Psychological Review
1963, Vol. 70, No. 6, 534–546

HUMAN ACQUISITION OF CONCEPTS FOR SEQUENTIAL PATTERNS

HERBERT A. SIMON AND KENNETH KOTOVSKY[1]

Carnegie Institute of Technology

A theory is proposed to explain how a human S produces a serial pattern from a remembered "concept" or rule, and how he acquires the concept by induction from an example of a patterned sequence. The theory consists of a formal language and a computer program, one part of which simulates the process of sequence production, the other, the process of rule acquisition. The acquired rule is represented by a symbolic structure stored in memory. Several versions of the program show varying levels of inductive power. The theory predicts rather successfully which problems, from a set of letter series completion test items, will be the more difficult for human Ss.

In most research on the acquisition of concepts, a *concept* is taken to mean a subclass of some class of objects, or, alternatively, a procedure for identifying a particular object as belonging to, or not belonging to such a subclass. The usual behavioral evidence that a subject has attained a concept is that he is able to sort objects that embody the concept from objects that do not. For example, we would say that a subject had attained the concept "red" if, on instructions to sort a pile of variously colored objects, he placed all the red objects in one pile and all the others in another.

There is no necessary relation between the ability to identify objects exemplifying a concept and the ability to produce examples of the concept. A familiar example of the lack of relation between these two abilities is the discrepancy between an individual's reading vocabulary and his speaking vocabulary (his ability to understand and his ability to produce words in a language). In experiments on memorization, the same discrepancy is familiar as the difference between ability to recognize and ability to recall.

There are some kinds of concepts, however, where we commonly measure attainment by ability to *produce* an object satisfying the concept rather than mere ability to *identify* an object as belonging to the concept. Prominent among these are concepts in the form of *serial patterns*. For example, the sequence abababa embodies the concept of "simple alternation of the characters a and b." We might test the subject's attainment of the concept by presenting him with a sequence of characters and asking him to decide whether it embodies the concept or not. More often, however (e.g., in the Thurstone Letter Series Completion Test), we ask him to demonstrate his attainment of the concept by presenting him with a sequence that embodies it, and requiring him to extrapolate the sequence. Thus, we would say that he had attained the concept, or recognized the pattern, embodied in the sequence given above if he were able to write

[1] We are grateful to the Carnegie Corporation for a research grant that assisted us in this work, and to several of our colleagues, including L. W. Gregg and K. R. Laughery, who turned our attention to the problems of serial pattern acquisition.

down ba as the next two characters in the sequence.[2]

In this paper we propose to explain in what form a human subject remembers or "stores" a serial pattern; how he produces the serial pattern from the remembered concept or rule; and how he acquires the concept or rule by induction from an example. The theory takes the form of a computer program that simulates the processes of sequence production and rule acquisition, and that creates in the computer memory symbolic structures to represent the stored concept.[3]

Three kinds of evidence will be offered in support of the theory. First, an "existence" proof is provided—it is shown that the kinds of symbolic representations and information processes postulated in the theory are sufficient to permit a mechanism endowed with them to induct, produce, and extrapolate patterns. Second, the theory is shown to be parsimonious in a certain sense—the processes and forms of representation postulated in it are basically the same as those that have previously been used to explain certain forms of problem solving, and rote learning behavior. A mechanism possessing the basic capabilities for performing these other tasks has also the capabilities for performing tasks of the kind we are considering here. Third, the predictions of the theory show good qualitative agreement with the gross behavior of human subjects in the same tasks, in particular, predicting the relative difficulty of different tasks.

The theory casts considerable light on the psychological processes involved in series completion tasks. It indicates that task difficulty is closely related to immediate memory requirements. It suggests what kinds of errors may be expected from human subjects in series completion tasks. It provides a clear-cut and operational referrent for the notion of "meaningful" as distinct from rote organization of material in memory.

CHARACTERIZATION OF SEQUENTIAL PATTERNS

In Table 1 are shown 25 Thurstone letter series completion problems. The first 10 problems, designated by the letters A through J, were used as training problems with our human subjects, the last 15 designated by the numbers 1 through 15, as test problems. The test problems vary widely in difficulty, the number of subjects in a group fo 67 who solved each problem ranging from 27 to 65.

