A THEORETICAL EXPLORATION OF MECHANISMS FOR CODING THE STIMULUS

Allen Newell
April, 1972

This paper is to be published in Coding Processes in Human Memory, A. W. Melton and E. Martin (eds.), Washington, D.C., Winston (in press). It cannot be copied without the permission of the author.

Carnegie-Mellon University
Pittsburgh, Pennsylvania

ABSTRACT

This paper explores an information processing model of how stimuli are perceived and encoded. The model is an extension of recent work on human problem solving, which has yielded an explicit programming structure (a production system) as a representation of time course of human behavior in some relatively simply discrete symbolic tasks. The emphasis in the present paper is on obtaining an explicit representation of the control structure in the immediate processor and on the communication between the immediate processor and the perceptual system. The internal structure of the perceptual system is not explored in detail. The paper presents the original production system for problem solving and illustrates its structure and behavior. It then discusses the nature of stimulus encoding and what is provided by the model as it stands. This leads to the introduction of a task to guide the extension of the model. A model of the perceptual system is then presented and its behavior in conjunction with the main system illustrated.

A THEORETICAL EXPLORATION OF MECHANISMS FOR CODING THE STIMULUS*

Allen Newell

This paper explores the problem of developing an explicit model for how stimulus encoding occurs. It is primarily a theoretical exercise, attempting to extend some work in problem solving (Newell and Simon, 1972) to incorporate perceptual mechanisms and control structures to permit stimulus encoding. The set of conditions that we impose on the total model -- in terms of the sufficiency of the mechanisms and the detail of their interactions -- makes it unlikely that an initial formulation will be successful. And indeed this is the case: the model remains incomplete in a number of significant ways and we can only examine a minute part of its behavior with the confines of this paper. Thus, we have called the paper a theoretical exploration.

This work stems from the view that to study coding in human information processing requires a model of the total process -- a model that specifies exactly how coding operations take place. The general strategy in experimental psychology runs to the opposite side, namely, that one should posit a model by stating only a few general properties of the system. When well done, this leads to some implications for behavior which can then be tested. The net effect is slowly to close in on a mechanism, catching it in a conjunctive net of properties, each one established experimentally. Often the objects of most interest -- here the coding operations -- remain extraordinarilly ill specified.

* I would like to acknowledge fully the contribution to this effort of a Protocol Workshop held at CMU during Spring 1971, and especially Michelene Chase, David Klahr, Donald Waterman and Richard Young who all worked extensively on the series completion protocol discussed herein, developing production systems that were the starting point of this research. The work here is a direct continuation of joint research with H. A. Simon and draws in detail on material in (Newell and Simon, 1972). The research is supported in part by the National Institute of Mental Health (NH-07722) and in part by the Advanced Research Agency of the Office of the Secretary of Defense (F44620-70-C0107) which is monitored by the Air Force Office of Scientific Research.

Let me make the point concretely by quoting a few examples. All of these represent studies that I feel are successful and have given us both new information and provocative ideas about mechanisms. No straw men are intended.

Consider first the well known study of memory by Atkinson and Shiffrin (1968). Specific models of memory are proposed from which can be computed experimental results to be compared with extensive data. Still, I am left with an uncomfortable feeling. A central part of their story is the notion of control processes, which allow the subject to perform according to different strategies. But these control processes receive no representation in the theory. They are used informally to rationalize the application of specific models to specific situations. In some sense a specific representation of control mechanism is not needed to get on with the study. Still, it remains an incomplete paper from which I find it hard to move on.

Consider next a study by N. Johnson (1970) concerned with coding processes in memory, namely, those that lead to chunking stimuli in various ways. Again, he provides a quite specific model for part of the process, i.e., the control process for decoding a stimulus to give a response. This is enough for him to justify the relevance of his response measure and to argue for a number of effects. Still, the process he is studying -- coding and chunking -- is nowhere specified. He argues to a few properties of it, e.g., whether a code (i.e., the internal representation of a part of the stimulus) is like an opaque container. This is enough of a characterization to set up some experimental tests. But my greatest disappointment was that the paper proposed no theory of the operations of coding of verbal stimuli.

The McLean and Gregg (1967) study of induced chunking in serial learning offers an almost identical example from my point of view. It evokes a specific view of processing mechanisms and finds an ingenious way of revealing some effects of these processes in an experimental task* But what I want is a model of how the subject says the alphabet backwards, not simply that the backwards recitation can be used to reveal that the organization into chunks is really there.

One last example will suffice. Much recent work has occurred on imagery. One segment of this work is concerned with imagery as a mediator in various verbal learning tasks (e.g., Pavio, 1969; Bower, in press). It is a peculiar feature of all this work that it proposes no theory or model of imagery at all. In fact, if you ask how one knows that the mediator is imagery, rather than something else, the only link is in the semantics of the instructions to the subject (plus the experimenter's participatary conviction that imagery is involved). The problem is not the old saw about operationally. In fact, from one point of view, there is no problem at all. Strong effects are being produced and progress made. Still, if I were going to work on imagery, I would want a theory of imagery to stand at the center of my work, not a symbolic place-holder for which I had only enough intuitive grasp, along with a few explicitly stated properties, to guide further experimentation.

I trust the point is made. No criticism is directed at efforts that make progress, as all the above do. One can still wish for something different. One can also suspect that the reason why so many studies have this characteristic (this flaw?) is because of an accepted style of operation in psychology.

In all events, if I am going to study coding processes, I have to have a model of the coding operations themselves. I will, on balance, prefer to start with a grossly imperfect but complete model, hoping to improve it eventually; rather than start with an abstract but experimentally verified characterization, hoping to specify it further eventually. These may be looked at simply as different approximating sequences toward the same scientific end. They do dictate quite different approaches, as the present paper exemplifies.

Thus, the goal of this paper is to provide at least one explicit set of mechanisms for coding the stimulus. We could enunciate the fundamental operations that seem to be required and from there construct a system that seemed consonant with what is known generally about the information processing capabilities of humans. We will, instead, follow a somewhat different course and extend an existing model of human information processing. Consequently, we will start with an exposition of this model in Section II, and after this pose the issue of stimulus encoding in Section III. To make progress will require adopting a concrete task, which we do in Section IV. This permits us in Section V to define the extension to the system, which will be a perceptual mechanism, and to look briefly at its behavior in Section VI. In the final section (VII) we sum up the exploration.

II. THE BASIC MODEL

The basic model comes from the theory of problem solving that has developed from a study of small symbolic well-defined tasks (cryptarithmetic, chess, and elementary symbolic logic). The theory is set forth most completely in Newell and Simon (1972), but various earlier specialized versions and summaries exist (Newell, 1967; Newell, 1968; Simon and Newell, 1971).

The Elements of the Theory

Let me recapitulate briefly the elements of the theory. We will follow this up with a particular instantiation of the theory for a specific subject on a specific occasion. This latter will give us the requisite level of detail to pose the task of this paper. Since full detail will be provided in the second half, this initial statement can gloss over a number of details.

Structurally, the subject is an information processing system (IPS) consisting of a processor containing a short term memory (STM), which has access to a long term memory (LTM). The processor also has access to the external environment, which may be viewed as an external memory (EM).* The processor contains the mechanisms for elementary processes, for perception, for motor behavior, and for the evocation of conditional sequences of elementary processes.

The basic representation of information is in terms of symbols and symbolic expressions. Symbolic expressions are structures composed of discrete collections of symbol tokens, linked by relations (e.g., the next relation, where at most one symbol token immediately follows a given token, as in a list). Symbols, as realized in symbol tokens in symbolic expressions, designate other structures: of symbolic expressions, of elementary processes, and of the results of elementary processes. "X designates Y" is short hand for "X permits access to Y or to a representation of Y by some set of elementary information processes."

* Due account being taken for the initiation of action from the external environment, a feature not prominent in the task environments studied.

All action of the system takes place via the execution of elementary processes, which take their operands in STM. The only information available on which to base behavior is that in STM; other information (either in LTM or EM) must be brought into STM before it can effect behavior. At this level the system is serial in nature: only one elementary information process is executed at a time and has available to it the contents of STM as produced by the prior elementary processes. Seriality here does not imply seriality either of perception or of accessing of LTM.

Problem solving takes place as search in a problem space, each element of which represents a possible state of knowledge about the problem. A problem space is defined by (1) a representation of the possible states of knowledge (e.g., a language, such that each expression in the language constitutes a possible state of knowledge) and (2) a set of operators for moving from one element of the problem space to another, thus acquiring new knowledge or abandoning old knowledge. Central to the theory is the assertion that the problem space can be specified in finite terms for particular subjects and particular tasks. Not all the knowledge that a subject has is represented by his position in the problem space (e.g., knowledge about his path through the space).

The problem space is not represented in extension in the IPS (i.e., in the subject). However, it exists potentially, because at least one particular knowledge state is represented explicitly in the IPS (namely, the subject's current location in the space) and the IPS has processes corresponding to all the operators of the space, hence can generate other elements of the problem space. The language of knowledge states, then, is representable in the symbolic expressions that form the basic representation of the IPS. Further, the current knowledge state must exist in some form in the memories of the subject, namely in STM, LTM, and EM.

The program of the subject appears to be well represented by a production system.* This is a scheme of the form:

$$C_1 \dashrightarrow A_1$$

$$C_2 \dashrightarrow A_2$$

$$\ldots$$

$$C_n \dashrightarrow A_n$$

Each of the lines consists of a condition (the C_i) and an action (the A_i), and is called a production. The ordered list of productions is called a production system. The system operates by continually selecting for execution the first action from the top whose condition is satisfied. Since the actions modify the information on which the conditions are based, the same action need not (and in general, will not) be evoked on successive cycles of the system.

The conditions operate on the current knowledge state. (That is what makes it both current and knowledge: it determines the immediately next action of the subject.) Actually, the conditions are limited to that part of the knowledge that is in STM.** (That is what gives the STM its special role and makes knowledge in EM or LTM indirect.)

The actions may be operations of the problem space or sequences of such operations:

$$C_i \dashrightarrow Q_1\ Q_2\ \ldots\ Q_m$$

In this latter case the sequence is executed unconditionally, except that termination of the sequence is possible after any operation. Depending on how the

* Production systems constitute a family of computational and logical systems much studied in computer science (see Minsky, 1967; Hopcroft and Ullman, 1969). Members differ considerably in the details of the conditions, actions, control structure and the data types on which they work.

** There is a question about the status of the immediate perceived EM.

the problem space is defined, the actions may or may not include additional operators (e.g., those involved in attention control).

To provide a complete model for a subject's problem solving requires giving the problem space and the production system. It also requires giving the details of the memory structures and the symbolic representation, which is implied indirectly in the first two items. On the other hand, strategies and methods of problem solving are to be represented by the contents of production systems, and are not given as separate desiderata.

The work mentioned earlier (e.g., Newell and Simon, 1972) attempts to fill out the gross picture just given, as well as show that the behavior of human subjects can be described successfully by means of such a theory when the details are filled in. We are not concerned here with recapitulating that story, but in shedding light on the encoding of knowledge.

However, we will set out in the next section a specific version of the general theory. This will provide a detailed set of mechanisms for all the parts which have been described above only in general terms. We will use a version in a problem solving task called cryptarithmetic, not because it is well adapted to the study of stimulus encoding -- which it is not -- but because it represents well the current level of analysis.

A Production System for S2 on CROSS+ROADS=DANGER

We wish to model a subject (S2) behaving on the cryptarithmetic task, CROSS+ROADS=DANGER. For those not familiar with the task, Figure 1 gives the instructions. The protocol for this subject is discussed in detail in (Newell and Simon, 1972, Chapter 7); he is the subject for which we have detailed eye-movement records. The production system to be presented here corresponds to that presented in the book, but differs in the underlying language for production systems, the representation of knowledge elements and some details of the immediate processor.

The elements that constitute knowledge are linear expressions. For instance (NEW D = 1) is to be read: "D = 1 and this is new information." (GOAL * PC COL.2) is to be read: "The goal of applying the operator PC to column 2 and this goal current." In general, English terms are used in knowledge elements, e.g., GOAL, NEW, =, etc. In the model all such terms acquire their significance (i.e., their meaning, their semantics, their operational character, etc.) entirely by participation in productions. For example, elements containing the term GOAL are goal-like precisely to the extent that there are productions that respond to elements containing the term GOAL (by matching on their conditions) and manipulate them in goal-like ways, such as permitting subgoals, resuming superordinate goals, organizing behavior to attain goals, and so on.

STM consists of a list of knowledge elements, i.e., a list of symbolic expressions. It is of limited capacity in this regard, holding (in the example

```
  C R O S S
+ R O A D S
-----------
D A N G E R
```

The above expression is a simple arithmetic sum in disguise. Each letter represents a digit, that is, 0, 1, 2, ..., or 9. Each letter is a distinct digit. For example, C and A many not represent the same digit.

What digits should be assigned to the letters such that when the letters are replaced by their corresponding digits, the above expression is a true arithmetic sum?

Figure 1: Instructions for Cryptarithmetic Task.

run shown later) 7 elements.* STM holds the 7 most recent expressions: they are pushed into the front of the memory and disappear off the end.

Figure 2 gives the full definition of the production systems for S2. The expressions in the figure are interpreted by a production system program (called PSG, for production system, version G). The system is written in a system building system called L*(F) (for L*, version F), which is a homegrown system (Newell, McCracken, Robertson and Freeman, 1971) though nothing has to be known about L* for this paper.

There are 8 problem space operators. Three of them (FC, FNC and FLA) function to direct attention; essentially they obtain operands. Three others (PC, AV and TD) do the main work.** Finally, two operators (RA, RV) are devoted to recall of information in LTM.

A complete model of the subject's behavior would include a representation of the display (essentially as given in Figure 1) and programs for each operator. In fact, the model makes a distinction between the control structure for evoking the operators and the internal structure of the operators themselves. Consequently, the system of Figure 2 goes down only to the evocation of operators. It then asks for an exogenous specification of the ouput of the operator within the context in which it was evoked. This shows up in Figure 2 by the fact that all operators are defined as (OPR CALL). OPR identifies the symbol as designating

---

* The behavior of the system in problem solving appears to depend only weakly on the exact assumptions about the size of STM and whether it is constant or somewhat fluctuating in size. This is because STM is indeed a buffer memory, which is mostly filled with junk anyway. The general problem solving methods used by a subject avoid critical dependence on the size of STM. With respect to memory errors (which are rare events), the dependence on STM characteristics is not well understood for humans and is not represented in the system.

