The general problem of how to sequence subject matter content has never been satisfactorily resolved. Thus, whereas many teachers are convinced that when content is presented is just as important as what is presented, research on the issue has been equivocal. Most earlier studies of instructional sequencing have been either conflicting or inconclusive (e.g., Payne, Krathwohl, Gordon, 1967; Roe, 1962). It is not even clear from such research that sequence really makes a difference, and if so under what circumstances and to what effect.

Theoretically based approaches to sequencing typically fall into two general categories. One class of approaches has been based on the structure of the discipline (e.g., Gagné, 1970; Benick, Wang, & Kaplan, 1973; Heimer & Lotties, 1973). In this case, content is normally sequenced according to analytically determined hierarchical relationships among various "concepts". The word concept is placed in quotes because the technical use of the term in psychology is different from the broader intuitive sense implied here.) A major problem with hierarchies (or hierarchies, see Chapter 15) is that they do not take into account the fact that some people can skip levels whereas others cannot. As a consequence, approaches to sequencing based on simple hierarchies have not proved entirely successful.

A second class of approaches is less closely related to that proposed here and is based largely on learner characteristics (estimated from response histories). Although based for

1The first half of this chapter is based on Scandura (1977) and a research proposal of his to ARPA. Section 2 is based largely on the first author's (1975) dissertation and Wulfeck (1977), and was prepared jointly. The research was supported in part by a Dissertation Year Scholarship from the University of Pennsylvania, in part by a grant from the Office of Computing Services, University of Pennsylvania, and in part by a grant from the National Institute of Health (5185) to the second author.
the most part on simple learning models (e.g., Markov models), such investigations have frequently proven to be successful in dealing with such things as learning German vocabulary (e.g., Atkinson, 1972). At the present time, however, it is almost unclear whether the learning models on which they are based will prove adequate for complex learning and problem solving where structure plays a more important role.

Any really adequate approach to sequencing and instruction optimization, in our opinion, almost certainly will require attention to content structure, (general) cognitive processes, and individual differences, as well as to their interrelationships (e.g., including the assignment of values to objectives and costs to instruction, cf. Atkinson, 1972). Clearly, an approach that satisfies these requirements could have both theoretical and practical value. In particular, we believe that the incorporation of higher-order rules into training programs will require more sophisticated and flexible sequencing methodologies.

1. STRUCTURAL APPROACH TO CONTENT ANALYSIS AND TO INSTRUCTION SEQUENCING

With the exception of highly restricted task domains, it is impossible to teach students directly everything they need to know. In fact, contemporary training programs have usually been developed and carried out without even an adequate specification of just what it is that must be learned. The problem basically is that there is no general consensus on exactly how to represent the competence underlying given task domains (content), and even less on how to go about identifying such competence.

As an alternative to static relational nets, and to independent competencies associated with discrete behavioral objectives, competence in the structural learning theory is represented in terms of sets of processes (rules), which interact in prescribed, well defined ways (see Chapters 2 and 15). Among other things, structural analysis places considerable emphasis on (higher-order) rules/processes that act on other rules/processes. In effect, heuristics and other higher-level processes play a central role in the theory. Such processes should, in our opinion, play a more important role in instruction than is presently the case.2

2Although higher-order processes may have broad generality, it would be a mistake to think that any process, higher- or lower-order, is completely independent of content. Logical rules of inference, for example, perhaps as close as any to universality of application but, psychologically speaking, even they have restricted scope. In effect, the question of generality of processes is always one of degree relative to some problem domain, no matter how broad it may be.
Since there are indefinitely many different content domains of potential interest (e.g., domains of knowables) a really adequate solution to the problem of higher-order rules/processes will involve more than just a way to represent them (e.g., as higher-order rules), or just illustrations of how they might be used. As emphasized in Chapter 2, any general solution to the problem must also include some systematic method for identifying the processes underlying any content domain of interest.

The problems of content analysis (higher-order rule identification) and instructional sequencing are closely related in the structural learning theory (Scandura, 1977; cf. Chapters 3 and 13, this book). Even more generally, the theory provides a basic model of the teaching-learning process, one that gives explicit attention to content structure, cognitive processes, and individual differences, all in the context of an interactive teaching-learning system. Among other things, the theory includes: (1) a way of identifying competence (i.e., rule sets consisting of higher- and lower-order rules) underlying any given content domain, and associated with a given population of trainees, (2) an explicit basis for determining individual knowledge based on such competence, with the potential of adapting instruction to individual needs, and (3) general cognitive constraints (e.g., control processes, limitations on processing capacity) which determine how individuals can and can use the knowledge they have available. In this chapter, an extension of the theory is outlined that deals with the dynamic interaction between learner and teacher in ongoing instructional situations.