In explaining human behavior in this problem solving task we seek, first, to form a plausible hypothesis about "what is learned": about the way in which a subject stores such patterns in memory in order to remember them, reproduce them, and extrapolate them. The first part of our theory, based on a simple "language" for characterizing serial patterns, postulates that such patterns are represented in memory by symbolic structures built from the vocabulary of such a language.

It is obvious that if a subject is able to extrapolate a sequence, he holds in memory something different from the bare sequence with which he was pre-

[2] Notice that neither in concept identification nor concept production is it essential that the concept be referred to by a name, or even have a name. We shall not consider in this paper the relation between concept naming, on the one hand, and concept identification or production, on the other.

[3] On the methodological issues involved in the use of computer programs to express and test psychological theories see Newell and Simon (1962). Laughery and Gregg (1962) and Feldman, Tonge, and Kanter (1961) have incorporated similar mechanisms for detecting and generating serial patterns in tasks somewhat different from the one considered in this paper.

TABLE 1

LETTER SERIES COMPLETION PROBLEMS

Training problems

Your task is to write the correct letter in the blank.

Read the row of letters below.

A. abababab__

The next letter in this series would be a.
Write the letter a in the blank.
Now read the next row of letters and decide what the next letter should be. Write that letter in the blank.

B. cadaeafa__

You should have written the letter g.
Now read the series of letters below and fill in each blank with a letter.

C. aabbccdd__

D. abxcdxefxghx__

E. axbyaxbyaxb__

You will now be told what your answers should have been.
Now work the following problems for practice. Write the correct letter in each blank.

F. rsrtrurvr__

G. abcdabceabcfabc__

H. mnlnknjn__

I. mnomoompom__

J. cegedeheeeiefe__

You will now be told the correct answers.

Test problems

1. cdcdcd__
2. aaabbbcccdd__
3. atbataatbat__
4. abmcdmefmghm__
5. defgefghfghi__
6. qxapxbqxa__
7. aduacuaeuabuafua__
8. mabmbcmcdm__
9. urtustuttu__
10. abyabxabwab__
11. rscdstdetuef__
12. npaoqapraqsa__
13. wxaxybyzczadab__
14. jkqrklrslmst__
15. pononmnmlmlk__

sented. The sequence, taken by itself, provides no basis for its own extrapolation. Indeed, from a strict mathematical standpoint, there is no uniquely defined correct answer to a serial pattern extrapolation task. Consider, for example, the sequence 1, 2, 3, 4, ... What is its continuation? One answer might be 5 but another, equally valid, would be 1 (i.e., 1, 2, 3, 4, 1, 2, 3, 4, 1 . . . . Still another would be 2 (i.e., 1, 2, 3, 4, 2, 4, 6, 8, 3, 6, 9, 12, . . .).

What is common to all these alternative solutions is that each is produced by a rule that is capable of continuing the sequence indefinitely. It is pragmatically true, although not logically necessary, that for the items commonly used on serial pattern tests it is easy to get consensus about the correct continuation. Presumably, the reason for this is that one sequence is sufficiently "simpler" or "more obvious" than others, that almost all persons who find an answer find that one first. But it must be emphasized that this is a psychological, not a logical, matter.

It is also not logically necessary that a given continuation be associated uniquely with a particular rule. There may be several different ways of obtaining the same continuation, or of representing a particular rule. Indeed we shall encounter some examples of such multiple possibilities as we go along.

We must begin, then, by saying something about what the subjects bring to the task—for what they bring will certainly affect their criteria of simplicity, the kinds of patterns they will discover, and how difficult it will be for them to discover them. We assume the subjects have in memory the English alphabet, and the alphabet backwards. (There are some

alternatives to the latter assumption, but we make it, at present, for simplicity.) We assume the subjects have the concept of "same" or "equal" —e.g., c is the same as c. We assume they have the concept of "next" on a list—e.g., d is next to c on the alphabet, and f to g on the backward alphabet. We assume they are able to produce a cyclical pattern—e.g., to cycle on the list a, b in order to produce abababababa. . . . Finally, we assume that they are able to keep track of a small number of symbols in immediate memory—for present purposes, we need to assume only the capacity to keep track of two symbols simultaneously. We may call these the first and second symbols in immediate memory, respectively.