** Other descriptions include a fourth operator, GN, which generates the values of a letter. The bit of behavior we are simulating does not happen to evoke GN, so it is absent from the system described here.

```
J CY15Ft CRYPTARITHMETIC PRODUCTION SYSTEM
I        FOR S2, TRY 15 (BOOK VERSION) ON CROSS+ROADS.OANGER
t        REQUIRES PSGF, U1F, OICTF, UTILF
/
DEFINE.SYMBOLS!
I
CY.CONTEXT SET.CONTEXT!
J
> MOKE NAMES AVAILABLE FOR USE IN CY.CONTEXT
TO* TO CHANGE.NAMES!
I
OEFINE.PROCESSES!
;
5 NOTICING OPERATORS:
j   SET VALUES OF VARIABLES ANO (POSSIBLY) PROOUCE <NTC-EXP>
;
FC:  (OPR CALL)   ; FINO COLUMN CONTAINING LETTER <L> (•> <COL>)

FNCi  (OPR CALL)   FINO NEXT UNPROCESSED COLUMN («> <COL>)
FLA:  (OPR CALL)   FIND LETTER ABOVE LINE IN COLUMN <COL>U> <L>)
I
; STM OPERATORS
I   PRODUCE NEW ELEMENTS OR MODIFY EXISTING ELEMENTS IN STM
;
PCt  (OPR CALL)   PROCESS COLUMN <COL> («> <EXP>, <GOAL>)
AVt  (OPR CALL)  1 ASSIGN VARIABLE <VAR> (»> <EXP>, <GOAL>>
TD:  (OPR CALL)    TEST DIGIT <0> FOR LETTER <L> U> <EXP>,<COAL>)
RA:  (OPR CALL)  ; RECALL ANTECEDENT OF <EXP> («> <EXP>,<CCL>>
RVt  (OPR CALL)  ; RECALL VARMBLE <VAR> («> <D>)
;
DEFINE.SYMBOLS!
; DEFINE CLASSES FOR USE IN PRODUCTION CONDITIONS
;
; CLASSES FOR CRYPTARITHMETIC KNOULEOGE

<0>t    (CLASS 0 12 3 4 5 6 7 8 9)
<L>:    (CLASS A C D E G N 0 R S>
<C>:    (CLASS CI C2 C3 C4 C5 C6)
<COL>:  (CLASS COL.1 COL.2 COL.3 COL.4 COL.5 COL.6)
<VAR>:  (CLASS <L> <C>)
<OBJ>:  (CLASS <L> <D>)
<EQ>.   (CLASS x < ~ )
<IEQ>:  (CLASS  > < > « < « )
<REL>J  (CLASS <EQ> <IEQ>)
<TAG>:  (CLASS NEW OLD NOT)
<EXP>:  (CLASS (<VAR> <REL> <OBJ>) (<TAG> <VAR> <REL> <OBJ>>>
I
j CLASSES FOR GOAL EXPRESSIONS
/
<G>:     (CLASS GOAL OLOG)
<SIG>:  (CLASS * % + -)
<END>:  (CLASS + -)
<COND>:  (CLASS -COND +COND)
<SIG-EXP>:  (<SIG> <CONO>)
<COND-EXP>:  (COND <CONO> <END>)
<GOAL-TYPE>t  (CLASS USE GET CHECK RECALL SOLVE <OPR>)
<GOAL-SPEC>:  (CLASS <COL> <VAR> <OBJ>
                     (<VAR> <COL>) (<COL> <VAR>) (<VAR> <OBJ>>)
<GOAL>:  (CLASS  (<G> te <SIG-EXP> <GOAL-TYPE>)
                 (<G> &fi <SIG-EXP> <GOAL-TYPE> U <GOAL-SPEC>>)
/
<OPR>:  (CLASS PC AV TD RA RV)
```

Figure 2: Specifications for S2 on CROSS+ROADS=DANGER

```
<NTC>: (CLASS FNC FC FLA)
<NTC-COND>: (CLASS MORE END)
<NTC-EXP>: (CLASS (<NTC-COND> <NTC>) (OLD <NTC-COND> <NTC>))
;
<KNOWLEDGE-ELEMENT>: (CLASS <GOAL> <EXP> <COND-EXP> <NTC-EXP>)
;
; TOTAL PRODUCTION SYSTEM
;
PS1: (GS1 PS2 GS2)
;
; PRODUCTION SYSTEM FOR MANIPULATING GOALS
;
GS1: (G1 G3 G10 G9 G5 G6 G7 G8 G4)
GS2: (G2 G11)
;
G1: ((GOAL <END>) --> (GOAL ==> OLDG))
G2: ((GOAL *) ABS AND (GOAL %) --> (% ==> *))
G3: ((GOAL *) AND (GOAL *) --> (* ===> %))
G4: ((GOAL * <OPR>) --> <OPR>)
G5: ((GOAL * <COND>) AND (OLDG <END>) --> (<COND> ==>)
        (COND <COND> <END>))
G6: ((COND +COND +) AND (GOAL *) --> (COND ==> OLD COND)
        (* ===> +))
G7: ((COND -COND -) AND (GOAL *) --> (COND ==> OLD COND)
        (* ===> -))
G8: ((COND) AND (GOAL *) --> (COND ==> OLD COND))
G9: ((MORE) AND (GOAL *) --> (* ===> %))
G10: ((MORE <NTC>) AND (END <NTC>) --> (MORE ==> OLD MORE))
G11: ((GOAL %) ABS AND (GOAL *) ABS AND (GOAL <END> SOLVE) ABS
        --> (GOAL * SOLVE))
;
; PRODUCTION SYSTEM FOR TASK
;
PS2: (PD5 PD3 PD4 PD2 PD6 PD7 PD9 PD10 PD11 PD12 PD1 PD8)
;
PD1: ((NEW <L> = <D>) --> FC (GOAL * USE <COL>))
PD2: ((NEW <L> <-- <D>) --> (GOAL * PC))
PD3: ((GOAL * USE <COL>) --> (USE ==> PC))
PD4: ((GOAL * GET <VAR>) --> FC (GET ==> PC <COL>))
PD5: ((GOAL * USE <COL>) AND (OLDG - PC <COL>) -->
        FLA (USE <COL> ==> AV <COL> <L>))
PD6: ((NEW <L> <IEQ> <D>) --> (GOAL * AV <L>))
PD7: ((NOT <L> <-- <D>) --> (GOAL * AV <L>))
PD8: ((GOAL * SOLVE) --> FNC (* ==> %) (GOAL * USE <COL>))
PD9: ((NEW <L> = <OBJ>) AND (<G> <SIG> TO <L> <OBJ>) ABS -->
        (GOAL * TO <L> <OBJ>))
PD10: ((NOT <L> = <D>) --> RA (NOT && <EXP>))
PD11: ((GOAL * CHECK <VAR>) --> RA (NEW && <EXP>))
PD12: ((GOAL * RECALL <VAR>) --> RA (<VAR> ==> <VAR> <COL>) RV)
;
STM: (NIL NIL NIL NIL NIL NIL NIL)
;
"CY15F LOADED (NOTE: DIGITS ARE CHARS)" RETURN.TO.TTY!
```

FIGURE 2. SPECIFICATIONS FOR S2 ON CROSS+ROADS=DANGER

an operator. CALL calls to the terminal running the system to botain the required output of the operator. The user provides the behavior of each of the operators by typing in these requested outputs.

There are good reasons to run a model of problem solving this way. To model the operators requires a more detailed model of the immediate processor and perceptual mechanisms than the theory of problem solving is prepared to provide. Perhaps more important, in mapping the output of the system on the behavior of a subject there must be a way to correct the system when it commits errors (often called "putting the simulation back on the track"). If this is not done, the accumulation of a few errors causes the system and the behavior to diverge completely and bear no further resemblance to each other, even though the model may be perfect from then on. This follows from the memory-dependent character of cognitive behavior, which tends to magnify small differences. One technique to correct for errors is to force the behavior of the operators so as to keep the system on the track (though stringent limits bound how much a model can be steered in this way). Error scores can then be generated by examining the number of arbitrary outputs required of the operators. Ultimately, the system does not run either in pure CALL mode or in automatic. Rather, programs are used for the regular and predictable parts of the operators, and CALLs are used only when the output cannot be predicted. However, the system of Figure 2 calls for all operator outputs.

The condition sides of productions are written in terms of classes of expressions, which also serve to define completely the forms of knowledge elements. The classes assumed in the example are given after the operators in Figure 2.* The operational significance of these classes is determined by how they occur in the condition sides of the productions given later in the figure. (A few classes, e.g., <GOAL>, never occur per se in condition, but merely serve to show the form of expressions.)

* The angle-bracket notation for class names is purely mneumonic and is not interpreted by the system.

The productions themselves are divided into two functional groups, the G's and the PD's. The G's are concerned with the manipulation of the goal system. The PD's are concerned with the task of cryptarithmetic. The production system itself, PS1, is a single list of productions, but is given as three sublists: the productions of GS1 followed by those of PS2 followed by those of GS2. Seen as a single ordered list of productions, goal manipulation productions come first (i.e., have priority), except for the few in GS2 which provide a backup action in case none of the task productions is triggered by the current STM contents.

The detailed set of conventions for production systems are given in Appendix I. The easiest way to understand them is to consider simple examples of a particular production applied to STM. Afterwards we will comment on some of the psychologically relevant aspects. First, we describe the system in its own terms.

Figure 3 shows PD2 applied to a STM holding only a single expression.* Since this is matched by the condition form of PD2, the action is executed. The match consists of an identity between the constants NEW and <--, and class inclusion for s as a letter (the class <L>) and 1 as a digit (the class <D>). The system prints out that the condition of PD2 is satisfied (TRUE). This action consists of an expression, which then enters the STM. Since, the STM only contains a single element, this forces the prior element out of STM, as shown by the print out of STM after the action.

Figure 4 shows PD1 applied to a STM of three elements. The middle element matches PD1, thus evoking the action. Because this element, (NEW R = 5), was attended to by the evoked condition, it is moved to the front of STM. Thus, a continuous reshuffling of STM occurs according to what items are attended to (which amounts to an automatic rehearsal mechanism). The action of PD1 consists of two elements. The first is FC.

---

* The user's input is in lower case, the system's output in upper case. The system does not distinguish upper and lower case, e.g., stm = STM. try.pd is an executive routine preceding routine (here try.pd) immediately.

```
stm: ((new s <-- 1))
pd2 try.pd!
PD2: ((NEW <L> <-- <D>) --> (GOAL * PC))
PD2 TRUE
STM: ((GOAL * PC))
```

Figure 3: Entering new element into STM
Fixed size of STM

```
stm: ((now s <-- 1)(new r = 5)(goal * solve))
pd1 try.pd!
PD1: ((NEW <L> = <D>) --> FC (GOAL * USE <COL>))
PD1  TRUE
      (NEW R = 5)
      (<D> 5 <L> R)
      OUTPUT FOR FC = (<col> == col.1)
lz,z
STM: ((GOAL * USE COL.1) (NEW R = 5) (NEW S <-- 1))
```

Figure 4: Call on terminal for operator output
Assignment of value to class names
Sequence of actions

This operator produces the column which is to be attended to. However, as explained above, instead of executing a program for FC, the system calls to the terminal for an answer. It prints out the context in which this answer is to be provided, namely the elements that were recognized by the condition of PD1, including the values for variables and class names (that <D> is 5 and <L> is R). All other elements in STM are essentially out of reach by the actions (though another example later will qualify this statement). The answer, as typed in by the user (in lower case), indicates that the symbol <COL> is to have the value COL.1.* <COL> is a class name as well, but in the context of a production it can have associated with it the particular member of the class under consideration. The second element of the PD1 action is an element to be entered into STM, just as in the first example. However, this element contains a symbol that has an assigned value, so that the element is correspondingly instantiated.

Figure 5 shows a STM in which PD5 can be evoked. The condition of PD5 consists of a conjunction (AND) of two expressions both of which have to be found in STM. The order in STM is not important, as the example shows. However, the first element of the conditions serves to determine the value of <COL>, which is then used in the match of second element (notice that (OLDG - PC COL.3) was skipped over). The two elements matched by the condition of PD5 must be distinct; once the first one is matched it is excluded as a candidate for further matches. The action of PD5 is not to put a new element into STM, but to modify the one that is there. First, the attention-directing operator FLA is executed, leading to specifying <L> to be R. Then, in the first element of STM, (GOAL * USE COL.1), the symbol sequence "USE COL.1" is identified and replaced by "AV COL.1 R."

Figure 6 shows the operation of G3, the goal production that assures that only one goal is current at a time. STM contains two current goals

---

* The |z,z is a signal to return control from the user to the system. A signal is required because the system has given the user indefinite control.

```
stm: ((oldg - pc col.3)(goal * use col.3))
pd5 try.pd!
PD5: ((GOAL * USE <COL>) AND (OLDG - PC <COL>) --> FLA (USE <COL> ==> AV <COL> <L>))
PD5  TRUE

      (GOAL * USE COL.3)
      (<COL> COL.3)
      OUTPUT FOR FLA = (<l> == r)
l.z,z
STM: ((GOAL * AV COL.3 R) (OLDG - PC COL.3))
```

Figure 5: Conjunction of conditions

```
stm: ((goal * pc)(goal * solve))
g3 try.pd!
G3: ((GOAL*) AND (GOAL *) --> (* ===> %))
G3  TRUE
STM: ((GOAL * PC) (GOAL % SOLVE))
```

Figure 6: Each condition element matches distinct element
Modification of existing element

(each contains *). The condition side of G2 identifies both of these, because the match need only account for the symbols in the condition element. Thus (GOAL *) will match any goal element with the signal *. Since, as noted above, each element of a condition must match a distinct element of STM, the second (GOAL *), though identical to the first, matches the second element of that form in STM. The action of G3 is to replace the signal for current (*) with the signal for interrupted (%). Note that this takes place in the second element in STM, as designated by ===> (instead of ==> which operates on the first element).

Figure 7 shows the operation of G2, the goal production that assures that there is a current goal. It also consists of a conjunction of two condition elements. The first, however, requires the absence (ABS) of an element of the stated form, in this case the absence of a goal with the signal *. The second element identifies this most recently interrupted goal (the one with %): If there are several %-goals in the STM, then the first one is taken. Thus, the order of elements in STM is consequential, since an element toward the front can shield an element further back from being picked up. The action of G2 is to replace % by * in the second element identified. (Since the first element does not exist, the second is at the front of STM; hence ==> is appropriate rather than ===>).

G2 does not handle all situations that lack a current goal. If there is no interrupted goal in the STM (no goal with %), then G2 will not be evoked. However, G11 will then be evoked. It responds to an absence of a current goal, an absence of any interrupted goal and an absence of a goal saying the problem is all over (<END> being either of the terminating signals, + or -). Its action is to put the top goal (GOAL * SOLVE) back into STM. This production is one type of LTM retrieval, since it says that the top goal is remembered whether or not it remains in STM.

```
stm: ((goal - pc)(goal % solve))
g2 try.pd!
G2: ((GOAL *) ABS AND (GOAL %) --> (% ==> *))
G2  TRUE
STM: ((GOAL * SOLVE) (GOAL - PC))
```

Figure 7: Absence of element condition

```
stm: ((goal * pc)(goal % solve)(new s <-- 1)(oldg + av col.1 s)
      (oldg - pc col.4 r)(oldg - pc col.1)(old cond -cond -))
g4 try.pd!
G4: ((GOAL * <OPR>) --> <OPR>)
G4  TRUE
      (GOAL * PC)
      (<OPR> PC)
      OUTPUT FOR PC = (* ==> +)(ntc (new s <-- 1))
                      (new ==> old)(new r = 2)
!z,z
STM: ((NEW R = 2) (OLD S <-- 1) (GOAL + PC) (GOAL % SOLVE) (OLDG + AV COL.1 S) (OLDG - PC
COL.4 R) (OLDG - PC COL.1))
```

Figure 8: Complex output of operator
Use of NTC

A final example is given in Figure 8, which reveals something of the nature of the interaction between operators and productions. The STM is taken from the illustrative run shown later and contains a number of miscellaneous elements as well as those relevant to the current action. The current goal is to apply PC and this evokes goal production G4, leading to the call on the terminal. The output of PC, supplied by the user, provides several things. First, it changes the signal of the goal to +, since it is producing a new item of information. Second, in producing this item it makes use of the. element (NEW S <-- 1), and this must be changed to (OLD S <-- 1). If PC were realized by a production system itself, then its productions would both find this element in STM and modify it. A secondary effect would be to bring the element up toward the front of STM. Thus, to simulate this the action element (NTC(NEW S <-- 1)) notices (NEW S <-- 1) in STM and brings it forward; then the action (NEW ==> OLD) makes the change. Finally, the new knowledge element, (NEW R= 2) is produced. This example shows that the result of an operator, when called for, can be any sequence of actions that is legitimate for production.