11 STRUCTURAL ANALYSIS

Given a class of problems, structural analysis (for identifying underlying competence) involves: (1) selecting a representative sample of problems, (2) identifying a solution rule for solving each of the sampled tasks (these solution rules are designed to reflect the way in which prototypic subjects in a given target population might solve the sampled problems—the initial set of solution rules is denoted R), (3) identifying higher-order rules that reflect parallels among the initial solution rules, together with the relevant lower-order rules on which they operate, (4) eliminating lower-order rules made unnecessary by the higher-order rules, and/or otherwise constructing a second order rule set, containing both the higher-order rules and lower-order rules, that collectively make it possible to derive every rule in R, (5) testing and refining the result-
ing rule set on new problems, and (6) extending the rule set where necessary so that it accounts for both familiar and novel problems in the domain. (For details, see Chapter 2 and Park.)

Consider, for example, step (1): two sample problems from the domain of geometry construction problems, and step (2): their corresponding solution rules.

Sample Problem 1: Using only a straight-edge and compass, construct a point x at a given distance d from two given points A and B.

Solution Rule 1: [Set (the radius of) the compass to distance d, put the point of the compass on point A, and draw a circular arc (i.e., the "locus" of points at distance d from point A)]; [place the compass point on B and draw another circular arc]; [label the point(s) of intersection of the two circles x].

Sample Problem 2: Given a point A, a line l and a distance d, construct a circle with radius d which goes through point A and is tangent to line l.

Solution Rule 2: [Construct a circle with center at A and radius d]; [construct a locus of points at distance d from line l (i.e., parallel line at distance d from line l)]; [construct a circle with center x (the intersection of the circle and the parallel line) and radius d].

Step (3): Notice that the two solution rules have the same general structure ([set off by brackets]). Although the component rules of these solution rules differ substantially, each solution rule involves two independent "locus" constructions, with the intersection x of the two loci playing a critical role. In the first problem, x is the solution. In the second problem, it is the center of the desired goal circle. In effect, both solution rules can be derived by applying a higher-order "two
locus" rule to the respective component rules. Roughly speaking (this can all be made quite precise—see Chapter 3), a higher-order two-locus rule operates on simple locus rules (e.g., for constructing circular arcs and parallel lines) and generates solution rules (i.e., combinations of the simpler locus rules). It is important to emphasize, that the two-locus higher-order rule can be used to derive solution rules for a wide (potentially infinite) range of problems, not just for the two sampled problems.

Step (4): Given the higher-order two-locus rule and the lower-order component rules, the solution rules themselves may be eliminated as redundant since they can be derived from the former rules acting collectively. Illustrating Steps (5) and (6) of structural analysis here would require more space than would be desirable, but the general intent is clear. More detail is given in Chapter 3.

It should perhaps be emphasized that structural analysis has been applied to several rather complex content areas (see below). Moreover, structural analysis may be applied recursively. Given an initial set of solution rules, one need not stop by deriving a more basic set (e.g., a set including higher- and lower-order rules). The derived rule set, in turn, can be subjected to precisely the same type of analysis with the result being a rule set that is still more basic. In general, structural analysis may be reapplied as many times as desired, each time yielding a rule set which is more basic in two senses: (1) individual rules tend to become simpler and (2) the new rule set as a whole has greater generating power. This, with the result that it provides a basis for solving a greater variety of tasks (see Scandura, 1973, pp. 114-117). Ultimately, reaplication of structural analysis yields rules that are atomic with respect to (all) individual learners.

1.2 INSTRUCTIONAL SEQUENCING

Structural analysis, as described above, is a crucial first step in dealing with the problem of instructional sequencing. Let us assume, for example, that we have applied structural analysis recursively to an initial set of solution rules, denoted by \( R \).

Once such a sequence of rule sets \( R_1, R_2, \ldots, R_m \) has been generated, the knowledge available to individual students can be determined by the methods described in Chapter 2 (and, in part, empirically tested in Chapters 8 and 9). In particular, sequential testing can be used with respect to the various hierarchically related rule sets and paths (in individual rules).