Now, using a simple language capable of handling only the concepts that have just been described, representations can be constructed for all of the serial patterns in Table 1, and many others. It will be easiest to show how this is done by considering four examples of gradually increasing complexity.

Pattern 3: atbataatbat__. This sequence can be described most simply if we mark it off in periods of three-letter lengths: atb ata atb at__. Having done this, we observe that the first position in each period is occupied by an a, the second position, by a t. We refer to these patterns as *simple cycles* of a's and t's, respectively. The third position in the period is occupied by the cycle ba ba. . . . We refer to a pattern of this kind as a "cycle on the list b,a." Hence, we can describe the entire Pattern 3 by the notation:

3. [a, t, (b,a)]

Pattern 2: aaabbbcccdd__. Again, this sequence can be marked off in periods of three letters; but in this case, there are simple relations among the letters *within* each period (they are, in fact, identical). One way in which we can describe Pattern 2 is by the notation:

[M1 = Alph; a]
2. [M1, M1, M1, N(M1)]

The notation is interpreted as follows: we set a variable, M1, equal to the first letter, a, of the alphabet. Each period is executed by producing M1 three times, and then replacing M1 by the *next* (N) letter of the alphabet. An alternative representation of this pattern is shown as Example 2b in Table 2. We shall refer to it later.

Pattern 13: wxaxybyzczadab__. This sequence is more complicated than the previous two, but can again be analyzed in terms of a period of three symbols; with internal relations among the first two symbols of each period. One description of this pattern is:

[M1 = Alph; w: M2 = Alph; a]
13. [M1, N(M1), M1, M2, N(M2)]

Here are two variables, M1 and M2, corresponding to alphabetic sequences; but the M1 sequence begins with w, while the M2 sequence begins with a. Notice that when we come to the end of the list for such a sequence, we begin again at the beginning—z is followed by a. Thus, these alphabetic sequences are identical with what we have called cycles on a list; in this case, the list is the alphabet, (a . . .z).

Pattern 15: ponomnmlmlk__. Our final example, also based on a period of three, can be represented in the same notation as the others, with the addition of one new operator:

[M2 = M1 = Balph; p]
15. [M2, N(M2), M2, N(M2),
N(M1), E(M2,M1)]

TABLE 2

PATTERN DESCRIPTIONS OF THE TEST PROBLEMS

Example	Initialization	Sequence Iteration
1a.	M1 = (c, d,); c	M1, N(M1)
1b.	—	c, d
2a.	M1 = Alph; a	M1, M1, M1, N(M1)
2b.	M1 = Alph; a	M1(3), N(M1)
3.	M1 = (b, a); b	a, t, M1, N(M1)
4.	M1 = Alph; a	M1, N(M1), M1, N(M1), m
5.	M1 = M2 = Alph; d	M1, N(M1), M1, N(M1), M1, N(M1), M1, N(M2), E(M1, M2)
6a.	M1 = (q, p); q: M2 = (a, b); a	M1, N(M1), x, M2, N(M2)
6b.	—	q, x, a, p, x, b
7.	M1 = Alph; d: M2 = Balph; c	a, M1, N(M1), u, a, M2, N(M2), u
8.	M1 = Alph; a	m, M1, N(M1), M1
9.	M1 = Alph; r	u, M1, N(M1), t
10.	M1 = Balph; y	a, b, M1, N(M1)
11.	M1 = Alph; r: M2 = Alph; c	M1, N(M1), M1, M2, N(M2), M2
12.	M1 = Alph; n: M2 = Alph; p	M1, N(M1), M2, N(M2), a
13.	M1 = Alph; w: M2 = Alph; a	M1, N(M1), M1, M2, N(M2)
14.	M1 = Alph; j: M2 = Alph; q	M1, N(M1), M1, M2, N(M2), M2
15.	M1 = M2 = Balph; p	M1, N(M1), M1, N(M1), M1, N(M2), E(M1, M2)

Note.—Alph = alphabet; Balph = alphabet backwards.