The foregoing examples cover most of the types of actions possible. The full set is listed in Appendix I. We show a couple of pages of running trace from this system in Figure 9, so its total behavior can be followed through. The important thing to observe is the level of detail at which the system operates. We will not compare this trace with the subject's behavior, though for orientation Figure 10 gives the bit of protocol covered by the sequence of Figure 9.

```
psl ps!
0.  STM: (NIL NIL NIL NIL NIL NIL NIL)
G11  TRUE
1.  STM: ((GOAL * SOLVE) NIL NIL NIL NIL NIL NIL)
PD8  TRUE
      (GOAL * SOLVE)
      (NIL)
      OUTPUT FOR FNC = (<col> == col.1)
lz,z
5.  STM: ((GOAL * USE COL.1) (GOAL % SOLVE) NIL NIL NIL NIL NIL)
PD3  TRUE
6.  STM: ((GOAL * PC COL.1) (GOAL % SOLVE) NIL NIL NIL NIL NIL)
G4  TRUE
      (GOAL * PC COL.1)
      (<OPR> PC)
      OUTPUT FOR PC = (* ==> % -cond)(goal * get s)(goal * get r)
lz,z
10.  STM: ((GOAL * GET R) (GOAL * GET S) (GOAL % -COND PC COL.1) (GOAL % SOLVE) NIL NIL NIL)
G3  TRUE
11.  STM: ((GOAL * GET R) (GOAL % GET S) (GOAL % -COND PC COL.1) (GOAL % SOLVE) NIL NIL NIL)
PD4  TRUE
      (GOAL * GET R)
      (<VAR> R)
      OUTPUT FOR FC = (<col> == col.4)
lz,z
14.  STM: ((GOAL * PC COL.4 R) (GOAL % GET S) (GOAL % -COND PC COL.1) (GOAL % SOLVE) NIL NIL NIL)
G4  TRUE
      (GOAL * PC COL.4 R)
      (<OPR> PC)
      OUTPUT FOR PC = (* ==> -)
lz,z
16.  STM: ((GOAL - PC COL.4 R) (GOAL % GET S) (GOAL % -COND PC COL.1) (GOAL % SOLVE) NIL NIL NIL)
G1  TRUE
17.  STM: ((OLDG - PC COL.4 R) (GOAL % GET S) (GOAL % -COND PC COL.1) (GOAL % SOLVE) NIL NIL NIL)
G2  TRUE
18.  STM: ((GOAL * GET S) (OLDG - PC COL.4 R) (GOAL % -COND PC COL.1) (GOAL % SOLVE) NIL NIL NIL)
PD4  TRUE
      (GOAL * GET S)
      (<VAR> S)
      OUTPUT FOR FC = (<col> == col.2)
lz,z
21. STM: ((GOAL * PC COL.2 S) (OLDG - PC COL.4 R) (GOAL % -COND PC COL.1) (GOAL % SOLVE) NIL NIL
NIL)
G4  TRUE
      (GOAL * PC COL.2 S)
      (<OPR> PC)
      OUTPUT FOR PC = (* ==> -)
lz,z
23. STM: ((GOAL - PC COL.2 S) (OLDG - PC COL.4 R) (GOAL % -COND PC COL.1) (GOAL % SOLVE) NIL NIL
NIL)
G1  TRUE
24. STM:  ((OLDG - PC COL.2 S) (OLDG - PC COL.4 R) (GOAL % -COND PC COL.1) (GOAL % SOLVE) NIL
NIL NIL)
G2  TRUE
25.  STM: ((GOAL * -COND PC COL.1) (OLDG - PC COL.2 S) (OLDG - PC COL.4 R) (GOAL % SOLVE) NIL
NIL NIL)
G5  TRUE
27.  STM: ((COND -COND -) (GOAL * PC COL.1) (OLDG - PC COL.2 S) (OLDG - PC COL.4 R) (GOAL %
SOLVE) NIL NIL)
G7  TRUE
29.  STM: ((OLD COND -COND -) (GOAL - PC COL.1) (OLDG - PC COL.2 S) (OLDG - PC COL.4 R) (GOAL %
SOLVE) NIL NIL)
```

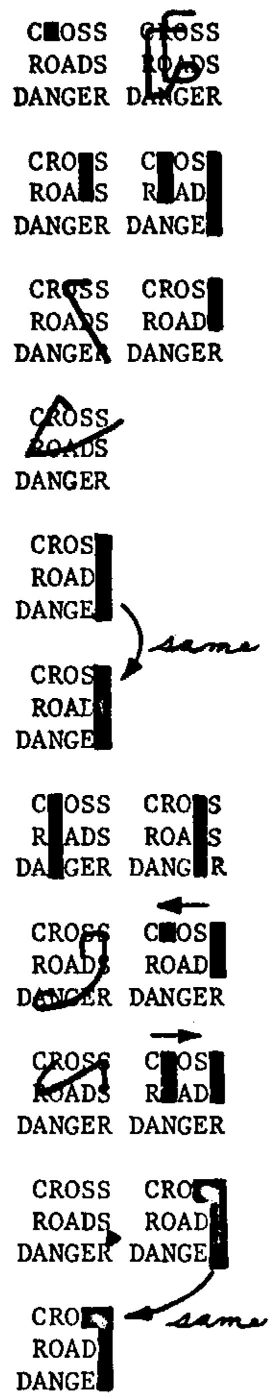
Figure 9: Trace from PS of Figure 2.

```
G1  TRUE
30. STM: ((OLDG - PC COL.1) (OLD COND -COND -) (OLDG - PC COL.2 S) (OLDG - PC COL.4 R) (GOAL %
SOLVE) NIL NIL)
G2  TRUE
31. STM: ((GOAL * SOLVE) (OLDG - PC COL.1) (OLD COND -COND -) (OLDG - PC COL.2 S) (OLDG - PC
COL.4 R) NIL NIL)
PD8  TRUE
      (GOAL * SOLVE)
      (NIL)
      OUTPUT FOR FNC = (<col> == col.1)
lz,z
35. STM: ((GOAL * USE COL.1) (GOAL % SOLVE) (OLDG - PC COL.1) (OLD COND -COND -) (OLDG - PC
COL.2 S) (OLDG - PC COL.4 R) NIL)
PD5  TRUE
      (GOAL * USE COL.1)
      (<COL> COL.1)
      OUTPUT FOR FLA = (<l> == s)
lz,z
38.  STM: ((GOAL * AV COL.1 S) (OLDG - PC COL.1) (GOAL % SOLVE) (OLD COND -COND -) (OLDG - PC
COL.2 S) (OLDG - PC COL.4 R) NIL)
G4   TRUE
       (GOAL * AV COL.1 S)
       (<OPR> AV)
       OUTPUT FOR AV = (* ==> %)(goal * get r)
lz,z
41.  STM: ((GOAL * GET R) (GOAL % AV COL.1 S) (OLDG - PC COL.1) (GOAL % SOLVE) (OLD COND -COND -)
(OLDG - PC COL.2 S) (OLDG - PC COL.4 R))
PD4  TRUE
      (GOAL * GET R)
      (<VAR> R)
      OUTPUT FOR FC = (<col> == col.4)
lz,z
44.  STM: ((GOAL * PC COL.4 R) (GOAL % AV COL.1 S) (OLDG - PC COL.1) (GOAL % SOLVE) (OLD COND
-COND -) (OLDG - PC COL.2 S) (OLDG - PC COL.4 R))
G4  TRUE
      (GOAL * PC COL.4 R)
      (<OPR> PC)
      OUTPUT FOR PC = (* ==> -)
lz,z
46.  STM: ((GOAL - PC COL.4 R) (GOAL % AV COL.1 S) (OLDG - PC COL.1) (GOAL % SOLVE) (OLD COND
-COND -) (OLDG - PC COL.2 S) (OLDG - PC COL.4 R))
G1  TRUE
47.  STM: ((OLDG - PC COL.4 R) (GOAL % AV COL.1 S) (OLDG - PC COL.1) (GOAL % SOLVE) (OLD COND
-COND -) (OLDG - PC COL.2 S) (OLDG - PC COL.4 R))
G2  TRUE
48.  STM: ((GOAL * AV COL.1 S) (OLDG - PC COL.4 R) (OLDG - PC COL.1) (GOAL % SOLVE) (OLD COND
-COND -) (OLDG - PC COL.2 S) (OLDG - PC COL.4 R))
G4  TRUE
      (GOAL * AV COL.1 S)
      (<OPR> AV)
      OUTPUT FOR AV = (* ==> +)(new s <-- 1)
lz,z
51.  STM: ((NEW S <-- 1) (GOAL + AV COL.1 S) (OLDG - PC COL.4 R) (OLDG - PC COL.1) (GOAL %
SOLVE) (OLD COND -COND -) (OLDG - PC COL.2 S))
G1  TRUE
52.  STM: ((OLDG + AV COL.1 S) (NEW S <-- 1) (OLDG - PC COL.4 R) (OLDG - PC COL.1) (GOAL %
SOLVE) (OLD COND -COND -) (OLDG - PC COL.2 S))
PD2  TRUE
53.  STM: ((GOAL * PC) (NEW S <-- 1) (OLDG + AV COL.1 S) (OLDG - PC COL.4 R) (OLDG - PC COL.1)
(GOAL % SOLVE) (OLD COND -COND -))
G4  TRUE
      (GOAL * PC)
```

Figure 9: (continued)

```
      (<OPR> PC)
      OUTPUT FOR PC = (* ==> +)(ntc (now s <-- 1))(now ==> old)(new r = 2)
Iz,z
58.  STM: ((NEW R = 2) (OLD S <-- 1) (GOAL + PC) (OLOG + AV COL.1 S) (OLOG - PC COL.4 R) (OLOG -
PC COL>1) (GOAL % SOLVE))
G1  TRUE
59.  STM: ((OLOG + PC) (NEW R = 2) (OLD S <-- 1) (OLOG + AV COL.1 S) (OLOG - PC COL.4 R) (OLOG -
PC COL.1) (GOAL % SOLVE))
PD9  TRUE
60.  STM: ((GOAL * TO R 2) (NEW R = 2) (OLOG + PC) (OLD S <-- 1) (OLOG + AV COL.1 S) (OLOG - PC
COL.4 R) (OLOG - PC COL.1))
G4  TRUE
      (GOAL * TO R 2)
      (<OPR> TD)
      OUTPUT FOR TD = (* ==> +)
```

Figure 9: Trace of PS of Figure 2



Phrase number	Time (secs)	Eye-movement Aggregations	Verbalization	STM number
B0	0			0
B1	6		CROSS plus ROADS is DANGER.	1
B2	10		Exp: Please talk.	(none)
B3	12		Yes.	
B4	14		S plus S has to equal R.	6
B5	18		And R will have to equal two S.	
B6	24		And S plus D also has to equal E.	14
B7	28		So I'll let S equal..	31
B8	36		Let S equal one.	48
B9	40		Therefore R will be two	53
				63

Figure 10: Protocol of S2 corresponding to trace in Figure 9.

Psychologically Relevant Features

We can now summarize and comment on a number of the psychologically relevant features of this system, both PSF, the production system, and CY15, the particular system for S2 on CROSS+ROADS=DANGER.

1. The system is serial, executing one action at a time.

2. In gross outline the memory structure is the classical one (Miller, 1956; Waugh and Norman, 1965) of an STM consisting of a limited number of chunks (here, symbolic expressions) and an LTM. No account has been taken of any of the indications that the memory structure might be more complex (e.g., Wicklegren, 1970; Broadbent, 1970). The problem solving behavior on which the model is based gives no hint that more complexity is required.

3. The representation of STM is complete and explicit. The number of chunks is a parameter of the system. The depth of detail that can be examined in each chunk is determined by the content of the production conditions.

4. There is no complete representation of LTM. A production is a retrieval on LTM; thus, the set of productions represents the content of LTM with the conditions of the production being the accessing paths. In addition, the ability to construct embedded expressions provides a second form of LTM. But there is no assertion that these constitute the only forms of LTM.

5. There is no direct representation of the writing of new information into LTM. Thus, the model does not handle learning situations that call for modification of LTM.*

---

* Currently, this is a key theoretical issue. It is not at all clear how LTM acquisition is to take place.

6. The productions represent a kind of S-R connection between a stimulus, as represented by elements in STM, and a response, as stored in LTM as an element on the action side of a production. However, productions are substantially more complex than classical S-R's. The link between S and R is made via a match operation that permits identification and instantiation of variables as well as tests for class membership. The actions permit modification of existing elements, as well as the addition of new ones, and in this latter case (the one more like the classical R) instantiation of variables is permitted, as determined by prior conditions or actions.

7. There is no representation of the EM, the perceptual mechanism, or the details of the immediate processor. Thus, the model is primarily about the control structure of behavior at the problem solving level.

8. Rehearsal occurs automatically in STM if something is attended to. this is a movement of the attended-to element in STM, not the creation of a copy. Strategies of rehearsal, therefore, are attempts to attend to something, possibly without concern for what processing occurs.

9. There is a highly particular matching system in PSF, the rules of which are summarized in Appendix I. Much of the variation in versions of the production system have been in details of this matching scheme. Almost no psychological information is available on which to make direct determination of these details. Several central issues can be identified in information processing terms, but for none of these can the psychological consequences be given:

(1) The productions deal with information they do not already know in full detail. That is, elements are identified by only partial information. What form should this indirectness take? The use of variables (the class names) is one form. Matching only the symbols in the condition element, not all the ones in STM element, is another (it lets an entire expression be picked up

by one part of it, as in the (GOAL *) conditions). Not matching in sequential order is yet another (providing something somewhere does respond to the order).

(2) What role does order in STM play? In the current system order is revealed in part by the masking of old elements by recent ones, which is a function of the match. This interacts strongly with the more general question of how STM should be structured (as a circulating memory, as a stack (as here), as an unorganized set of cells, as a constructed set of embedded expressions, etc.)

(3) Should an STM element be able to satisfy more than one element in a condition? The current systems insists on exclusiveness and without it many additional condition elements would be required to force exclusiveness. But should there be some mechanism to permit a designated condition element to be matched to any element in STM independent of other matches? Exclusiveness implies serial dependence in conditions, so that (A AND B) is not the same condition as (B AND A).

(4) How deep can the match search in an expression? The current system searches recursively; earlier versions did not, and in fact CY15 demands only a single level of search. That is, no embedded expressions such as (GOAL * (NEW <L> = (OLD <D>))) occur on the condition side of productions.

(5) What kind of processing can be done during a match? The current match permits a variable to be defined in one element and used in match elsewhere in the same element or in a following element. This enlarges the class of conditions that can be discriminated.