Learning may be assumed to take place as the learner interacts with the teaching environment according to the proposed control mechanism (see Chapters 2, 5, and 6). Since this mechanism makes no provision for processing capacity, however, reasonable limits must be placed on goal switching. Ideally, according to the present theory this should be done in accord-
ance with computed memory loads (Scandura, 1973; Chapters 2 and 7, this book) but other approximations may be more practical in real world applications. In the research described in Section 2, for example, only a fixed number of levels of derivation is allowed.

In line with the above let the final result of structural analysis be a basic rule set $B (R_m)$. Without loss of generality, $B$ can be thought of as the knowledge available to some subject on entering a course. Suppose $B$ includes, for example, such simple rules as: setting a compass to a given radius, using a fixed compass to draw an arc, using a straight-edge to draw a line segment and, perhaps, a higher-order rule that applies to pairs of available rules and forms their composition.

Given such a set, it is possible to determine by algorithmic means (e.g., by computer) whether or not given problems might be solved by applying available rules to other available rules and correspondingly which rules might be learned (derived) as a result. Because of cognitive limitations on the learner (the number of levels of derivation that he can successfully execute, processing limitations, time), only certain problems may be solved at each stage of learning. Thus, for example, whereas the first sample problem (of finding a point at a given distance from two given points) might be solved using the rules in $B$, the second sample problem would almost certainly be too complex.

We can denote the set of rules that might be learned by a person who knows exactly the rules in $B$ by $B^2$. In general, set $B^2$ will include the original rules in $B$ plus those rules that may be derived from them directly (i.e., within the operating cognitive constraints). In general then, set $B^2$ will include rules that are more complex, including more complex higher-order rules, than those that are contained in the initial basic rule set $B$. In turn, the rules $B^n$ can be used to solve more complex problems. In general, we can think of learning progressing from stage to stage. Each time learning occurs (as a result of problem solving) new and more powerful rules (which can be used to solve new and more complex problems) are added to the subject's knowledge. In general, $B^n$ may be used to denote the rules learnable directly given the rules in $B^{n-1}$.

Ordinarily, $B^n$ will be a far more encompassing (powerful) rule set than the initial sample set $R$ from which it ultimately is derived. It is this feature which allows for "creative" problem solving potential.

What does all of this have to do with sequencing? To understand this, notice that the ability to solve problems associated with $B^n$ comes about gradually as a result of solving sequences of simpler problems associated with $B$. (In effect, each rule in $B^n$ represents a unit of knowledge that might be acquired as a result of solving problems; by a subject who on entering a curriculum knows only the rules in $B$. Given any random selection of problems from the problem set...}
(e.g., geometry construction problems), it is possible to determine algorithmically which of the problems might be learned at any given stage and which problems require further instruction (e.g., in the form of prior problem-solving experience). In turn, this makes it possible to algorithmically arrange the problems in a learnable order(s).

In general, of course, it would be impossible and/or impractical to teach directly all of the solution rules contained in a. With any real non-trivial content domain, they would almost certainly be too large in number.

In order to talk about the optimization of instruction, two additional things must be done. First, educational goals must be identified, and (relative) values assigned to them. Since testing is based on underlying rules (and not goals), however, it would not be sufficient to simply assign values to educational goals. In general, achievement of an educational goal does not specify what rules are learned. Consequently, philosophically determined values assigned to educational goals, must be convertible into numerical values for rules associated with these goals. (Note: Such rules correspond to those that would be acquired by students who have mastered the curriculum. Compare this situation to that in structural analysis where the desired (atomic) rules correspond to entering capabilities that are tailored to the weakest students.)

In general, a rule will be valued just to the extent that it is involved in accounting for tasks associated with the corresponding educational goal(s). One problem for research will be to determine suitable ways of measuring degree of involvement. The requirement, for example, might be that a rule enter directly into the solution of tasks associated with some educational goal. In this case, one would rule out subordinate or source rules that are only indirectly involved. Values need not, however, be assigned to particular solution rules or particular higher-order rules, but may be assigned to classes of same (e.g., including relational nets/structures; see Chapters 2 and 15).

Specifically, at one extreme, relatively high weight might be placed on especially important rules (as is often the case in technical training). At the other extreme, classes of higher-order rules, which correspond to complex structures and thereby figure broad transfer potential, might be given the highest weights.