There are two variables, M1 and M2, which follow the sequence of the alphabet backwards (Balph), starting with the letter p. The variable M2 is produced, then the next letter of the sequence, then the next; then the next in sequence to M1 is found, and M2 is set equal (E) to the new M1.

Table 2 gives pattern descriptions for the entire set of 15 test problems in the notation we have just introduced. We remark again that the descriptions are not necessarily unique—in many, if not all, cases, it is fairly easy to find alternative descriptions of the patterns. Those provided appear intuitively to be the simplest among the alternatives we have found. In the case of Patterns 1, 2, and 6, we give two alternatives.

The pattern descriptions contain all the information contained in the sequences from which they were derived. They can be used to reconstruct the sequences. More than that, they can be used to extrapolate the sequences indefinitely—hence they can be used to perform the task with which the subjects were confronted in the Letter Series Completion Test. Thus, we may assert that anyone who has learned the pattern description has learned the concept embodied in the corresponding sequence. Our central hypothesis about human concept attainment in situations involving serial patterns is the converse of this assertation, namely: *subjects attain a serial pattern concept by generating and fixating a pattern description of that concept.*

GENERATING SEQUENCES

We have now achieved our first objective: to formulate a simple, parsimonious language of pattern description, based on plausible hypotheses about what subjects bring to the serial pattern task. Our next tasks are (*a*) to propose a mechanism that would enable a subject, holding such a pattern description in memory, to produce and extrapolate a sequence; and (*b*) to propose a mechanism that would enable a subject to induct such pattern descriptions from segments of letter sequences. We consider first the possible structure of a sequence generator.

Information processing theories already exist that seek to explain how humans perform certain other tasks,

including problem solving—the General Problem Solver (GPS)—and rote memory—the Elementary Perceiver and Memorizer (EPAM). In constructing our present theory, we wish to avoid creating elaborate mechanisms ad hoc, and seek, instead, to build the hypothesized system from the same elementary mechanisms that have been used in GPS and EPAM. Our language of pattern description makes this easy to do, since the processes required to produce sequences fitting the list descriptions can be formulated naturally and simply in the list processing language that has been used in constructing these earlier theories.

A list processing language, as its name implies, is a system of processes for acting upon symbolic information represented in the form of lists and list structures (lists of lists). Among the fundamental processes in such a language are the process of writing or producing a symbol, the process of copying a symbol (i.e., writing a symbol that is the same as the given symbol), and the process of finding the symbol that is next to a given symbol on a list. In addition, there are processes for inserting symbols in lists, deleting symbols from lists, and otherwise modifying lists and list structures.[4]

[4] We cannot enter here into a full discussion of the reasons for supposing that human thinking processes are fundamentally list processes. For a general nontechnical introduction to this point of view see Miller, Galanter, and Pribram (1960). The particular list processing language that has been used to define GPS, EPAM, and the theory set forth in this paper is IPL-V (Information Processing Language V). The language is described in Newell (1961). Investigators who wish more detail about the program of the Sequence and Pattern Generators can obtain a program listing from the authors. The program can be run on most of the large computers that are available on university campuses, for example the Bendix G-20; IBM 704, 709, and 7090; the CDC 1604; or the Univac-Scientific 1507.

We can see rather immediately that processes of these kinds will enable the subject, having stored the pattern description, to produce and extrapolate the sequence. By way of example, let us consider in detail Pattern 9 in Table 2. To produce the sequence described by the pattern, we simply interpret the pattern description, symbol by symbol, as follows:

1. *Hold* the letter "*r*" on the list named "Alphabet" *in immediate memory.*
2. *Produce* the letter "*u.*"
3. *Produce* the letter that is in immediate memory (initially, this will be "*r*").
4. *Put the next* letter on the list in immediate memory (on the first round, this will move the pointer to "*s*").
5. *Produce* the letter "*t.*"
6. Return to Step 2, and *repeat* the sequence as often as desired.

Any mechanism that follows the program outlined in Steps 1 through 6 will produce the sequence: urtustuttuut. . . . Thus, all that is required to construct such a mechanism, is to give it the capacity to interpret the symbols in the pattern description, and to execute the actions they signify —actions like "hold in immediate memory," "produce," "find next on a list," "repeat."