Earlier matches permitted only class inclusion to be recognized. (E.g., the system could match (<L> = <L>) but could not discriminate (R = R) from (R = D). Note that we are talking about what goes on in the match, not what is ultimately possible in the total system by the action of a sequence of productions. The current use of variables introduces a second form of serial dependence in condition elements.

10. Although it may have escaped the reader's notice, an additional "very immediate memory" is required to make the system operate. The actions of a condition make use of variable assignments determined during the match (e.g., the use of <COL> in Figure 4). This means that these assignments must be remembered from the moment that they are made (in the match) until they are used (in the action). This may be a matter of a few hundred milliseconds up to second, depending on the time span alloted to a production (a matter discussed below). The STM cannot be used for this memory in any simple way, since if these assignments were put into STM as an element, then another production would have to recongize them again for the action element to deal with. There is a temptation to identify this very-immediate-memory with some of the iconic stores. All that is established, of course, is a functional requirement. Conceivably it can dispensed with, but the contortions required are not yet clear.

11. There is no general way to designate directly the various elements of STM, e.g., by a naming or addressing scheme. The actions obtain access to the elements via their position in the condition of the match (which is essentially mirrored in terms of position in the front of the STM, though it need not be with slight variants of the shuffle scheme used for rehearsal). Non-matched STM elements do not exist for the actions (though subsearches can be made using the NTC mechanism). This leads to some awkwardness, e.g., in having separate modification operators (==>, ===>, ====>) corresponding to 1st, 2nd and 3rd elements.

However, the alternative of an additional naming device raises conceptual problems of how to use it and what it would mean in terms of implied mechanism.

12. Operators do not have arguments in the usual sense, e.g., PC(COL.3) or FC(R)., This latter form of operand designation is equivalent to a closed subroutine organization, in which the internal processes of the operator have access only to the arguments. Operators do have access to a context, ultimately bounded by STM. But they are more like open subroutines, which do their work in the same workspace as everyone else, having access to contextually embedding information, as well as leaving around their temporary internal working data, possibly to be responded to by other productions. Thus an operator, such as PC, should be viewed as if it were simply another collection of productions written in line with the main set. This raises problems about the maintenance of control within PC until it is finished, but these are to be solved by matching the productions of PC dependent on elements placed in STM by PC (such as goal elements).

This lack of clean subroutine hierarchy appears to have both positive and negative consequences* On the systems side, it makes it difficult to construct production systems that accomplish specific tasks. The programmer (so to speak) cannot easily control what processing occurs, as he can when working in a standard programming system. On the psychological side, the lack of hierarchy accords well with a single level of awareness and with the sort of supervisory awareness that appears to be a concomitant of much conscious processing (e.g., observing the on-going processing). It also accords well with the potential for distraction that appears to characterize much human processing. In all cases, unfortunately, no good empirical characterizations exist that permit more than informal comparison.

13. When to copy a data structure and when to use the same data structure that occurs in a different context is a general systems problem. It is unresolved here as well. Identity of structure is required at some level, yet if the identical structure is used in two places, a modification at one place communicates (so to speak) simultaneous modification to the other place. This is both a powerful device and a source of confusion and error. The issues are not clear from an information processing viewpoint, much less from a psychological one.

14. The productions represent the basic action cycle of the cognitive system. Thus, the time associated with a production must be somewhere in the 50 - 100 ms range. It is unclear whether the times typically generated in a Sternberg type of experiment, which are around 30 ms per symbol examined, are to be taken as per-production or as indicating something about the search of a single production through STM. Typical internal processing acts, such as going down the alphabet, seem to require of the order of 200 ms per item. But these would seem to require several productions per item. The counts shown in Figure 9 are obtained by adding 1 for each action element. They underestimate the time involved (i.e., do not multiply them by 100 ms per production to get the time), since the time of the operators are not included. For instance, the subject actually takes 8 seconds to perform the simple addition of S+S with (S <-- 1) to get (R = 2), which only gets a count of 1 in the figure.

15. Although the implementation of the selection of the next production is clearly a serial affair in PSF, it undoubtedly corresponds to some parallel process.* The little production systems, such as CY15, are to be considered embedded in a very large set of production (10^8 ?), i.e., of the order of LTM. There may be context mechanisms that in fact select out a small production system for the control of local behavior, but the theory does not yet contain any hint of these.

In general the notion of parallel matching poses no difficulties, with two exceptions. First, the ordering of the productions imposes a global constraint, which could make parallel processing difficult. However, the functional aspect of the ordering appears that specific productions shield general (back-up) versions of related productions. Thus, the ordering is only effective in little strands, which may prove tolerable. Second, with a complex match, involving variable identification and subsequent use within the match itself, the problems of carrying out an indefinite set of such processes simultaneously poses some difficulties. The imaginable sort of broadcast, content-addressed memories work with the matching of constants, i.e., with locally definite patterns. With enough local logic, of course, almost anything is possible, but there may still be a strong interaction between the amount of parallelism and the sophistication of the matching process.

---

* As a side note, there is no dissonance (much less conflict) in a system being both highly serial and highly parallel at the same time (though not, of course, in the same respects).

16. The system has a system of goals, meaning thereby a set of symbols that control processing in the service of ends to be achieved, permitting the creation of subgoals and the interruption of goal activity with its resumption at a later time.* The goal stack is not a separate memory, but is part of STM, with the various goal elements co-existing with other knowledge elements and taking up capacity. The production system for handling the goals (GS1) could be considered hardware relative to the production system for cryptarithmetic (PS2). There are additional advantages to handling the goal stack in STM (besides avoiding the assumption of a distinct memory), namely, that STM contains knowledge of old goals, even after they have been popped off the goal stack by succeeding or failing. This feature is actually used in PD5 and PD9.

* See Newell and Simon (1972, Chapter 14) for a discussion of the essential features of a goal system.

III. ON ENCODING THE STIMULUS

With the context provided by the model of information processing just described we can turn to the formulation of the problem of encoding the stimulus. It is worth noting, right at the start, that despite the somewhat recent emergence of coding as a significant theme in the main stream of psychology, the problem is not at all special. As soon as one proposes to design an information processing system to accomplish any of the tasks studied, say, in the psychology of learning, then the issue of representing the stimulus and the encoding operations to map the stimulus into its internal representation are forced to center stage. Only by approaching the problems of psychology by descriptive models that deal only in abstract features of behavior, can the issues of encoding be avoided.*

Three things would seem to be involved in the encoding of a stimulus: (1) the act of encoding; (2) the representation of the code; and (3) the act of decoding. However, it is only in a pure communication system that matters are so simple, where the only use made of the code is to decode it at the other end of the line. In a cognitive system, all manner of processing is accomplished in terms of the internal representation (i.e., the code): it is analysed for significant features, problem solving methods are selected for it, these methods manipulate and modify it, determination of whether the task is accomplished is made by further processing of it, and so on. Thus, the act of decoding must be extended to an indefinite notion of use of the internal representation.

---

* Actually, constructing discrete symbolic simulations of the human contains its own dangers in masking the question of encoding. The stimulus must be represented in a discrete symbolic form for use in such simulations, hence it must in fact be encoded (relative to the actual stimulus faced by the human). Is is possible to unwittingly perform a significant part of the stimulus encoding performed by the human in setting up the "stimulus" in the model.

Let us consider, then, the first two items: the act of encoding and the code. In some sense the most important of these is the code. As indicated above, it is the code that influences all the processing that follows. Conversely, it is the code that is most easy to determine experimentally, since its characteristics are evidenced in many sorts of behavior. In agreement with this, most studies of coding have been devoted to establishing either that coding per se was present (a somewhat redundant exercise given the present viewpoint) or the nature of the code in a specific task environment.

The reasons for concern with the mechanisms of encoding, rather than just with the final code, are at least three-fold. First is the general presumption, stated at the beginning of the paper, that if one is to study coding one should have a model of the encoding process. Second, and a partial justificiation of the first, is the presumtpion that knowing how codes are formed will tell something about which codes eventually get formed and under what conditions. We will find out why we appear to be so sensitive to repetitions and alternations in the most diverse guises, when familiar patterns dominate over ruley patterns and vice-versa, when an established pattern inhibits another pattern from being seen, and so on. Third, coding is such a central feature of human information processing that it is necessary to have some model of it in order to develop a model of the immediate processor.

Encoding is not equivalent to all information processing, as the above remarks on the use of codes was meant to indicate. Yet, encoding is equivalent to the generation of internal representations. As such, the processes of encoding are not to be inferred from viewing the collection of different internal representations in use by humans. That collection is too diverse and its sources too multifold to permit such inferences.* The story of any major representation for an individual (such as how an astronaut encoded the stimulus of the approaching moon) involves chapters on learning, education, calculation, perception, conversation, and on.

We wish to focus on the coding events that happen immediately when a stimulus is presented. An act of encoding happens there, since the subject cannot deal with the stimulus at all without producing such an encoding. This encoding may be the product of an indefinite amout of past processing and experience embedded in a current operating context of some depth. It still must be effected with only a modest amount of processing and with only a modest amount of understanding of the stimulus. These limitations follow from the decision to look at the leading edge of encoding: there is not time to do much processing or to develop much understanding; additionally, to do so would imply operating on the encoded stimulus, which would put the processing beyond the point of our interest.

This focus may be viewed as primarily tactical, to produce a scientific problem of manageable size. However, there are more substantial reasons. Changes of representation during the course of processing appear to be rare (though by no means absent). Certainly, in the problem solving tasks studied in

---

* Indeed, what is surprising is the need to demonstrate that encoding is present, which has been the clear attempt in much of the psychological literature on coding. That is, it would be surprising, except for the prior position of SR psychology that ignored the encoding problem, except in rather carefully framed ways (such as the methodological issues of the nature of the functional stimulus).

Newell and Simon (1972) the problem representation remained fixed for most subjects. Furthermore, these representations were quite close to the problem-as-presented. Thus, the major part of stimulus encoding may occur in the instant, so to speak, when the new situation is presented. Building up a representation may require the extensive chapters mentioned above, but it may only become effective if it can be assimilated into an encoding operation that takes place in short order.*

Concern with the immediate processing of the new stimulus implies contact with perceptual mechanisms. Indeed, perception may be conveniently defined as the initial encoding of the stimulus -- the one that cannot be fractionated further by the behaving subject by normal means. However, the study of encoding mechanisms cannot be limited to perception, as it is usually defined and studied, since many of the issues of encoding involve the participation of conceptual information and conceptual processing.

* We do not put aside the processes involved in change of representation as uninteresting. Indeed, they seem both crucial and fascinating. Being rare events and under subject control, they are somewhat harder to capture experimentally than initial encodings, which are time locked to the presentation of a new stimulus.

Existing Proposals for the Mechanisms of Coding

We asserted above that the coding literature generally addresses itself to the existence and nature of the code, and not to the mechanisms of encoding. There are, however, a few studies that provide concrete proposals.

The work on EPAM (Elementary Perceiver and Memorizer) provides a detailed model of the encoding of verbal stimuli (Feigenbaum, 1961; Simon and Feigenbaum, 1964). If a presented stimulus can yield a familiar sequence of features then it is encoded as a recognized chunk. The discrimination net used by EPAM is the mechanism of encoding and the growth of this net is a model of how new encodings become possible. Although the original work did not emphasize the encoding aspects, current work on how people perceive and remember complex chess positions constitutes a direct study of encoding (Chase and Simon, in press).

EPAM is a model of perception, the net being a mechanism that is evoked prior to STM, which receives the coded chunks as they are recognized. Thus, EPAM places the encoding operation in the perceptual mechanism and places the modificiation of the encoding in the relatively slow process of storage in LTM. The encodings permitted by EPAM are essentially structureless -- whatever familiar patterns have been stored away. Some structure can be imposed on the patterns by suitable constraint in the learning mechanism. This has been done in the chess perception situation, where the patterns to be learned on the chess board are generated by relations that have chess-functional significance (e.g., who defends who). Still EPAM does not provide a model for the encoding of novel structured situations.

A variety of programs dealing with tasks involving the creation of conceptual structures do provide proposals for the mechanisms for encoding novel structure: classical discrete attribute-value concepts (Hunt, 1962; Johnson, 1964); binary choice experiments (Feldman, 1961; Feldman, Tonge and Kanter, 1963)*; and sequence extrapolation tasks (Simon and Kotovsky, 1963). Let us consider the latter example briefly; it will include the lessons from the others.

The task is to predict the next members in a sequence whose initial terms are given, e.g., A B B C C D D _ _ . Simon and Kotovsky put forward a theory whose essential element was the representation that a subject would develop for the series, i.e., an encoding of the stimulus. For the above series the encoding would be (Alphabet; Ml = A) [Say (Ml) , Next (Ml), Say (Ml)] which can be read: the alphabet is the standard alphabet; the initial value of pointer Ml is the letter A; say Ml; move Ml to the next member in the alphabet; say Ml; now repeat the sequence in brackets. The interpretation rules we have just indicated in concrete form tell how to use the representation. The subject presumably can manipulate such a representation rather freely. For example, he could answer such questions as: Will W ever occur in the sequence? (yes); or What letters occur in the sequence only once? (Only A).

In addition, Simon and Kotovsky provided a program for how the subject would induct the sequence from the given data. He would first attempt to discover a period in the given data (here 2) and the alphabet (here the standard alphabet). The he would set up a hypothesis in the form of the specifications for each term in the cycle, e.g., [x^ x^], where each x^ is an expression that ends in the production of the given member of the sequence. Matching these against successive cycles of the given data would show that x^ has to

It is necessary to reach back to early work of an information processing sort to obtain suggestions about encoding mechanisms. Although some recent work in binary sequence prediction has emphasized strongly the structured aspects (e.g., Myers, 1970), it has done so by focussing on the codes themselves, i.e., the run structure. This is a good illustration of the point made earlier about the character of the literature, even when working in a generalized information processing framework.

be Say(M1) (where M1 is a variable pointer into the alphabet) and x_2 has to be Next(M1), Say(M1).

The important aspect of Simon and Kotovsky's proposal for the encoding of the stimulus (the sequence) is that it is conceptual -- that is, it occurs in the subject by deliberate acts of investigation and hypothecation in time periods of the order of tens of seconds. The initial encoding of the sequence is taken as we have represented it in the text, as a sequence of distinct letters (A B B ...). The additional structure is sufficiently disguised that the subject requires cognitive investigation to uncover it. This is in marked contrast with EPAM, in which the subject becomes aware only of the recongized chunks in the stimulus.

The other examples of work on concept formation generally concur.* The behavior model is at the processing level of many trials (covering tens to hundreds of seconds), thus being behavior at the cognitive level. The basic mechanisms are those of hypothesis and test, where sometimes the hypothesis is a form, whose details can be filled in by matching to the available data about exemplars. Most of these models, in common with the work of Simon and Kotovsky, do not incorporate a detailed model of the immediate processor and of STM, although they sometimes reflect short term memory load in a gross way. For example, Simon and Kotovsky measure the difficulty of a concept by the number of independent pointers, M1, M2, ..., that have to be maintained.