The second major addition concerns the various costs of instruction. The time required to teach various types of rules, various types of instruction, would provide one natural measure. In general, the cost (e.g., time, money) of testing instruction will vary directly with task complexity relative to what the individual knows at the time. Hence, any really adequate measure will have to deal with relative complexity; complexity in the abstract can be no more than normative.

Given (4), a rule set / which represents the knowledge...
available to a learner on entering a course, (2) constraints on student behavior (e.g., number of levels of allowed goal switching), (3) the values assigned to the rules in $B^N$, and (4) the time "costs" of (experience in solving) various problems and/or of expository rule instruction (which in principle can be evaluated relative to what each learner knows at each point in the instruction), it is possible to determine the total cost and instructional value associated with any given sequence of instruction. Moreover, the number of possible instructional sequences will necessarily be finite. In principle, therefore, associated values and costs can all be determined and compared to determine sequences that are optimal in some desired sense (e.g., maximum value/unit cost, minimum cost/fixed value).

As a very simple example, suppose $B = \{r_{ab}, r_{pc}, r_{ad}, \ldots\}$ where $r_{ab}$ is a rule for converting from measure $a$ (e.g., yards) to measure $b$ (e.g., feet); $r_{pc}$ from $b$ to $c$; $r_{cd}$ from $c$ to $d$; and $*$ is a higher-order composition rule that operates on pairs of compatible rules (e.g., $r_{ab}$ and $r_{pc}$) to generate their composites (i.e., $r_{ac}$, a rule that converts $a$ to $c$).

If a subject is presented first with the task of converting a given measure $a$ to the appropriate number of $d$ units, and second the task of converting from $a$ to $c$, the learner will (according to the theory) fail on the first task and succeed on the second. According to the theory, the solution to the first problem (i.e., $a$ to $d$) requires composing all three component rules simultaneously whereas the available higher-order rule can compose only two at a time. The subject would succeed on the second task, of course, because it requires composing only a pair of rules. In the process, the subject would acquire a rule for solving any $a$-to-$c$ task. However, if the learner is first presented with the $a$-to-$c$ task, and then the $a$-to-$d$ task, he will succeed on both and in the process learn one rule for converting from $a$ to $c$, and another for converting from $a$ to $d$. (Once $r_{ac}$ is learned, $r_{ad}$ can be generated by applying * to $r_{ac}$ and $r_{cd}$.) If each learned rule is given a value of one and the time cost for each task is assumed to be 5 minutes, then while the time required for each sequence would be 10 minutes, the educational value of the second instructional sequence would be twice the first.

Although direct comparison of alternative sequences is possible in principle (where reasonable assumptions can be made as to the entering capabilities of learners or where the cost of testing is either known or minimal), such comparison will rarely be feasible via brute force evaluation of all possible sequences. Practical considerations will ordinarily require the use of more efficient, heuristic methods.

(Note: In Wulfeck's dissertation (summarized in Section

5In actuality, the postulated rules may be used collectively (by the control mechanism assumed in the theory (Chapter 5).
which was an important first step in this direction, the sequencing problem was limited to rearranging random sequences of tasks into learnable sequences [i.e., sequences of tasks that can be solved in turn according to the postulated mechanisms]. In this case, it was possible to determine learnable sequences by strictly algorithmic means.)

In general, optimization will involve a far more complex balancing of gains versus costs. The case of testing versus instruction is particularly interesting, because it poses qualitatively different problems. The theory provides a strictly deterministic account of what can be learned only where sufficient information is available (via testing) concerning the learner's available knowledge. On the other hand, expending the costs necessary to get sufficient information could be counterproductive. In some cases, the teacher might do better by proceeding on the basis of partial information. Thus, ideally, the automated teacher (it is hard to imagine a human being with this much flexibility) must continually decide between testing and instruction. Does the information gained (about the learner's knowledge) via testing, and the subsequent prediction (via the theory) that this testing makes possible, justify the costs, when compared with the value that might be gained through instruction?

In the case of partial information, the teacher would necessarily have to content himself with nondeterministic judgements (cf. Chapter 8 of Scandura, 1973), or with probabilistic approximations (e.g., Hilke, Kempf, & Scandura, 1977).