The second part of our theory, then, is a program, written in IPL-V—that is capable of generating sequences from pattern descriptions by executing the elementary list processes called for by the descriptions. As we have seen, the program is extremely simple. We postulate: *normal adult beings have stored in memory a program capable of interpreting and executing descriptions of serial patterns. In its essential structure, the program is like the one we have just described.*

Our main evidence for these assertions is that the program we have written, containing the mechanisms

and processes we have described, is in fact capable of generating and extrapolating letter series from stored descriptions. We are not aware that any alternative mechanism has been hypothesized capable of doing this. Further, the basic processes incorporated in the program are processes that have already been shown to be efficacious in simulating human problem solving and memorizing behavior.

PATTERN GENERATOR

We come, finally, to the question of how subjects induct a pattern description from the pattern segment that is presented to them. Our answer to this question again takes the form of a program that is capable of doing just this, in cases where the pattern is not too complex. We shall describe the program, and then consider the reasons for supposing it bears a close family resemblance to the programs used by human subjects.

The inputs to the pattern generator are the letter sequences presented to the subject. The outputs of the generator are the corresponding pattern descriptions. By considering what is involved in translating a sequence (Table 1) into its pattern description (Table 2), we can achieve some understanding of what is involved in the generator. Basically a description characterizes a sequence in terms of some initial conditions—for example, the symbol to be stored at the outset in immediate memory—and some relations among symbols—for example, that one symbol follows another in the alphabet. The main task, then, of the pattern generator is to detect these initial conditions and relations in the given sequence, and to arrange them in the corresponding pattern.

There is gross behavioral evidence that the subjects accomplish these tasks by first discovering a periodicity in the sequence. Sequence 1, for example, has a period of two, for every other symbol is a c. Similarly, Sequence 2 has a period of three, for it consists of segments of three equal symbols each. The pattern generator seeks periodicity in the sequence by looking for a relation that repeats at regular intervals. Thus, it discovers that the *same* symbol occurs in every second position in Sequence 1, and that the *next* symbol occurs at every fourth position starting with the first in Sequence 5. If this kind of periodicity is not found, the pattern generator looks for a relation that is *interrupted* at regular intervals—Sequence 2 provides an example, where the relation of "same" is interrupted at every third position. Thus, to discover periodicity in the sequence, the pattern generator needs merely the capacity to detect relations like "same" and "next" with familiar alphabets.

Once a basic periodicity has been discovered, the details of the pattern are supplied in almost the same way—by detecting and recording the relations—of equal and next—that hold between successive symbols within a period or between symbols in corresponding positions of successive periods. The pattern of Sequence 9, for example, records that a period of three was discovered; that the first position in the period is always occupied by the same symbol—u, and the third position always by t. In the second position, however, each successive period has the symbol next in the alphabet to the second symbol in the previous period.

A number of different variants of the pattern generator have been written, all of them, however, based on these same simple relation recognizing processes. The several variants show

different degrees of success in describing the 15 test sequences. A particular pattern generator may fail to describe a given pattern for either one of several reasons. It may be unfamiliar with an alphabet used in constructing the pattern. It may not have a sufficiently wide repertoire of relations it can test. It may have inadequate means for organizing and recording as a coherent pattern description the relations it discovers. All of these reasons for failure can be identified in our experiments.

We would expect that among our human subjects, also, different levels of performance on the Letter Series Tests might be associated with the same kinds of limitations. We shall raise this point again when we look at some of the data on human performance and its comparison with the computer simulation.

Some information on the performance of four variants of the program—A, B, C, and D—is provided in Table 3. The program became progressively more powerful, Variant A solving 3 of the 15 problems; Variant B, 6; Variant C, 7; and Variant D, 13. Except for Problem 3, all problems solved by a less powerful variant were solved by the more powerful variants. There is no *logical* necessity for this ordering relation to hold, but as an empirical matter it would be rather difficult to construct a variant that would succeed on the "hard" problems and fail on the "easy" ones. Thus, the programs reveal a "natural" metric of difficulty—a point we shall discuss further in the next section.