---

* It is worth noting that a number of studies have appeared dealing with coding of sequences (Leewenberg, 1969; Restle, 1970; Vitz and Todd, 1969), similar to the Simon and Kotovsky study. None of these, except that of Simon and Kotovsky, provide proposals about the encoding mechanisms. However, in an as yet unpublished paper Simon (1972) analyses all of these schemes and shows their fundamental similarity in terms of the code. Thus, we can assume, perhaps, similarity of the encoding procedures.

What is Provided by the Existing System

Let us now consider the present system, as exemplified by the production system in Figure 2, to see what it provides in the way of encoding mechanisms and what it is missing.

First, in line with the view already expressed of the ubiquity of encoding, as equivalent with internal representation, the theory provides a clear formulation of the encoding used by the subject for the task (here cryptarithmetic). The problem space is, in fact, exactly a statement of how the subject encodes the task: the basic concepts he uses; the way he can form them into larger concepts; and the operations he has for creating new instances of these concepts and responding to the instances he already has. Although we have not detailed it here, it is shown in great detail in Newell and Simon (1972) that the problem space is not determined by the task, but represents a construction by the subject. Thus, different subjects can have different problem spaces and, as one would expect, problem solving is strongly affected by the problem space used by a subject.

However, no theory is put forth about how a subject comes to have a specific problem space or what mechanisms determined it from the given information about the task (i.e., the stimulus). If we examine the model in Figure 2, We see that it finesses completely the input side from the environment, dealing only with the cognitive behavior on the internal representation in STM. Even if we extend the model to include specific processes for the operators (and substantial detail is given on these in the book), it would still say nothing about the encoding of the perceived stimulus.

However, the theory does provide: (1) the form of the encoding, namely, the knowledge elements in STM; (2) the ways encoded knowledge can be read, namely, the types of conditions; and (3) the cognitive operations that manipulate encoded knowledge, namely, the types of actions that are possible. These provide a frame into which a complete theory of encoding must fit. Moreover, the theory provides an essentially complete set of mechanisms for the encoding that goes on at the cognitive level, as revealed by the various studies of concept attainment described above. For these encodings operate on representations that already exist in STM, producing other encodings in STM.

To clarify exactly what is provided by the theory as initially given, let us consider a simpler example than the sequence extrapolation. The task of Neal Johnson (1970),* already mentioned at the beginning of the paper, is a good example of a direct study of encoding. The subject is asked to perform a paired associate task in which the stimuli are digits and the responses are sequences of consonants, e.g., 1 - XQKFH. However the consonant sequences are presented (in the various experimental conditions) with different spacing: X QK FH versus X QKF H versus XQ KF H, etc. The underlying hypothesis is that the subject will encode the stimuli in the "obvious" fashion indicated by the spacing and that this will be revealed by the existence of errors in the responses, given some assumptions about the way the decoding occurs to make the response.

The theory at hand provides for a direct translation of a number of the features of this task, while remaining silent on some others. Figure 11 gives a small system that contains the natural encoding corresponding to Neal Johnson's theory plus a set of productions for decoding this representation to yield the response. The example contains a single memorized paired associate (1 - X QK FH), since all that is important is to illustrate the scheme. It is represented as a production (PJ20), with the stimulus on the condition side and the encoded response as the action. The production PJ1

---

* A discussion is given in the present volume as well.

```
; NJ: PERFORMANCE SYSTEM FOR NEAL JOHNSON CHUNKING TASK
;     (IDENTICAL TO NJ.A03)
;
DEFINE.PROCESSES!
;
SAY: (OPR <ITEM>@L PRVL)
;
DEFINE.SYMBOLS!
;
<D>: (CLASS 0 1 2 3 4 5 6 7 8 9)
<K>: (CLASS B C D F G H J K L M N P Q R S T V W X Y Z)
;
<ITEM>: (VAR)
X0: (VAR)
X1: (VAR)
X2: (VAR)
X3: (VAR)
X4: (VAR)
;
PJ4: ((SEQ X1 X2 X3 X4) --> (SEQ ==> OLD SEQ)
      X4 X3 X2 X1)
;
PJ3: ((SEQ X1 X2 X3) --> (SEQ ==> OLD SEQ)
      X3 X2 X1)
;
PJ2: ((SEQ X1 X2) --> (SEQ ==> OLD SEQ)
      X2 X1)
;
PJ1: ((SEQ X1) --> (SEQ ==> OLD SEQ) X1)
;
PJ10: (<ITEM> == <K> --> SAY EMBED (<ITEM> ==> SAID <ITEM>))
;
PJ20: ((SR 1) -->
      (SEQ X (SEQ Q K) (SEQ F H)))
;
PS2: (PJ4 PJ3 PJ2 PJ1)
PS1: (PJ10 PS2 PJ20)
;
STM: (NIL NIL NIL NIL NIL NIL NIL)
TOP.GOAL: (SR 1)
;
"NJ.A03 LOADED" RETURN.TO.TTY!
```

FIGURE 11. PRODUCTION SYSTEM FOR THE DECODING AND RESPONDING PART OF NEAL JOHNSON TASK

to PJ4 decode the response by putting the subelements into STM directly (and marking the original sequence to show that it has been processed). The final production, PJ10, generates a response whenever a letter (<K>) shows up in STM, by evoking the operator SAY. The other two actions in PJ10 mark the letter occurrence as having been uttered, by converting a letter, say X, first into (X) and then into (SAID X).

Figure 12 shows the operation of this system, in which the responses are printed as <ITEM>: X, <ITEM>: Q, etc. The matter of interest here is what is and what is not represented. The code and the details of the decoding are represented, including the information in STM at any instant. The act of encoding from the stimulus into the nested set of elements is not represented. In addition, the act of learning, in which productions such as PJ20 are created, is not represented. With the lack of the learning and encoding, the response measure used by Neal Johnson (the probability of error at a given transition) falls through. Instead, the model reveals the internal coding by means of the pause structure in the response, assuming that the subject does not totally decode the response before uttering the letters, but does so as he goes.

Suppose the subject were asked to respond by giving the letters in pairs, i.e., XQ KF H (a task that Neal Johnson did not ask of his subjects). Two (non-exclusive) strategies are open to the subject (assuming he has no further access to the stimulus display). He can attempt a different decoding strategy, in which he accumulates at least two letters before he utters them. He can undertake to relearn the response in the new organization, so he can respond using the same simple decoding strategy. Within the present system both the more complex responding strategy and the recoding of the stimulus can be represented. Thus, Figure 13 gives the additional productions required for the pairwise responding and Figure 14 shows a run with the same paired

```
0.  STM: ((SR 1) NIL NIL NIL NIL NIL NIL)
PJ20  TRUE

1.  STM: ((SEQ X (SEQ Q K) (SEQ F H)) (SR 1) NIL NIL NIL NIL NIL)
PJ3  TRUE

5.  STM: (X (SEQ Q K) (SEQ F H) (OLD SEQ X (SEQ Q K) (SEQ F H)) (SR 1) NIL NIL)
PJ10  TRUE

<ITEM>: X
8.  STM: ((SAID X) (SEQ Q K) (SEQ F H) (OLD SEQ X (SEQ Q K) (SEQ F H)) (SR 1) NIL NIL)
PJ2  TRUE

11.  STM: (Q K (OLD SEQ Q K) (SAID X) (SEQ F H) (OLD SEQ X (OLD SEQ Q K) (SEQ F H)) (SR 1))
PJ10  TRUE

<ITEM>: Q
14.  STM: ((SAID Q) K (OLD SEQ Q K) (SAID X) (SEQ F H) (OLD SEQ X (OLD SEQ Q K) (SEQ F H)) (SR
1))
PJ10  TRUE

<ITEM>: K
17.  STM: ((SAID K) (SAID Q) (OLD SEQ Q K) (SAID X) (SEQ F H) (OLD SEQ X (OLD SEQ Q K) (SEQ F
H)) (SR 1))
PJ2  TRUE

20.  STM: (F H (OLD SEQ F H) (SAID K) (SAID Q) (OLD SEQ Q K) (SAID X))
PJ10  TRUE

<ITEM>: F
23.  STM: ((SAID F) H (OLD SEQ F H) (SAID K) (SAID Q) (OLD SEQ Q K) (SAID X))
PJ10  TRUE

<ITEM>: H
26.  STM: ((SAID H) (SAID F) (OLD SEQ F H) (SAID K) (SAID Q) (OLD SEQ Q K) (SAID X))
END: NO PD TRUE
```

FIGURE 12. BASIC OPERATION OF NJ SYSTEM

```
; NJ2: VARIATION ON NEAL JOHNSON'S CHUNKING TASK:
;       RESPOND IN PAIRS INDEPENDENT OF HOW LIST GIVEN.
;       E.G.:  IN:  1 - A BC D EFG
;              OUT:     AB CD EF G
;
;       (IDENTICAL TO NJ2.A03)
;       ASSUMES NJ ALREADY LOADED
;
DEFINE.PROCESSES!
;
SAY-NOTE: (ACTION SAY EMBED (<ITEM> ==> SAID <ITEM>))
;
DEFINE.SYMBOLS!
;
PJ8: ((OLD SEQ) AND (SEQ) ABS AND (END SEQ) ABS --> (END SEQ))
;
PJ11: (<ITEM> == <K> AND X0 == <K> --> SAY-NOTE (<ITEM> == X0)
        (NTC <ITEM>) SAY-NOTE)
PJ12: (<K> --> EMBED (<K> ==> HOLD <K>))
PJ13: ((HOLD X0) AND <K> --> (HOLD ==> OLD HOLD) X0)
PJ14: ((HOLD <ITEM>) AND (END SEQ) --> (HOLD ==> SAID) SAY)
;
PS2: (PJ4 PJ3 PJ2 PJ1 PJ0)
PS3: (PJ13 PJ11 PJ14 PJ12)
PS4: (PS3 PS2 PJ20)
;
"NJ2.A03 LOADED" RETURN.TO.TTY!
```

FIGURE 13. MODIFICATION OF NJ TO RESPOND TO A CODED STIMULUS IN PAIRS

```
8.  STM: ((SR 1) NIL NIL NIL NIL NIL NIL)
PJ28  TRUE

1.  STM: ((SEQ X (SEQ Q K) (SEQ F H)) (SR 1) NIL NIL NIL NIL NIL)
PJ3  TRUE

5.  STM: (X (SEQ Q K) (SEQ F H) (OLD SEQ X (SEQ Q K) (SEQ F H)) (SR 1) NIL NIL)
PJ12  TRUE

7.  STM: ((HOLD X) (SEQ Q K) (SEQ F H) (OLD SEQ X (SEQ Q K) (SEQ F H)) (SR 1) NIL NIL)
PJ2  TRUE

10.  STM: (Q K (OLD SEQ Q K) (HOLD X) (SEQ F H) (OLD SEQ X (OLD SEQ Q K) (SEQ F H)) (SR 1))
PJ13  TRUE

12.  STM: (X (OLD HOLD X) Q K (OLD SEQ Q K) (SEQ F H) (OLD SEQ X (OLD SEQ Q K) (SEQ F H)))
PJ11  TRUE

<ITEM>: X
<ITEM>: Q
22.  STM: ((SAID Q) (SAID X) (OLD HOLD X) K (OLD SEQ Q K) (SEQ F H) (OLD SEQ X (OLD SEQ Q K)
(SEQ F H)))
PJ12  TRUE

24.  STM: ((HOLD K) (SAID Q) (SAID X) (OLD HOLD X) (OLD SEQ Q K) (SEQ F H) (OLD SEQ X (OLD SEQ
Q K) (SEQ F H)))
PJ2  TRUE

27.  STM: (F H (OLD SEQ F H) (HOLD K) (SAID Q) (SAID X) (OLD HOLD X))
PJ13  TRUE

29.  STM: (K (OLD HOLD K) F H (OLD SEQ F H) (SAID Q) (SAID X))
PJ11  TRUE

<ITEM>: K
<ITEM>: F
39.  STM: ((SAID F) (SAID K) (OLD HOLD K) H (OLD SEQ F H) (SAID Q) (SAID X))
PJ12  TRUE

41.  STM: ((HOLD H) (SAID F) (SAID K) (OLD HOLD K) (OLD SEQ F H) (SAID Q) (SAID X))
PJ8  TRUE

42.  STM: ((END SEQ) (OLD SEQ F H) (HOLD H) (SAID F) (SAID K) (OLD HOLD K) (SAID Q))
PJ14  TRUE

<ITEM>: H
47.  STM: ((SAID H) (END SEQ) (OLD SEQ F H) (SAID F) (SAID K) (OLD HOLD K) (SAID Q))
END: NO PD TRUE
```

FIGURE 14. BEHAVIOR OF NJ2

associate as used in Figure 12. We have taken the action of PJ10 and made it into an operator, SAY-NOTE. Thus the main production is PJ11 which notes two letters and says the both. However, more is required. For one, a single letter left over at the end must be said. PJ14 takes care of this response. It is necessary to add to this something to recognize the end of sequence, to avoid inadvertent responding with an earlier single letter (e.g., at 5 in Figure 14). PJO takes care of this by putting in an (END SEQ) marker, which corresponds to the explicit awareness in STM that no more decoding is possible.

More important, if several chunks must be decoded to obtain a pair of letters, the order of the letters can be lost. To assure the correct order the system must temporarily reencode the letter in (HOLD <K>), use this code to reestablish the order, and then decode it again for responding with PJ11. This encoding and decoding can be followed in Figure 14, e.g., at 5-12 for the letter X. Thus, already with simple coding tasks additional phenomena arise when an explicit and operational control system is required.

Figure 15 shows another set of productions to be added to those of Figure 11 to create a new internal representation in pairs, rather than simple respond in pairs. Some, but not all, of the productions used in the other version (Figure 13) also occur in this one: analogs of P11 and P14, one to take care of pairs and the other to take core of the possibility of a single letter at the end. The same HOLD mechanism for keeping order is also used. But in addition there needs to be a production (PJ15*) to grow the representation as the groups are put together.

Figure 16 gives a run of this system, which ends up with the new element in STM. The relearning of the paired associate is not represented, just as it was not in the original version (Figure 11). However, this type of recoding corresponds to the cognitive encoding postulated by the Simon and Kotovsky model and by the other concept attainment schemes.