To summarize the current situation briefly, the area of optimization poses a veritable gold mine of unanswered and, in many cases, unexplored problems. For example, a major open question of great theoretical and potential practical importance concerns the conditions under which local optimization will ensure overall optimization.

APPLICATION OF THE THEORY

This section describes an empirical study concerned with the problem of task sequencing. In this study: (1) The method of structural analysis was applied recursively to the geometry construction analysis described in Chapter 3. The resulting rule set was taken as basic (i.e., it was assumed to contain only rules that are atomic or correspond to entering capabilities available to all subjects in the target population). (2) A computer program was devised and implemented which takes as input arbitrary (random) sequences of geometry construction problems and arranges the problems in a (hypothesized) learnable sequence. In order to accomplish this, the problems and basic rules were represented in a form suitable for coding in SIMOOL.
and learning was assumed to take place according to a programmed version of the "first approximation" control mechanism described in Chapters 2, 5, and 6. (3) In order to evaluate the learnability of the derived problem sequences, four groups of subjects were asked to solve the problems. Two groups received the problems in the theoretically derived order. In a third group, the problems were presented in random order. Sequencing in the fourth group was learner-controlled; individuals selected problems as they saw fit.

2.1 EXTENDED COMPETENCE ANALYSIS OF THE GEOMETRY CONSTRUCTION PROBLEMS

In extending the geometry construction analysis of Chapter 3, the method of structural analysis was applied recursively. Analysis continued until the rules identified could reasonably be assumed to be atomic or uniformly available to (essentially) all students working at the seventh-grade level. The identified basic rules included: (1) lower-order rules for constructing a circle (C) given the center point (P) and radius (D) (i.e., a preset compass), denoted C(P,D); (2) drawing a line (L) or segment (S) through (between) two given points (P, P'), denoted L(P,P') or S(P,P'); (3) finding points of intersection (PX) of intersecting lines (L) or circles (C), denoted P(L, L'), P(C, C'), or simply PX; (4) measuring (setting a compass to) the distance (D) between given points (P, P'), denoted D(P,P'); and (5) choosing arbitrary points (P) or distances (D). Also included were higher-order rules for concatenating pairs of simple lower-order rules in which the output of one was a required input of the second (composition), and combining pairs of independent rules to form a single rule having the same effect (conjunction). Letting \( Y(X), Z(Y \times Y') \), and \( Y'(X') \) be lower-order rules, these higher-order rules can be represented as follows.

\[
\text{Composition: } Y(X), Z(Y \times Y') \rightarrow Z(Y(X) \times Y') \\
\text{Conjunction: } Y(X), Y'(X') \rightarrow Y \times Y'(X \times X')
\]

Since the essentials of structural analysis have already been illustrated, we shall present here a few specific examples to help the reader better understand the method we used. For illustrative purposes, consider the following geometry construction problems.

1. Given point P, line L, and distance D, construct a circle with radius D which passes through P and is tangent to L, and

2. Given triangle ABC, construct (circumscribe) a circle which passes through points A, B, and C.

In the analysis of Chapter 3, solution rules for these and other tasks were specified, parallels were identified and a technique...
higher-order rule was constructed to reflect these parallels. Solution rules for Problems 1 and 2, as well as other two-loci problems, could be derived by applying the two-loci higher-order rule to the following lower-order rules.

R1: Construct a circle (C) with a given point (P) as center, and a given distance (D) as radius. Denote it C(P,D).

R2: Construct a line (L') parallel to a given line (L) at a given distance (D) from the given line. Denote it L'(L,D).

R3: Construct the locus of points (a line, L') equidistant from two given points (P,P'). Denote it L'(P,P').

R4: Construct a circle (C), with a given point (P) as center, which passes through another given point (P'). Denote it C(P,D[P,P']).

The above higher-order rule (essentially) concatenates pairs of rules, each of which constructs a locus, and then adds a rule for constructing the goal figure from an intersection point (PX) of the two-loci. The solution rules generated for Problems 1 and 2 can be represented, respectively, in terms of the following sequences of component rules: C(P,D), L'(L,D), C(PX,D) and L'(A,B), L'(B,C), C(PX,D(PX,A)).

Rule R1 above is basic enough so that it need not be further analyzed. However, the others (including the higher-order rule) can be further analyzed. For example, Rule R4 can be generated by applying a simple composition higher-order rule to the following lower-order rules:

R5: Set a compass to the distance between two given points. Denote it D(P,P').