The pattern generator—we shall not attempt to distinguish among the several variants—constitutes the third part of our theory of human serial

TABLE 3

PROBLEMS FAILED BY GROUP OF 12 SUBJECTS, AND BY VARIANTS OF THE COMPUTER PROGRAM

Problem number	Subjects[a]												Program variants			
	1	2	3	4	5	6	7	8	9	10	11	12	A	B	C	D
1																
2													X			
3				X		X								X	X	X
4													X			
5		X	X	X	X		X	X	X	X	X	X	X	X	X	
6				X			X			X	X					
7						X	X	X	X	X	X	X	X	X	X	X
8										X	X	X	X			
9			X		X			X	X	X	X	X	X			
10									X	X		X	X	X		
11					X	X	X	X	X	X	X	X	X	X	X	
12					X	X	X	X	X		X	X	X	X	X	
13				X	X	X	X			X			X	X	X	
14			X					X	X	X	X	X	X	X	X	
15				X		X	X	X	X		X	X	X	X	X	
Total correct	15	14	12	10	10	9	8	8	7	6	6	6	3	6	7	13

Note.—X = problem missed.
[a] In order of performance.

pattern learning. We postulate: *normal adult human beings have stored in memory a program, essentially like the pattern generator just described, capable of detecting relations and recording a pattern description for a simple sequence.*

EXAMINATION OF SOME EMPIRICAL DATA

Thus far we have been concerned primarily with describing a set of programs capable of doing what human subjects demonstrably can to: discover, remember, and produce simple serial patterns. We have been able to find some quite simple mechanisms, incorporating elementary symbol manipulating and list manipulating processes, that have this capacity. The next stage in inquiry is to see what light these mechanisms—hypothesized as an explanation of human performance in these tasks—can cast on the behavior of subjects in the laboratory; and conversely, to seek more positive tests of the validity of the explanation.

The data we shall discuss here were obtained by giving the Letter Series Completion Test to two sets of subjects. Since our main interest was in analyzing differences in difficulty among problems rather than differences in ability among subjects, no special care was taken to obtain samples representative of any particular population. The first group, 12 subjects, ranged from college graduates to housewives. The second group, 67 subjects, comprised an entire class of high school seniors. Problems 1–15 of Table 1 were administered to the 12 subjects individually and to the 67 subjects as a group.

The three columns of Table 4 show for each problem: (*a*) the number of subjects in the first group who solved it, (*b*) the number of subjects in the second group who solved it, and (*c*) whether it was solved (S), or left unsolved (U) by Variant C of the computer program.[5] The problems of less than median difficulty, as defined by the numbers of subjects solving them, are shown in boldface type in Columns 1 and 2.

TABLE 4

PROBLEM DIFFICULTY: COMPARISON OF HUMAN SUBJECTS WITH VARIANT C OF PROGRAM

Problem number	Number of subjects obtaining correct solution		Program[a]
	Group of 12	Group of 67	
1	12	**65**	S
2	12	**61**	S
3	10	**60**	U
4	12	**57**	S
5	2	45	U
6	8	**48**	S
7	5	27	U
8	9	**49**	S
9	5	43	S
10	9	**51**	S
11	4	39	U
12	5	42	U
13	7	43	U
14	6	**48**	U
15	5	34	U

Note.—Problems shown in boldface were below median in difficulty for subjects in question.
[a] S = solved, U = failed.

The three columns of Table 5 show: (*a*) the average time per problem for those subjects in the group of 12 who obtained the problem solution, (*b*) the average time spent by all subjects in the first group on each problem, and (*c*) the time spent per problem by Variant D of the program (which solved 13 of the 15 problems). The times that were below median in Columns 1 and 2 are shown in boldface type.

[5] Variant C was used for this comparison because it solved about half (7 out of 15) of the problems, thus permitting them to be divided evenly into "easy" and "hard."

