for the third stage input. Please refer to Table 4.

Table 4. Stage II Decision Table

Conditions	Rules	1	2	3
1. Does the lesson's rank = "r"?		Y	N	N
2. Does the lesson's rank = 0?		-	Y	N
Actions				
1. Include the lesson for next stage input.		Y	Y	N

Note: Determine the lowest non-zero rank of the lessons in the input list, and call that "r."

The third stage removes lessons which can not be offered because the student must wait for them, as a result of their frequency or delay. It also eliminates lessons which PLATO staff have indicated are temporarily not operational. Furthermore, a lesson will be eliminated if the student's remaining session time is insufficient for presentation of that lesson. Please refer to Table 5.

Table 5. Stage III Decision Table

Conditions	Rules	1	2	3	4
1. Is the student waiting for that lesson?		Y	N	N	N
2. Is the lesson inhibited by PLATO staff?		-	Y	N	N
3. Is there enough session time left for the lesson?		-	-	Y	N
Actions					
1. Include the lesson for next stage input.		N	N	Y	N

The fourth and final stage of the algorithm prunes the list by choosing the lessons with the lowest priority value which can pass the following tests. The student's expected time in a lesson must not exceed the time left in the current slot. Also, the lesson must already be in the computer memory or there must be room in memory[5] for the lesson. (The slot time test is skipped if the priority of the lesson is zero.) Please refer to Table 6.

Table 6. Stage IV Decision Table

Conditions	Rules 1	2	3	4	5
1. Is "p" = 0 ?	Y	Y	N	N	N
2. Is there enough slot time left for the lesson?	-	-	Y	-	N
3. Is the lesson in memory OR is there room for the lesson in memory?	Y	N	Y	N	Y
Actions					
1. Include the lesson as output from this stage.	Y	N	Y	N	N

Note: *Find the lowest priority value of the lessons in the input list; call that "p." Test the Stage IV conditions for each lesson of priority "p." If the output list has a non-zero length, this stage terminates. Otherwise, determine the next higher priority value, say "q," set "p" to "q," and retest the Stage IV conditions. Continue testing higher priority values until the output list is non-empty, or the list is empty and there is no higher priority value in the input list.*

At this point, the decision process is complete. The list of lessons and their priority value is examined by the session selector. If the priority value of the lessons is two or greater, the entire list is offered to the student, along with the appropriate alternative lesson choices. If the priority of the list is one, it is offered without alternative lesson choices. If the priority of the list is zero, the student is sent without choice to the first lesson in the list.

VII. SUMMARY

MACSYS is an automated curriculum system designed to deliver curricular material in accordance with the classroom teacher's wishes but without the need for more than a single interaction on the part of the teacher. Instruction is delivered by MACSYS in thirty minute sessions which are structured to provide familiar material at the start of the session, ongoing curriculum in the middle, and games at the end.

MACSYS offers a choice of three curricula to the teacher: Graphing, Whole Numbers, and Fractions. A curriculum consists of a linear list of chapters. A chapter contains one or more curriculum trees. Students progress down a curriculum tree by satisfying or finishing lessons, as determined by the curriculum interface on the basis of student performance.

A lesson becomes available to the student if he has satisfied or finished all its predecessors. Students choose their lessons from among the available lessons. Which available lessons the session selector offers to the student depends primarily on the attributes of the available lessons.

An attribute is a characteristic of a lesson. Seven attributes enter directly into the determination of which available lessons the student is eligible for. These include the time the lesson is expected to take, the slot of the session it belongs in, the rank and priority of the lesson, the minimum frequency between lesson executions, the delay before initial execution, and the maximum length of time a student may go without choosing the lesson.

VIII. NOTES

1. For those familiar with the PLATO system, "session selector" is simply the name we have given to our router lesson. For those unfamiliar with PLATO and router lessons, the most important characteristic of a router lesson is that students are sent there by PLATO when they sign in and each time they leave an instructional lesson; this characteristic of our session selector is shown in Figure 1.

2. The only exception to this rule is that an unfinished session from one day does not carry over to another. The first sign-in on a new day starts a new session.

3. Over 90% of all completed sessions end in 25 to 35 minutes.

4. Some confusion is certain to be introduced by the use of the word "game." Most often what we call a game is simply a lesson cast in the <u>form</u> of a game, i.e., with a game board, some kind of competition, and the element of chance. Beneath this obvious form, each game is a lesson, carefully designed around some mathematical idea or theme. It is worth remarking that some lessons in "Main Slot" also use the game form, but we do not consider these as games since they are introducing new material. Lessons in "Game Slot" only include mathematical ideas that the student should already know.

5. At the present time, the PLATO system does not have enough computer memory to allow each student to study a different lesson at the same time. Thus a shortage of memory sometimes occurs during peak usage periods, if there is not enough "sharing" of lessons. The consideration of whether there is enough memory space for a lesson is therefore a pragmatic one and not an integral feature of the determination of eligible lessons.

In general, MACSYS components and instructional lessons use far more computer resources than the suggested PLATO system maxima -- up to twice as much computing time, nearly three times as many disk accesses, and two to three times as much memory space. While these figures can be reduced some through various optimizing procedures, the fact is that the uses to which MACSYS puts PLATO are more complex than most of its other educational uses. However, since not every user group uses all of the resources it is entitled to all of the time, relatively "expensive" uses such as MACSYS can be supported by the PLATO system, which distributes resources according to need whenever possible.

IX. REFERENCES

Cohen, Donald and Gerald Glynn. "Description of Graphing Strand Lessons." CERL Report, June 1974.

Davis, Robert B. "Mathematics Session Structure." Unpublished manuscript, 1973.

Davis, Robert B. "Observing Children's Mathematical Behavior as a Foundation for Curriculum Planning." The Journal of Children's Mathematical Behavior, I (Winter, 1971-72), 7-60.

Davis, Robert B. "What Classroom Role Should the PLATO Computer System Play?" AFIPS Conference Proceedings, XLIII, 1974, 169-173.

Dugdale, Sharon and David Kibbey. "The Fractions Curriculum: PLATO Elementary School Mathematics Project." CERL Report, March 1975.

Risken, John and Ed Webber. "A Computer-based Curriculum Management System." Educational Technology, XIV (September 1974), 38-41.

Seiler, Bonnie Anderson and Charles S. Weaver. "Description of PLATO Whole Number Arithmetic Lessons." CERL Report, July 1976.