R6: Construct a circle given a center point and a preset compass. Denote it C(P,D).

Thus, application of the higher-order rule generates solution rule R5, R6, which is equivalent to R4.

The two-loci higher-order rule also can be further analyzed. This rule (essentially) involves forming the conjunction of two rules and, then, concatenating the conjunction rule with a goal figure rule. In effect, the higher-order two-loci rule can be generated by concatenating higher-order conjunction and concatenation rules. To complete the analysis, the above ideas are reapplied both recursively and exhaustively as required.

[For details, see Wulfeck, 1975.]

From the above, it might appear that learnable sequences of problems might be determined by simply tracing backward through the analysis that yielded the basic rule set. However, this strategy is insufficient because there are many other learnable sequences. More importantly, there also are learnable sequences that include problems not even considered during the analysis, but that are solvable via the identified rules. There is only a small probability, for example, that the problems explicitly selected (e.g., by a teacher) will be the same as those determined via structural analysis.
It is, of course, possible to enumerate all problems for which solution rules are generable from any given set of rules; one simply lets the higher-order and lower-order rules interact in all possible ways according to the hypothesized control mechanism (cf. Chapter 2). However, given a non-trivial initial rule set, such a procedure would rapidly become unmanageable in practice. The point is that structural analysis in general may not yield strict hierarchies from which it is possible to "read" all learnable sequences. Instead, one gets an incomplete lattice of rules/tasks that, generally speaking, includes only a small portion of the possible (learnable) sequences. Given any set of problems, then, our basic task was to develop a systematic way to arrange them in a learnable order.

2.2 COMPUTER IMPLEMENTATION

SNOBOL 4 was used in the computer implementation because of its powerful pattern matching capability, which was essential in testing rules, and because newly constructed strings of characters can be added to, compiled, and executed by the program itself during its own operation. The latter feature made it possible to derive new rules during the course of problem solving and, then, to turn around and use them without reprogramming.

The learning control mechanism lies at the core of the computer implementation. Specifically, the mechanism depicted in Figure 1 was implemented in SNOBOL 4 so as to parallel the "first approximation" mechanism, as described, for example, in Chapter 5.

Given a problem represented in a suitable format (as a string) this portion of the program: (Step 1) checks each available rule to see if the problem goal is contained in its range and if there are given elements and/or available rules in the domain of the rule. If not, (Step 2) control moves to the next level goal (provided a preset goal limit has not been reached). That is, the current goal level (CG) is increased by one (CG + 1) and control returns to Step 1.

If the conditions of Step 1 are satisfied, (Steps 3) the rule is applied to the given elements. Where the output is a new rule it is added to the available set of rules, the current goal level (CG) is decreased by one (CG - 1) and control returns to Step 1. The problem is solved whenever a new output is generated and the current goal is equal to one. (In the context of the program, this is equivalent to testing a potential solution against the problem goal.) The program fails to solve the problem whenever the current goal exceeds the preset goal limit (GLIMIT).

Notice that limits on processing capacity are imposed directly (by a preset goal limit). At some point we hope to extend the implementation to deal with processing capacity more directly as well as to generalize the mechanism itself along the lines described in Chapter 2.
Figure 1. Problem solving and learning mechanism.

The overall operation of the program is described in Figure 1. The program takes as input initial sets of rules and arbitrary lists of problems represented in a suitable format. (In the program both kinds of input were represented similarly with domains and ranges of the rules being interpreted as patterns and the goal and givens of problems, as objects to be matched.) The set of geometry construction rules, the only rule set considered, was the only part of the program specific to a particular problem domain. The program attempts the given problems in turn. Solved problems are added to a (learnable) sequence and rules derived while solving them are added to the rule set. Failed problems are retained on a failed problem list, and reattempted after all problems have been tried once. The process continues until all problems are solved, or until the number of failed problems reaches some pre-specified failure.
Problem 9 facilitated solving Problem 20. Although affected by
the deletions, Problem 11 had an alternate solution so that
relative difficulty was unchanged; similarly, Problem 17 used
the higher-order rule derived in solving Problem 13.

Groups R (random) and L (learner-controlled) received the
original 20 problems with the first 6 problems in the same order
as the other groups. After Problem 6, the problem order in group
R was random. In Group L, the individual subjects were allowed
to choose which (of 7 through 20) to attempt next.