TABLE 5

COMPARISON OF 12 SUBJECTS WITH VARIANT D OF PROGRAM: TIME PER PROBLEM

Problem number	Seconds per subject		Seconds
	Subjects who solved	All subjects	Program D
1	**6.0**	**6.0**	9
2	**3.8**	**3.8**	28
3	**24.7**	**23.0**	18[a]
4	**16.8**	**16.8**	23
5	27.5	40.6	35
6	37.0	31.4	19
7	37.8	49.2	19[a]
8	**24.9**	**24.9**	23
9	**18.8**	**28.7**	17
10	**20.9**	**20.7**	18
11	**21.5**	37.3	35
12	49.8	49.0	24
13	61.7	65.5	29
14	41.2	47.8	36
15	48.0	56.8	30

Note.—Times below median shown in boldface.
[a] Program D failed to solve problem.

Considering both tables together, we have four measures of problem difficulty for the human subjects—two measures of numbers of subjects who solved the problems, and two measures of problem solving time. Not surprisingly, there is a high level of agreement among the four measures as to which problems were easy, and which hard. On all four measures, Problems 1, 2, 3, 4, 8, and 10 ranked below the medians in difficulty, as did Problem 6 on two measures (number solving) and Problem 9 on two measures (problem solving time). Problems 14 and 11 were each below median on one measure, while Problems 5, 7, 12, 13, and 15 were above the median in difficulty on all four measures. For purposes of gross comparison, we will call the eight problems in the first two groups the easy ones, and the seven problems in the last two groups the hard ones.

To see whether there is anything in our theory that would account for these differences in difficulty, we examine the pattern descriptions in Table 2. By common sense inspection of the pattern descriptions, it is clear that the easier problems have simpler descriptions—we could have made an almost perfect prediction of which problems would be above median in difficulty simply by counting the number of symbols in their pattern descriptions. (There is some ambiguity for Problem 6, which is neither as difficult as Description 6a would suggest, nor as simple as Description 6b would imply.)

But the lengths of the descriptions do not tell the whole story. If we now examine the patterns more closely we see that *all* of the patterns for the hard problems, and *none* of the patterns for the easy ones (except 6a) call for two positions in immediate memory. To extrapolate these more difficult sequences, the subject has to keep his place in two separate lists, but only in one at most, for the easier sequences. Moreover, to build up the patterns for the former sequences, the subject had to detect and keep track of relations on two distinct lists, as against one for the latter sequences.

An alternative hypothesis would be that the length of period was the source of difficulty. It is true that all the patterns with a period of four or more symbols (Patterns 5, 7, 11, and 14) are among the hard ones; but Patterns 12, 13, and 15, which have periods of three, are hard; while Patterns 3, 4, 8, and 10, which also have periods of three, are easy. Although the evidence is far from conclusive, number of positions in immediate memory appears to be more closely related to difficulty than length of period.

We cannot undertake here a detailed analysis of the errors made by our subjects, but we can make one observation that helps explain why Problem 9 appeared rather more difficult (in terms of failure to solve, not solution time) than its pattern description would have predicted. The main process, we have hypothesized, for solving these problems is to detect relations between adjoining symbols, or symbols in corresponding positions of successive periods. But towards the end of the sequence in Problem 9—the symbols *tuttu*—there are a number of spurious relations of "equals" and "next" that are not part of the pattern. Discovery of these relations, and failure to check them through the earlier part of the sequence, would lead to wrong answers. For example, the partial sequence given above could reasonably be extrapolated by annexing the symbol t.

COMPARISONS WITH PROGRAM PERFORMANCE

We have seen that one part of our theory—the pattern descriptions—allows us to make predictions about the relative difficulty of serial pattern problems for human subjects. It may be objected that the test is subjective, since we cannot know that the patterns used by the subjects are the same as those we have written down. The objection would be more convincing if it could be shown that the patterns could be described in a manner quite different from the one we have proposed. But there is additional evidence we can bring to bear on the question, derived from the programs used to generate the patterns —the third part of our theory.

Of the several variants of the pattern generating program we have studied, Variant C will be considered here, because it solved 7 of the 15 problems, hence found about half of them "easy" and half "hard." From Table 4 it can be seen that the program solved none of the problems we have previously labeled hard, and all but one (Problem 3) of the problems previously labeled easy. Hence the pattern generator also provides excellent predictions of the relative difficulty of the problems for human subjects.