The two deficiencies of the present scheme -- the lack of a perceptual mechanism and the lack of a production-learning mechanism -- stem from entirely different sources. As mentioned earlier, the question of learning appears to be rather deep. We will not attempt to deal with it further here, but will simply select situations to work with that do not require it. The lack of a perceptual mechanism is due to the problem solving tasks not requiring one. Thus, we will attempt in the remainder of the paper to define the design issues for a perceptual mechanism for the production system and to construct an initial experimental version.

```
; NJR: 2ND VARIATION ON NEAL JOHNSON'S CHUNKING TASK:
;        RECODE IN PAIRS INDEPENDENT OF HOW LIST GIVEN.
;        E.G.:  IN:  1 - A BC D EFG
;               CODE: (SEQ A (SEQ B C) D (SEQ E F G))
;            RECODE: (SEQ (SEQ A B) (SEQ C D) (SEQ E F) G)
;            NO OUTPUT TO THE EXTERNAL ENVIRONMENT
;
;        (IDENTICAL TO NJR.A02)
;        ASSUMES NJ
;        INDEPENDENDENT OF NJ2, BUT USES SAME NAMES WHERE SAME
;
DEFINE.SYMBOLS!
;
PJ8: ((OLD SEQ) AND (SEQ) ABS AND (END SEQ) ABS --> (END SEQ))
;
PJ1*: ((GROUP X0) AND (NEW SEQ) --> (GROUP ==> OLD GROUP)
        (SEQ ===> SEQ X0))
PJ2*: ((GROUP X0) AND (NEW SEQ X1) --> (GROUP ==> OLD GROUP)
        (X1 ===> X1 X0))
PJ3*: ((GROUP X0) AND (NEW SEQ X2 X1) --> (GROUP ==> OLD GROUP)
        (X1 ===> X1 X0))
PJ4*: ((GROUP X0) AND (NEW SEQ X3 X2 X1) -->
        (GROUP ==> OLD GROUP) (X1 ===> X1 X0))
;
PJ11*: (X1 == <K> AND X2 == <K> --> (NTC X2) EMBED (NTC X1)
         EMBED (GROUP (SEQ X1 X2)))
PJ12: (<K> --> EMBED (<K> ==> HOLD <K>))
PJ13: ((HOLD X0) AND <K> --> (HOLD ==> OLD HOLD) X0)
PJ14*: ((HOLD X1) AND (END SEQ) --> (HOLD ==> OLD HOLD)
         (GROUP X1))
PJ15*: ((GROUP) AND (NEW SEQ) ABS --> (NEW SEQ))
;
PS2: (PJ4 PJ3 PJ2 PJ1 PJ8)
PS2*: (PJ4* PJ3* PJ2* PJ1*)
PS3: (PJ13 PJ11* PJ14* PJ15* PJ12)
PS4: (PS3 PS2* PS2 PJ20)
;
"NJR.A02 LOADED" RETURN.TO.TTY!
```

FIGURE 15. MODIFICATION OF NJ TO RECODE STIMULUS IN PAIRS

```
0.  STM: ((SR 1) NIL NIL NIL NIL NIL NIL NIL)
PJ20  TRUE

1.  STM: ((SEQ X (SEQ Q K) (SEQ F H)) (SR 1) NIL NIL NIL NIL NIL NIL)
PJ3  TRUE

5.  STM: (X (SEQ Q K) (SEQ F H) (OLD SEQ X (SEQ Q K) (SEQ F H)) (SR 1) NIL NIL NIL)
PJ12  TRUE

7.  STM: ((HOLD X) (SEQ Q K) (SEQ F H) (OLD SEQ X (SEQ Q K) (SEQ F H)) (SR 1) NIL NIL NIL)
PJ2  TRUE

10.  STM: (Q K (OLD SEQ Q K) (HOLD X) (SEQ F H) (OLD SEQ X (OLD SEQ Q K) (SEQ F H)) (SR 1) NIL)
PJ13  TRUE

12.  STM: (X (OLD HOLD X) Q K (OLD SEQ Q K) (SEQ F H) (OLD SEQ X (OLD SEQ Q K) (SEQ F H)) (SR
1))
PJ11*  TRUE

17.  STM: ((GROUP (SEQ X Q)) (X) (Q) (OLD HOLD X) K (OLD SEQ Q K) (SEQ F H) (OLD SEQ X (OLD
SEQ Q K) (SEQ F H)))
PJ15*  TRUE

18.  STM: ((NEW SEQ) (GROUP (SEQ X Q)) (X) (Q) (OLD HOLD X) K (OLD SEQ Q K) (SEQ F H))
PJ12  TRUE

20.  STM: ((HOLD K) (NEW SEQ) (GROUP (SEQ X Q)) (X) (Q) (OLD HOLD X) (OLD SEQ Q K) (SEQ F H))
PJ1*  TRUE

22.  STM: ((OLD GROUP (SEQ X Q)) (NEW SEQ (SEQ X Q)) (HOLD K) (X) (Q) (OLD HOLD X) (OLD SEQ Q
K) (SEQ F H))
PJ2  TRUE

25.  STM: (F H (OLD SEQ F H) (OLD GROUP (SEQ X Q)) (NEW SEQ (SEQ X Q)) (HOLD K) (X) (Q))
PJ13  TRUE

27.  STM: (K (OLD HOLD K) F H (OLD SEQ F H) (OLD GROUP (SEQ X Q)) (NEW SEQ (SEQ X Q)) (X))
PJ11*  TRUE

32.  STM: ((GROUP (SEQ K F)) (K) (F) (OLD HOLD K) H (OLD SEQ F H) (OLD GROUP (SEQ X Q)) (NEW
SEQ (SEQ X Q)))
PJ12  TRUE

34.  STM: ((HOLD H) (GROUP (SEQ K F)) (K) (F) (OLD HOLD K) (OLD SEQ F H) (OLD GROUP (SEQ X Q))
(NEW SEQ (SEQ X Q)))
PJ2*  TRUE

36.  STM: ((OLD GROUP (SEQ K F)) (NEW SEQ (SEQ X Q) (SEQ K F)) (HOLD H) (K) (F) (OLD HOLD K)
(OLD SEQ F H) (OLD GROUP (SEQ X Q)))
PJ8  TRUE

37.  STM: ((END SEQ) (OLD SEQ F H) (OLD GROUP (SEQ K F)) (NEW SEQ (SEQ X Q) (SEQ K F)) (HOLD
H) (K) (F) (OLD HOLD K))
PJ14*  TRUE

39.  STM: ((GROUP H) (OLD HOLD H) (END SEQ) (OLD SEQ F H) (OLD GROUP (SEQ K F)) (NEW SEQ (SEQ
X Q) (SEQ K F)) (K) (F))
PJ3*  TRUE

41.  STM: ((OLD GROUP H) (NEW SEQ (SEQ X Q) (SEQ K F) H) (OLD HOLD H) (END SEQ) (OLD SEQ F H)
(OLD GROUP (SEQ K F)) (K) (F))
END: NO PD TRUE
```

FIGURE 16. BEHAVIOR OF NJR

IV. A TASK FOR EXTENDING THE MODEL

To guide the development of a perceptual mechanism we need a specific task. This should be one that involves both perceptual and cognitive processing and in which the encoding performed by the subject is highly apparent. The data should be on single individuals, so that evidence as to the details of the response are not lost by aggregative data analysis.

The following series completion task used by Dave Klahr (Klahr and Wallace, 1970) appears suitable. The subject sees a display (from a slide projector) consisting of a linear array of picutres of schematic bottles. Each bottle has two attributes: color, with values of blue, green, red and yellow; and orientation, with values of up, down, left, right (taking the neck of the bottle as the head of a vector). The subject's task is to say what bottle will occur as the next element to the right of the linear array.

Figure 17 shows an example task along with the protocol of a male college undergraduate.* The colors of the bottles appear as labels here; actually they were bright colors on the slides. We have given two additional representation of the display, which will occur in this paper. The task (P15) was one of 23 tasks given during a single session to the subject. It yielded one of the most complex protocols (but it is also the only task that shows all colors and orientations on a single display).

---

* Klahr developed the task for work with children, but is also using it with adults. The protocol is from work by Michelene Chase, and I wish to thank her for letting me use it.

Series completion task (Klahr)
Protocol of run with subject LM, 20 Oct 78

16-th problem in a series of 23.

P15


```
GN  YL  GN  RD  BL  RD
RT  DN  RT  UP  LF  UP

(BTL GN RT)(BTL YL DN)(BTL GN RT)(BTL RD UP)(BTL BL LF)
(BTL RD UP)
```

B1 Ah, alternating, up down..
B2 I mean horizontal, vertical..
B3 type of pattern.
B4 Two greens surrounding a blue.
B5 Ah, two greens are laying on their side
B6 and then you've got two reds surrounding..
B7 or rather two greens surrounding a yellow..
B8 and the two reds surrounding a blue.
B9 And the blue..
B10 The reds are upright,
B11 as opposed to the greens,
B12 which are on their sides.
B13 Ah, since they are alternating,
B14 I would expect the next bottle to be laying on its side.
B15 Ah, since they're facing the same direction..
B16 No, there's a sequence,
B17 and then there's a second sequence.
B18 I would expect this..
B19 There's a three-patterned sequence,
B20 like a.. ah., bottle surrounding..
B21 two green surrounding a yellow
B22 both facing..
B23 the two green surrounding..
B24 the two surrounding colors facing in the same direction.
B25 I would expect another pattern like this.
B26 This time they should be facing..
B27 ah.. again towards the..
B28 Well, I'm not quite sure which direction they would be facing.
B29 I suppose they would be facing again towards the ah..
B30 A bottle laying on its side facing the right.
B31 Ah this time it should be yellow,
B32 since yellow has not surrounded a color yet.
B33 Next slide.

Figure 17: Protocol of Subject LM on series completion task

The basic feature of this task that recommends it for our purposes is its combination of perceptual and conceptual aspects. The subject perceives the display of bottles in some way. For example (at B1-B2), he sees the line in Figure 17 as an alternation of vertical and horizontal objects (thus abstracting from the distinction between up-down and right-left respectively). Also (B4), he sees patterns in which two colors "surround" another. But besides these perceptual organizations he symbolizes the stimulus so as to be able to reason about it (and talk about it, as well). For example, in B32 he makes a clear inference involving the non-occurrence of a given color in the prior part of the sequence. These reasonings are sufficiently similar to the sort of problem solving analysed by means of production systems so that we might expect a similar analysis to apply to it.

An interesting feature of S's behavior is that his first utterance in each task is a description of the display. A useful hypothesis is that this represents the way the S perceives the display and constitutes the starting point for further processing. Verification of this hypothesis depends mostly on the analysis of subsequent behavior after the initial statement. Here, we will simply assume it, and take the initial descriptions as evidence for intial perceptions. Figure 18 gives for each of the 23 tasks the display and the initial statements that were made by the subject.*

As the figure shows, the subject engages in a rich variety of descriptions. To give some idea of this we present in Figure 19 a grammar of the constructs used by the subject. We take E as the class of encodings. E can be any of 12 different expressions. In these expressions, E occurs recursively,

---

* We do not reproduce all of the protocols, since we will be concerned in this paper only with these first parts.

```
Series completion task (Klahr)
Protocol of run with subject LM, 20 Oct 70
.
Excerpt of first utterances for each task.
Appears to indicate initial perceptual view of stimulus.
...
P1      RD  RD  RD  GN  GN  GN
        RT  RT  RT  DN  DN  DN
.
B1      Three red bottles,
B2      three green bottles.
...
P2      GN  BL  GN  BL  GN  BL
        UP  UP  UP  LF  LF  LF
.
B1      Three bottles upright again
B2      followed by three that are not..
B3      that are horizontal.
...
P3      YL  BL  YL  BL  YL  BL
        DN  RT  DN  RT  DN  RT
.
B1      Alternating bottles,
B2      upright down.
B3      They're yellow, blue.
...
P4      BL  BL  YL  YL  BL  BL
        UP  UP  UP  RT  RT  RT
.
B1      Ah, two blue bottles,
B2      a yellow bottle,
B3      and a yellow bottle on its side.
...
P5      RD  RD  BL  BL  RD  RD
      LF  RT  RT  LF  RT  LF
.
B1      Ah, bottles facing opposite
B2      ah, then facing inward,
B3      changing colors.
...
P6      YL  YL  GN  GN  YL  YL
        RT  RT  DN  DN  RT  RT
.
B1      Green surrounded by two pair of yellow.
...
P7      BL  GN  BL  BL  GN  BL
        UP  UP  UP  DN  DN  DN
.
B1      Ah, sequence.
B2      Ah, now you've got one blue,
        (continues to enumerate each bottle's color)
...
P8      GN  RD  GN  GN  RD  GN
        LF  DN  LF  DN  LF  DN
.
B1      Alternating.
B2      Ah green always on its..
...
P9      YL  RD  YL  YL  RD  YL
        DN  DN  LF  LF  DN  DN
.
B1      Ah
```

Figure 18: First utterances of Subject LM on all tasks.

B2 you have two yellow in the middle
B3 all..
...
P10 GN RD GN GN RD GN
RT LF RT RT LF RT
.
B1 Ah.. green always facing towards the right.
B2 Red is always facing towards the left.
...
P11 RD BL RD GN YL GN
RT RT RT LF LF LF
.
B1 Ah.. three facing inward
B2 and then three facing it again.
...
P12 BL GN BL YL RD YL
DN LF DN LF DN LF
.
B1 Ah.. alternating up and laying on its side
...
P12B GN BL GN BL GN BL
UP RT UP LF DN LF
.
B1 Ah alternating.
B2 Ah blue green.
...
P13 RD YL RD GN BL GN
LF LF UP UP LF LF
.
B1 Ah, you have a sequence
B2 such that the pattern is two surrounding,
B3 two laying on their side facing left,
B4 surrounding two going upright
...
P14 RD YL RD BL GN BL
DN UP DN DN UP DN
.
B1 All upright.
...
P15 GN YL GN RD BL RD
RT DN RT UP LF UP
.
B1 Ah, alternating, up down..
B2 I mean horizontal, vertical..
B3 type of pattern.
...
P16 YL RD GN YL RD GN
LF LF LF UP UP UP
.
B1 Ah three laying on its side.
B2 three standing up.
B3 Both in the same pattern..
...
P16B BL BL BL RD RD RD
LF DN RT LF DN RT
.
B3 All right, you have blue surrounded by blue..
B4 They're going in opposite directions,
B5 such that it's a symmetric type of situation.
...
P17 BL YL RD BL YL
LF DN LF DN LF

Figure 18: (continued)

BI You have <h same sort of situation..
B2 You have an all horirental bottler, facing touard the- loft
B3 and the vortical bottlou aro do^.r*

P19 BL GN YL BL GN YL
LF UP LF LF UP LF

BI fin, it's all bottles horizontal are facing towards the left.

P20 YL BL RO YL BL RO YL BL RO
ON UP ON RT LF RT ON UP ON

BI Ph.. you have patterns of three horizontal..
B2 I mean vert ical..
B3 surrounding a block of three horizontal
B4 and then another ah block of three vertical again.

P28B GN RO GN YL BL YL GN RO GN
LF UP ON LF UP ON LF UP ON

BI Pl l right, you have patterns broken up
B2 such that there's a horizontal bottlo
B3 and two vertical bottles
B4 facing In tho opposite directions,

P21 BL GN YL BL GN YL
UP ON LF UP ON LF

BI Ph.. alternating bottles,
B2 two upright.

(End tasks)

Figure 18: First utterances of Subject LH on all tasks

	Pattern	Description	Number of occurrences
E:	SEQUENCE	No pattern to the sequence	3
	E1 + E2 + ...	E1 followed by E2 followed by ...	15
	[E1]	A repetition of E1	
	[E1 + E2]	E.g., an alternation of E1 and E2	
	E1 ≪ E2 ≫	E1 surrounds E2	4
	N E1 where N = 1, 2, ... ALL	A sequence of N E1's	24
	E1 & E2	E1 and E2, independently	5
	E1 ⊃ E2	Every E1 implies E2	6
	E1 AT L where L = ... MIDDLE ...	An E1 located at L	1
	CHANGE DIM where DIM = DIRECTION, COLOR	E differs along dimension DIM	2
	SAME DIM-PATTERN	E is same pattern with respect to dimension DIM	3
	COLOR-VALUE:		23
	RD	Red	2
	YL	Yellow	5
	GN	Green	7
	BL	Blue	9
	DIRECTION-VALUE:		43
	ABSOLUTE-DIRECTIONS:	Defined independently of unit	34
	HZ	Horizontal	15
	LF	Left	4
	RT	Right	1
	VT	Vertical	13
	UP	Up	0
	DN	Down	1
	RELATIVE-DIRECTIONS:	Defined relative to unit	7
	IN	Inward toward middle of unit	4
	OUT	Outward from middle of unit	2
	OPPOSITE	Opposite to other unit	1
	PATTERNED-DIRECTIONS:	Patterns on sequence of directions	2
	SYMMETRIC	Symmetric about middle	1
	BROKEN	Not symmetric or same	1

Figure 19. Grammar for empirical description of S's initial utterances.

since the subpattern also may be described. We have written these classes as E1 and E2 simply to make identification possible in the descriptive phrase given to the right of each type of encoding. Also, at the far right, we give the number of occurrences of the expression in the subjects utterances (as encoded in Figure 21, to be described).