Method: Subjects were run individually (under nonidentical
conditions). The problems were presented to the subjects on
separate sheets of paper one at a time (except to Group R after
Problem 6). Subjects were given a compass, straight-edge (not
eraser), and pencil and were required to show their work on
the problem sheet.

To help insure correct interpretation of the problems, each
problem statement (when presented) was read aloud by the exper-
imenters; given elements were pointed out, relational terms (such
parallel) were explained, and a sketch of the goal figure (in
required relationships to given elements) was drawn. If a prob-
lem was failed, the subject was shown a solution rule for it,
and was required to execute the rule correctly on the problem
page. To help ensure continued availability of derived solution
rules, subjects retained the problem pages and were allowed to
refer to them as desired. (In effect, differences among the
treatment groups may be attributed to differences in the higher-
order rules learned.)

Problems 1 through 6 were used to pretest subjects’ prior
knowledge. All subjects were initially given Problem 6; no
subject solved it correctly on its first presentation. Subjects
then were given Problems 1 through 6 in that order. All subjects
solved Problems 1 through 6. (These results supported our ini-
tial knowledge assumptions; the basic rules seemed to be at an
appropriate level of detail.)

Results and Discussion: Mean percent success, on problems
attempted after Problem 6, for Groups X1, X2, L, and R were
percent, 73 percent, 47 percent, and 38 percent, respectively.
All differences were significant (p < .05) except X1-X2, and
L-R. Evidently, under these experimental conditions, problem
sequences in which problem difficulty is kept relatively low,
as was the case with sequences X1 and X2 (determined by the
program), lead to significantly better performance than de-
random or learner-controlled sequences. Furthermore, subjects
must have used previously derived rules in generating solutions to
later problems. If this were not so, that is if sequence played
no role in determining success or failure on given problems,
than all groups would have performed similarly.

Mean times to solution on problems common to Groups X1 and
X2 were in complete agreement with predictions from the struc-
tural learning theory. The only significant differences in
(mean log) solution times across Groups X1 and X2 occurred in
predicted directions on Problems 9 ($X_1 < X_2$), 13 ($X_1 < X_2$), and 10 ($X_1 > X_2$). These differences are consistent with what one would expect if goal switching had been used during problem solving, and therefore provide additional support for such a mechanism. Moreover, $X_2$ subjects evidently were able to retain and use higher-order rules derived on some problems (9 and 13) on later problems (20 and 17), even though they were not given memory support with respect to higher-order rules.

As expected, there also was a significant positive relationship between (relative) problem difficulty and frequency of failure. No subject ever solved a problem where the required goal level (difficulty level) was greater than three (72 cases). Presumably, memory load approximates subjects' processing capacities at step sizes around three.

In addition, a significant interaction (on percent success) was found between problem difficulty (2 or 3) and experimental groups. Subjects in Group L, and particularly in Group R, performed differentially more poorly on difficulty-level-three problems than did $X_1$ or $X_2$ subjects. Although other factors may be involved as well (e.g., motivation), we note that the Group $X_1$ and $X_2$ sequences had a "chaining" property, such that solution rules for problems were often derived using (composite) rules derived in solving immediately preceding problems. The $L$ and $R$ problem sequences were less frequently chained in this way. Hence, memory may have had a differentially greater negative effect with the $R$ and $L$ subjects.

(Note: Chaining is only indirectly related to problem difficulty. However, when a relatively small set of problems is given to the program to be sequenced (as was the case in this study), restrictions on goal levels tend to force a moderate degree of chaining. With a larger set of problems, on the other hand, where there are many different intermediate problems between basic and terminal problems, restricting problem difficulty would tend not to force as much chaining. To the extent that rule recency, or Einstellung, is involved in subjects' processing, an even closer coordination between predictions and theory would have been obtained had limitations on processing capacity been built into the program. In this case, closer attention would have to be given to exactly what rules each subject had available at each stage of problem solving. For discussion of these issues, see Chapters 2 and 7 of this book, and Scandura, 1973, Chapter 10.)