A closer investigation of the program's failure with the hard problems showed that the difficulties arose specifically in keeping track of the lists associated with distinct positions in immediate memory. The program was incapable of organizing the parts of the pattern into an overall structure when two immediate memory positions were involved. We take this as additional evidence for the plausibility of our hypothesis that this was the ocus, also, of the difficulties the lessl successful human subjects encountered. A more powerful version of the program, Variant D, overcame most of these difficulties, and failed only on Problems 3 and 7. A still more powerful version has solved all but Problem 7.

A few more words are in order about Problems 3 and 6. Problem 6 was solved by Variant C relatively rapidly, but the pattern discovered was 6b rather than 6a. With respect to Problem 3, we must simply say that the program of Variant C was different from that of most of the human subjects. (It might be mentioned, however, that the fourth and sixth ranking in the group of 12 adult subjects also missed this problem.) The occurrence of a following b in Sequence 3 led the program to attempt to use the relation of "next on the backward alphabet" instead of describing the pattern in terms of the

circular list (a, b). It did not do enough checking to discover and correct its error. The majority of the human subjects either did not make that error, or were able to correct it.

In the third column of Table 4 we have recorded the times spent by Variant D on each of the 15 problems. There is a modest positive correlation between the times taken by the program and the subjects (Column 1), but the agreement cannot be claimed to be close. Analysis suggests that the time required by the program depended much more on *length of period* than did the time required by the human subjects. If we consider only the nine patterns of Period 3, the correlation of times is very much improved. Since the theory does not postulate that the relative times required for the several elementary processes will be the same for the computer as for human subjects, there is no real justification for comparing human with computer times between tasks that have quite different "mixes" of the elementary processes.

Among the patterns of Period 3, Pattern 2 took the human subjects a very short time, but the program a rather long time. We would conjecture that in this case, the program was slow because it lacked a concept that most of the subjects had—the concept of repeating a symbol a fixed number of times (see Tables 1 and 2). Thus, while the program discovered Pattern 2a, we believe that most subjects represented the pattern in a manner more nearly resembling 2b.

We have mentioned these details because they illustrate how a theory of the sort we have proposed permits one to examine the microstructure of the data, and to develop quite specific hypotheses about the processes that human subjects use in performing these tasks. Of course, to test these hypotheses we shall require additional observations, particularly observations like those we have reported on problem solving tasks, which record not simply the success or failure of the subject, but as much detail as can be detected of behavior during the problem solving process. The possibility of confronting the theory with such detail greatly facilitates its testing and improvement.

CONCLUSION

In this paper we have set forth a theory, comprising a language for pattern description and a program, to explain the processes used by human subjects in performing the Thurstone Letter Series Completion task. We have devised measures of problem difficulty based on the pattern descriptions and upon the ability of variants of the program to solve particular problems. These measures of problem difficulty correlate well with measures derived from the behavior of the human subjects. By analysis of the pattern descriptions and programs, we have been able to form, and partially test, some hypotheses as to the main sources of problem difficulty. By detailed comparison with the human behavior, we have formed some conjectures about the detail of processes that can be subjected to additional tests in the further development of the theory.

We conclude on the basis of the evidence presented here that the theory provides a tenable explanation for the main pattern forming and pattern extrapolating processes involved in the performance of the letter series completion task. Different variants of the theory can be used to account for individual differences among human subjects in performing this task.

REFERENCES

FELDMAN, J., TONGE, F. M., JR., & KANTER, H. Empirical explorations of a hypothesis-testing model of binary choice behavior. Report No. SP-546, December 19, 1961, System Development Corporation, Santa Monica.

LAUGHERY, K. R., & GREGG, L. W. Simulation of human problem-solving behavior. *Psychometrika*, 1962, **27**, 265–282.

MILLER, G. A., GALANTER, E., & PRIBRAM, K. H. *Plans and the structure of behavior.* New York: Holt, 1960.

NEWELL, A., & SIMON, H. A. Computer simulation of human thinking. *Science*, 1962, **134**, 2011–2017.

NEWELL, A. (Ed.) *Information processing language-V manual.* Englewood Cliffs, New Jersey: Prentice-Hall, 1961.

(Received October 12, 1962)