A noteworthy feature is the elaboration on the notion of direction. In the stimulus itself there are simply four directions and four colors. The subject, however, imposes several distinct structures on this. One is to describe LF and RT at horizontal (HZ) and UP and DN as vertial (VT). The language the subject uses for this appears confusing, since he uses words like "upright" to mean vertical and "down" to sometimes mean horizontal and sometime DN. Figure 20 gives the translations. The reality of this extra level of organization is not in doubt. For example, in P17 the subject categorizes the bottles first as being horizontal or vertical and then, within this, as pointing in a particular direction (see Figure 18).

Besides the use of horizontal and vertical, the subject also describes directions in relative terms, as facing inward, or opposite, and even as being symmetric. Nothing like this elaboration occurs with colors, though there is some indirect indication that BL and GN are much more alike than are any of the other colors. For example, in P7, where the subject does not pick up any perceptual grouping at all, the entire sequence apparently looks like identical objects to a first approximation (note, that UP and DN both go into VT).

Word	Translation	Occurrences
upright	vertical (VT)	P2 P3 P13* P14 P21
up	vertical (VT)	P12 P16*
down	down (DN)	P17
down	horizontal (HZ)	P3
side, on side	horizontal (HZ)	P4 P8 P12 P13 P16

* Ambiguous whether signifies VT or UP

Figure 20. Words used with special meaning by S.

Figure 21 gives a quite faithful rendition of the subject's initial utterances in terms of the grammar. The subject's particular description is only one out of many possible encodings permitted by the grammar. The subject himself sometimes provides more than one code, as in P2 where he first codes the second group of three bottles as not the same direction as the first three, and then specifies this further as being horizontal. We use the slash to indicate subsequent encodings, the single slash (/) indicating a refinement of the whole and the double slash (//) indicating a refinement of one of the subunits. Also, the subject sometimes does not complete an encoding, which we indicate with three dots (...). This is not the same as the abstraction that occurs in all encodings. Here, the subject simply ignores all bottles after a given point. The usual reason is that the encoding fails (e.g., at P8 where only the first two GNs are horizontal).

It must be remembered that the responses catalogued in Figure 21 are the results of at least two encoding processes: (1) a perceptual-conceptual process that leads to the subject seeing the object with a given perceptual structure; and (2) the selection of descriptive phrases to be uttered in the linguistic response. There is a close dependence between these. For instance, one cannot (as in P9) talk of two yellows in the middle, without distinguishing the relation of middle. But one can (still in P9) group the entire sequence into (VT VT) (HZ HZ) (VT VT) and choose only to mention the (HZ HZ) group in the middle. However, they are still distinct processes and one many want to represent them separately in a model of the subject.

The role of the task and the behavioral data presented is to provide a concrete situation against which to extend our model and to define a perceptual system. Ultimately, of course, we wish to model this subject's behavior in detail, much as we have done with the cryptarithmetic task. But initially, as will be seen, we must be content to use it more as a foil and a guide.

P1	3RD + 3GN
P2	3VT + 3(CHANGE DIRECTION)//HZ
P3	[VT + HZ] / [YL + BL]
P4	2BL + 1YL + 1YL&HZ ...
P5	OUT + (IN + IN&(CHANGE COLOR))
P6	2YL ≪ 2GN ≫
P7	SEQUENCE / 1BL + 1GN + 1BL + 1BL + 1GN + 1BL
P8	[HZ + VT] / GN ⊃ HZ ...
P9	2YL LOC MIDDLE ...
P10	(GN ⊃ RT) & (RD ⊃ LF)
P11	3IN + 3IN
P12	[VT + HZ]
P12B	[VT + HZ] / [BL + GN]
P13	SEQUENCE / 2(HZ&LF) ≪ VT ≫
P14	ALL VT
P15	[HZ + VT]
P16	3HZ + 3VT // (SAME COLOR-PATTERN)
P16B	BL ≪ BL ≫ ... / OUT / SYMMETRIC
P17	SEQUENCE / (HZ ⊃ LF)&(VT ⊃ DN)
P19	HZ ⊃ LF
P20	3(SAME COLOR-PATTERN) / 3VT ≪ 3HZ ≫
P20B	N(SAME DIR-PATTERN) // BROKEN / HZ + 2VT // OPPOSITE
P21	[2VT +HZ]

Note:

...	Description not completed
E1/E2	E2 is a refinement or addition of E1
E1//E2	E2 is a refinement of a subpattern of E1

Figure 21. Initial patterns uttered by S.

V. A PERCEPTUAL MECHANISM

Our task, then, is to construct a (visual) perceptual system that fits with our production system and which produces the symbolized views of the stimulus as shown in Figure 18. Several conditions of this problem are not completely specified. What is a perceptual system? What is it to "fit" with a production system? What aspects of the productions system must be invariant -- PSG, PSG + GS1, PSG + GS1 + some parts of PS2? What is it to have a view of the display corresponding to S's initial statements? Still we should be able to recognize a plausible solution when we find one. Before describing a particular design, let us try to clarify these issues.

We may stipulate the overall structure shown in Figure 22. The perceptual mechanism sits between the STM and the external environment (the display, viewed as an external memory). At a particular moment the environment is in some possible state, i.e., there is a particular display of colored oriented bottles. The perceptual mechanism is also in some possible state, which has been determined partly by prior acts of perception, partly by instructions flowing from the STM to the perceptual mechanism, and partly by longer term adaptations and learnings. The momentary states of the display and the perceptual mechanism jointly determine the output delivered to the STM out of a set of possible outputs whose form is jointly determined by the structure of the perceptual mechanism and the STM.

Basic Issues

Much must be specified to determine an operational perceptual mechanism. The following list of considerations will narrow that specification and make the remainder of the design task more concrete. This considerations are responsive only in part to the known facts of visual functioning. Much remains open, though undoubtedly there are many existing studies that could determine matters further.

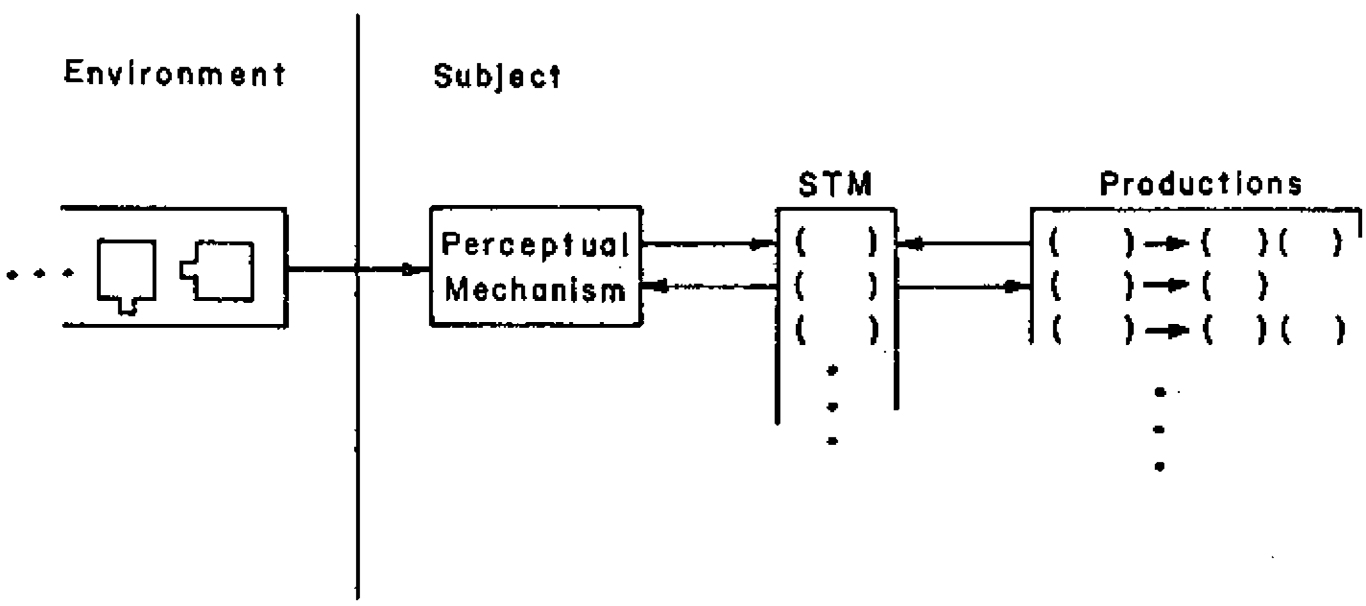


Figure 22: Overall structure of the system.

The discrete nature of perception. Vision, in tasks with a static display, operates by a sequence of discrete fixations. The duration of a fixation is 200-700 ms, which is of the order of the duration of a production, though on the upper side. There is evidence for units of perceptual attention both larger (groups of fixations) and smaller (attention movement within the field obtained from a single fixation). In any attempt to deal with the detail of a perceptual field (e.g., find all items of a given sort, read all words of test, etc.) there are fewer fixations than acts of directed perception. Thus, the functional unit can not be identified with the fixation, defined in terms of constancy of gaze direction. We can take each perceptual act to produce, ultimately, a symbolic structure (or a modification of a symbolic structure) in STM. This discrete nature of perception would be required by the discrete nature of the rest of the processing system, in any event.

The information taken from the display. The display, as a physical structure, is an infinite source of information. The perceptual mechanism selects (extracts, measures, abstracts, ...) from this a set of aspects on each perceptual act. It seems safe to consider this a discrete set of features. Although some pattern recognition schemes operate with spatial elements directly (template schemes), almost all reasonable recognition schemes involve the extraction of features at some stage. The set of features is fixed in the short run (i.e., the few hundred seconds of the experiment).

The locus of recognition. One extreme position is that the features themselves are symbolized (i.e., there are sensations) and made available in STM (i.e., to awareness). The recognition process then goes on in STM, so that further abstraction and classification occurs via productions. This makes all encoding conceptual, as that term was used earlier. It is an untenable position. At the other extreme, all recognition occurs within the perceptual mechanism, and only the final symbolized result becomes available in STM. This is not so much untenable as ambiguous, since it is not clear when to withhold the appellation of "recognition process" in describing the processing accomplished by productions. The following seems clear: (1) a recognition apparatus does reside in the perceptual mechanism; (2) features can be symbolized and made objects of awareness (i.e., become elements in STM)*; (3) inferences to new perceptual objects are also possible, especially in situation where perception is difficult; (4) conceptual recoding occurs routinely. The question of the back-flow from conceptually constructed perceptual objects to their subsequent perception is somewhat more open, though there is no doubt that perception itself can be affected by conceptual operations (e.g., setting expectations by verbal instructions.

The momentary state of the perceptual mechanism. Perception is selective, taking out of the display only certain information. The perceptual act is complex, consisting of an alternation of saccade and fixation, and within this additional attentional saccades and fixations. Thus, the specifications for the momentary state are correspondingly complex. Actually, the distinction between an eye movement system and a within-fixation system may not be functional at the level at which our model operates. The perceptual system may be defined in terms of perceptual acts which operate out of a memory (an iconic buffer), this memory being refreshed under local control by succeeding fixations of the eyes. In any event, it is problematical whether we must always continue to distinguish two systems of saccades and fixations, or can simply operate with a single system.

There does not appear to be much vision during the saccade itself, and the saccade appears to be determined (in direction and angular extent) prior to take-off. Thus, the momentary state can be divided into two parts:

E.g., we regularly discuss sensations.

that for perception at a fixation and that for the next saccade. However, the saccade itself appears often to be determined by the characteristics of the perceptual object sought, i.e., it has the characteristics of a search operation. In this respect it makes sense to consider the perceptual act as consisting of a saccade followed by intake at the subsequent fixation. In fact, often the appropriate unit appears to be a series of saccades and minimal fixations which end up in a fixation directed at the desired perceptual field. These sequences are often seen even in gross eye-movements, in which a long saccade is followed quickly by a very short, obviously corrective, saccade. But the existence of a continuous distribution of saccade lengths down to saccades of several minutes of arc also fits the same view.

There is ample evidence for the role of peripheral vision in general and it obviously plays a strong role in defining the next saccade. However, there seems to be little data at the level of detail required for our model.

We can at least list the items that should be considered in defining the perceptual state:

At fixation:

(1) The direction of gaze.

(2) Vergence.

(3) Light adaptation.

(4) The features to be noticed.

(5) Ordering of features and/or conditional cutoffs.

(6) The set of recognizable objects.

(7) Expectations for perceptual objects to be recognized.

(8) The grain of perception, i.e., the level of detail.

At saccade:

(9) The direction of the current gaze.

(10) The perceptual target desired.

(11) Knowledge of the peripheral field.

The list is not very operational and it is unclear how to make it so prior to setting out a particular perceptual mechanism.

Determinants of the perceptual state. Operation of the perceptual system implies that changes take place in the perceptual state from within the system itself. But in addition, all of the state variables (i.e., the items on the above list) must be subject to determination by systems outside the perceptual system itself, i.e., either by the display or by the remainder of the IPS. The key design issue is to specify, for each aspect of the momentary state, who determines it and with what time constant. The timing issue is critical. For example light adaptation is relatively slow and can be generally disregarded as a state variable in our task. New objects can be added to the stock of recognizables at rates consonant with the write operation into LTM (indeed such recognition later is a test of LTM retention). This is the control mechanism used in EPAM, as noted earlier. But what aspects can be set by symbolic expressions in STM? This is instruction on the time scale of a single perceptual act. Certainly, the next saccade is instructable (as in the verbal command "Look right!" or the perception of an arrow that points). But are short run (i.e., instantaneous) expectations set for each saccade? Are the features to be noticed set (or ordered) for each fixation, or does the cognitive system simply take what the perceptual system gives it, after telling it the rough direction in which to look? These and many other finer grained questions about who determines what appear not to be specifiable in terms of existing knowledge.

What is symbolized from a perception. After a perceptual act has taken place what is included in the symbolic expression (or expressions) produced in STM? Is there a recollection of the instructions given to the perceptual system? If there is some set of expectations (either of perceptual features or objects) is there knowledge of what was expected as well as what was found? If additional information is obtained about the object, is it remembered what was expected as well as what was observed, or is it all combined in a single result? Are the features used to recognize an object remembered, as well as the object? And so on.

Summary. We have listed a large number of considerations that enter into the specification of a perceptual system, though the list is not yet systematic. Our purpose in doing so is to make evident the range of design options. The particular system described in the next section results from one set of design decisions covering all the above issues. We do not understand this design space yet, nor the consequences of many of the specifications. Consequently, the presented perceptual system is simply a first cut.