Finally, it is worth noting that there was a fairly wide range of problem solving success over Group L subjects. Some subjects apparently had bases for choosing next problems, that were related to problem difficulty, and correspondingly to success on the problems. Three subjects stated that they chose on the basis of similarity (of problem statement and display) to previous problems. The problems they chose tended to be chained
and of low difficulty (most often level two); about 73 percent of these problems were solved. By way of contrast, two subjects chose dissimilar problems, where the relative difficulty levels were never as low as two; these subjects solved none of their chosen problems. (The remaining subjects indicated no particular basis for selection, and solved about half of the problems they chose.) These results and others (e.g., Pask & Scott, 1971) suggest that some subjects may have useful problem selection skills (rules), toward which additional research might be directed.

3. DIRECTIONS FOR FURTHER RESEARCH

In addition to the extensions previously discussed, and the need to consider other types of rule acquisition (e.g., by exposition, cf. Scandura, 1973), we would like to underline two particularly important directions for future research. As discussed in Section 1 of this chapter, (1) more consideration should be given to the problems of individualization. The balancing of gains and costs associated with testing seems to provide an especially challenging area of inquiry. In this regard, the assignment of values and costs will not be an entirely straightforward matter; both values and costs will be relative, and will depend on what individual subjects know at particular points during the course of learning. General characteristics, such as processing capacity, may also vary over subjects and a way should be found to incorporate them into a working system.

(2) The second problem, which we just mention, concerns the need to "automate" the method of structural analysis. We believe that structural analysis could play a crucial role in instructional design (as it does in much of our own work) but, as yet, the method is neither fully general nor fully objective or systematic. For one thing, it requires an analyst who is thoroughly conversant with both the content and the underlying theory. Making the method more objective and systematic would appear to be a necessary prerequisite to its widespread use.
REFERENCES


START

Load Rule Set with Domain, Range, and procedure for each initial rule.
Initialize system.
Read FLIMIT (failure limit).
Read GLIMIT (goal limit).

Are there more problems on input list? no
Are there more problems on failed problem list? no

Obtain and remove first problem on input list.
Obtain and remove first problem on failed problem list.

Execute problem solving-learning mechanism.

Fail

Is the number of failed problems > FLIMIT?

yes

Save failed problem on failed problem list.

no

STOP

Figure 2. Overall program operation.
limit. This process has the effect of reordering presented problems so that each problem (that is potentially solvable via the rule set and control assumptions) is solvable on its first presentation. (Further details are given in Wulfeck, 1975.) More generally, program outputs may be used to discard redundant problems, to rearrange problems, or to add intermediate problems so that unsolved problems become solvable.

Notice also that the highest-level goal reached in solving a problem provides a reasonable measure of the difficulty level of individual problems. Problems for which a solution rule already exists in the current rule set, for example, are solvable at the initial goal level (CG = 1 in Figure 1). Problems whose solution rules are derivable from rules in the set via an available higher-order rule are solvable at the second goal level (CG = 2), etc. This measure of difficulty summarizes (but is not perfectly or linearly related to) the amount of processing required in solving problems and provided a useful basis for predicting problem solving latencies (see Section 2.3).

The central point here, however, is that there is no fixed a priori difficulty level for any problem. Difficulty is always relative to the knowledge base—the rule set—available at the time a problem is presented. (Note: Setting the goal limit (LIMIT in Figure 1) to some whole number N restricts program output [sequences of problems] so that each problem in the sequence is solvable at CG \leq N. Sequenced problems are thus restricted to a maximum level of difficulty.)

1.3 EMPIRICAL EVALUATION

This section summarizes relevant results from an experiment conducted by Wulfeck (1975). According to the above discussion concerning relative difficulty level and amounts of processing, we would expect (with human subjects) direct relationships between difficulty levels and failure frequencies and between difficulty levels and solution latencies. In effect, problem sequences in which relative difficulty is kept small should lead to better overall performance than those in which difficulty is uncontrolled. In addition, solution latencies should increase with level of difficulty.

Four groups of 10 seventh-grade subjects each, were given different sequences of 20 geometric construction problems:

- Group X₁ received the 20 problems arranged by the computer program so that the difficulty levels for Problems 18 and 20 were three, and for all others, two. Group X₂ received a sequence obtained from the first by deleting Problems 8, 10, 12, and 16. This increased the relative difficulty levels for Problems 9 and 13 to three, and decreased the difficulty level for Problem 10 to two. (The higher-order rule derived at CG = 3 in solving

The authors wish to thank Jonathan Baron for his helpful suggestions concerning certain aspects of the design of this experiment.