LKE: A Particular Perceptual Mechanism

Given the background of the previous sections, we simply present the details of a particular subsystem, called LKE (for the Eth version of a system for looking). This system augments the basic production system, PSG, described in the earlier section.

The display for the series completion task is one dimensional, and can be conveniently modeled as a list. Figure 23 shows the display with the eyes located (>>) at the third bottle from the right, which has three features: the shape BTL, the color RD, and the orientation, RT. LKE assumes a single system of saccades and fixations, which therefore have a finer grain than gross eye movements. The interior logic design of the perceptual system is not modeled, so we talk indifferently of the eyes and of the locus of perceptual attention.

Initiation of perception may be under the control of either STM or the environment, though in a self-paced task such as series completion almost all of the initiation will come from STM. Thus there are perceptual operators, analogous to the operators in the cryptarithmetic task. LKE has two perceptual operators, LOOK.FOR and LOOK.AT. Each requires additional instructions from STM. LOOK.FOR requires a direction for the eye-movement (RIGHT, LEFT or STAY) and a perceptual object to guide the search in a display. For example, a typical instruction in STM might be:

(LOOK.FOR RIGHT (OBJ BTL))

This is an instruction to look to the right for an object with the shape of a bottle (i.e., in the present modeling, with the feature BTL). The operator LOOK.AT assumes that the eyes are already located at a proper place. It requires only that a perceptual object be given in its instruction, e.g.,

DSP1: (EDGE >> (BTL RD RT) (BTL RD RT) (BTL RD RT) (BTL GN DN) (BTL GN DN) (BTL GN DN) EDGE)

Figure 23: Display for task P1

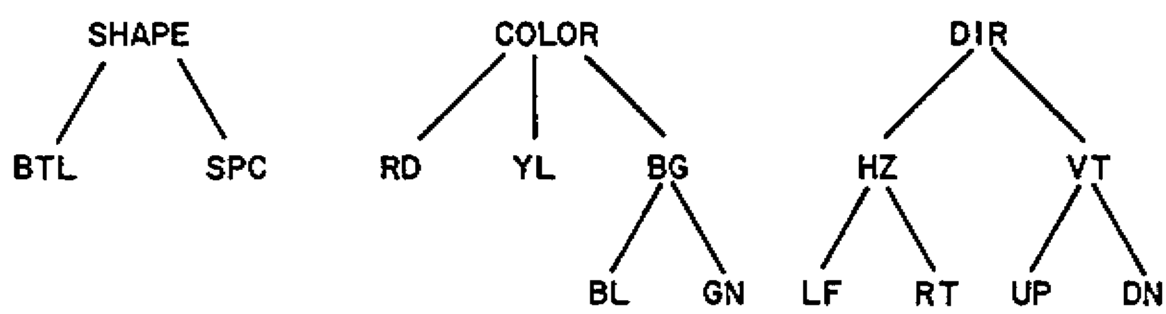


Figure 24: Hierarchy of features.

(LOOK.AT (OBJ BTL RD))

The result of a perceptual operation is the construction in STM of one (or more) symbolic structures giving what has been observed. For example, one might get

(OBS (OBJ BTL RD RT))

which is to say, that an object which was a red bottle pointing to the right was observed. Or one might get

(NOBS (OBJ BTL))

which is to say, that no object that was a bottle was found.

Perception often leaves open the possibility that additional observations may be possible. Thus, when doing (LOOK.FOR RIGHT (OBJ BTL)) in the situation of Figure 23 there are three more bottles that could be observed. LOOK.FOR will observe the first one, but if it were executed again it would obtain yet another observation. At some stage no more observations are possible. This is symbolized in an additional structure:

(END LOOK.FOR)

Thus the system creates positive knowledge of termination.

The features detectable by the perceptual system form a structured system of successive degrees of abstraction. The system for our subject is shown in Figure 24. There are three dimensions, SHAPE, COLOR and DIRECTION (DIR). For SHAPE there are only the two features, SPC and BTL. For COLOR, since the subject appears to see BL and GR as the same for some situations, an intermediate color, blue/green (BG), is stipulated (which is not to say that the subject has a color name for this, only that on occasion he does not discriminate between these colors). For DIRECTION the subject appears to make a discrimination between horizontal (HZ) and vertical (VT) and then within each between LF and RT, and UP and DN.

The control of the features to be detected and of the detail of these features is shared between the perceptual system and the STM. Thus, giving the perceptual object in the instructions determines much of what will be used. The function of LOOK.AT is to obtain additional detail about a perceived object. Thus the initiation of such a quest is under the control of STM. But what detail is seen is under the control of the perceptual system (consonant with the actual display). There is a fixed order to the observation along new dimensions and to the observation down the feature hierarchies of Figure 24. For instance, if the situation were as given in Figure 23 and the following instruction were given:

(LOOK.AT (OBJ BTL COLOR))

then the result would be:

(OBS (OBJ BTL RD))

If the instruction were:

(LOOK.AT (OBJ BTL RD))

then the result would be:

(OBS (OBJ BTL RD HZ))

And if, finally, the instruction were:

(LOOK.AT (OBJ BTL RD HZ))

the result would be:

(OBS (OBJ BTL RD RT)).

One aspect of the above example is misleading (and in an important way). Each successive observation with LOOK.AT does not generate a new element, (OBS (OBJ ...)). Rather, it constitutes an additional observation on an element that already exists (i.e., has been symbolized) in STM. Thus the three observations above constitute modifications of a single observation and the system does not believe that it has seen four distinct things, (OBJ BTL), (OBJ BTL RD), (OBJ BTL RD HZ) and (OBJ BTL RD RT). The instruction

(LOOK.AT X1) where X1 ← (OBJ BTL)

is also successively modified as X1 becomes modified and it serves to provide all the instructions for additional detail. (This has both advantages and disadvantages in terms of controlling perception.)

LKE has two kinds of perceptual objects, OBJ and SEQ. An OBJ is specified by a set of features and numerous examples have been given above. The features can be given at any level of detail, according to the hierarchies in Figure 24. A SEQ is a sequence of perceptual objects. For example:

(SEQ (OBJ BTL) (OBJ BTL))

is a sequence of two bottles. A sequence of two red bottles followed by a green bottle might be given as:

(SEQ (SEQ (OBJ BTL RD) (OBJ BTL RD)) (OBJ BTL GN))

Thus, recursive structures can be built up. However, the scheme in LKE does not take advantage of the redundancies in patterns. Thus, in terms of symbolization, it is as easy to perceive three different bottles as three identical ones:

(SEQ (OBJ BTL RD) (OBJ BTL BL) (OBJ BTL HZ))

(SEQ (OBJ BTL RD) (OBJ BTL RD) (OBJ BTL RD))

Which of the two will get constructed depends on the constructive processes and regular sequences may get built, whereas heterogeneous ones do not. But the difference is not reflected in the underlying representation.

The search in LOOK.FOR is for an absolute object, i.e., for the features as given in the symbolic element labeled (<POBJ.TYPE>), where <POBJ.TYPE>: (CLASS OBJ SEQ).

Any relativization to the local situation in the display is to be obtained by constructing the perceptual object that guides the search from the display itself (with LOOK.AT). In particular, the detection of differences in the display is not delegated to the perceptual mechanism.

Similarly, the construction of new perceptual objects, e.g., of (SEQ (OBJ BTL) (OBJ BTL)) from two occurrences of (OBJ BTL), is not determined by the perceptual mechanism autonemously, but is done by the formation of the new object in STM. Once such an object is formed, of course, it can be made part of a perceptual instruction and the display perceived in its terms.

Because of the requirements to simulate the environment in a discrete symbolic system, (i.e., in L* on a digital computer), there is a finite grain of the display. The display of Figure 23 precludes examining the curvature of the neck of the bottle, though this is possible on the slide, and subjects may even do so on occasion. More detail could beprovided if the characterization of the display in terms of a sequence of objects with three attributes did not seem sufficient. However, it would be necessary to extend the types of perceptual objects beyond OBJ and SEQ to cover the types of spatial relations possible e.g., to add WHOLE, whose components are attached parts, each of which is a perceptual object, plus and interfacing connection between parts.

Though the simulation provides a lower bound to the grain, it does not provide an upper bound. Thus, the eyes are located at an object in the display that represents the lowest level of detail. But the perceptual object that is seen from that locus may extend beyond the confines of that single object. SEQ does exactly this.

The structure of LKE, as it stands, permits certain patterns to be formed and not others. Thus, it put some limits in advance on the enterprise of obtaining the pattern descriptions made by the subject (in Figure 21). We give in Figure 25 a set of possibilities for the patterns that might be developed in a production system using LKE. Notice, for instance that the characterizations involving numbers, e.g., (3 RD) are replaced by extensive lists : (RD RD RD). One view of this is as a deficiency in LKE, to be rectified by a more adequate perceptual mechanism. A second possible view is that the additional encoding to obtain the codes of Figure 21 is done at the conceptual level in developing the linguistic utterance. In this case, the trip from (RD RD RD) to (3 RD) is made conceptually, i.e., by productions that count.

We have covered the essential design characteristics of LKE and the kinds of perceptual encodings it admits. Figure 26 gives a summary of these characteristics, which should be sufficient to understand the behavior of the system.

```
POTENTIAL BEHAVIOR IN PERFORMING ON TASK SCTF

P1: (SEQ (SEQ (OBJ BTL RD)(OBJ BTL RD)(OBJ BTL RD))
         (SEQ (OBJ BTL GN) (OBJ BTL GN)(OBJ BTL GN)))

P2: (SEQ (SEQ (OBJ BTL VT)(OBJ BTL VT)(OBJ BTL VT))
         (SEQ (OBJ BTL HZ)(OBJ BTL HZ)(OBJ BTL HZ)))
;
P3: (SEQ (SEQ (OBJ BTL VT)(OBJ BTL HZ))
         (SEQ (OBJ BTL VT)(OBJ BTL HZ))
         (SEQ (OBJ BTL VT)(OBJ BTL HZ)))

P4: (SEQ (OBJ BTL BL)(OBJ BTL BL))
    (OBJ BTL YL)
    (OBJ BTL YL HZ))
    UNCLEAR WHETHER SUPERORDINATE STRUCTURE IMPOSED

P5: (SEQ (SEQ (OBJ BTL RD)(OBJ BTL RD))
         (SEQ (OBJ BTL BL)(OBJ BTL BL))
         (SEQ (OBJ BTL RD)(OBJ BTL RD)))

P6: (SEQ (SEQ (OBJ BTL YL)(OBJ BTL YL))
         (SEQ (OBJ BTL GN)(OBJ BTL GN))
         (SEQ (OBJ BTL YL)(OBJ BTL YL)))

P7: NO ORGANIZATION ON FIRST PASS

P8: (SEQ (SEQ (OBJ BTL HZ)(OBJ BTL VT))
         (SEQ (OBJ BTL HZ)(OBJ BTL VT))
         (SEQ (OBJ BTL HZ)(OBJ BTL VT)))

P9: (SEQ (OBJ BTL YL)(OBJ BTL YL))

P10: NO SEQUENTIAL ORGANIZATION
     CANNOT CODE NON-SEQUENTIAL ORGANIZATION

P11: (SEQ (SEQ (OBJ BTL RT)(OBJ BTL RT)(OBJ BTL RT))
          (SEQ (OBJ BTL LF)(OBJ BTL LF)(OBJ BTL LF)))
     CANNOT CODE DIRECTION AS IN-OUT

P12: (SEQ (SEQ (OBJ BTL VT)(OBJ BTL HZ))
          (SEQ (OBJ BTL VT)(OBJ BTL HZ))
          (SEQ (OBJ BTL VT)(OBJ BTL HZ)))

P12B: (SEQ (SEQ (OBJ BTL VT)(OBJ BTL HZ))
           (SEQ (OBJ BTL VT)(OBJ BTL HZ))
           (SEQ (OBJ BTL VT)(OBJ BTL HZ)))

P13: NO ORGANIZATION ON FIRST PASS
     (SEQ (SEQ (OBJ BTL LF)(OBJ BTL LF))
          (SEQ (OBJ BTL UP)(OBJ BTL UP))
          (SEQ (OBJ BTL LF)(OBJ BTL LF)))

P14: (SEQ (OBJ BTL VT)(OBJ BTL VT)(OBJ BTL VT)
          (OBJ BTL VT)(OBJ BTL VT)(OBJ BTL VT))
     NOTE: DEPENDS ON WHETHER GROUPS OF 6 CAN BE BUILT UP
           IF NOT, THEN CAN'T CODE NON-SEQUENTIAL ORGANIZATION

P15: (SEQ (SEQ (OBJ BTL HZ)(OBJ BTL VT))
          (SEQ (OBJ BTL HZ)(OBJ BTL VT))
          (SEQ (OBJ BTL HZ)(OBJ BTL VT)))
```

Figure 25: Possible Encodings of Displays by System.

```
P16: (SEQ (SEQ (OBJ BTL HZ)(OBJ BTL HZ)(OBJ BTL HZ))
           (SEQ (OBJ BTL VT)(OBJ BTL VT)(OBJ BTL VT)))

P16B: CANNOT CODE

P17: NO ORGANIZATION ON FIRST PASS
     CANNOT CODE NON-SEQUENTIAL ORGANIZATION OF SECOND PASS

P19: CANNOT CODE NON-SEQUENTIAL ORGANIZATION

P20: (SEQ (SEQ (OBJ BTL YL)(OBJ BTL BL)(OBJ BTL RD))
           (SEQ (OBJ BTL YL)(OBJ BTL BL)(OBJ BTL RD))
           (SEQ (OBJ BTL YL)(OBJ BTL BL)(OBJ BTL RD)))

P20B: (SEQ (SEQ (OBJ BTL HZ)(OBJ BTL VT)(OBJ BTL VT))
            (SEQ (OBJ BTL HZ)(OBJ BTL VT)(OBJ BTL VT))
            (SEQ (OBJ BTL HZ)(OBJ BTL VT)(OBJ BTL VT)))

P21: (SEQ (SEQ (OBJ BTL VT)(OBJ BTL VT)(OBJ BTL HZ))
           (SEQ (OBJ BTL VT)(OBJ BTL VT)(OBJ BTL HZ)))
```

FIGURE 25. POSSIBLE ENCODINGS OF DISPLAYS BY SYSTEM

1. Each perceptual act is initiated by a perceptual operator, either LOOK.FOR or LOOK.AT.

2. Evocation of the perceptual operator is by the production system (interrupts from the environment are possible, but not modeled).

3. Each perceptual act requires an instruction from STM, which is taken to be the initial STM element.

4. Each perceptual act results in the creation of one or more STM elements (which enter STM just as do other elements created by productions) or by modification of elements accessible from the instruction element, e.g., the instruction element itself or the perceptual object it contains.

5. The perceptual mechanism retains the memory of the locus of perceptual attention (>>) in the display.

6. The perceptual mechanism retains the knowledge of the structure of perceptual features <FTR> and no operators currently exist for modifying this from STM or the production system.

7. A perceptual object <POBJ> is a symbolic structure of form (OBJ <FTR><FTR> ...) or (SEQ <POBJ> <POBJ> ...).

8. The perceptual system can ascertain if a given perceptual object is located in the environment at the point of attention (at >>). For (OBJ ...) it tests the features available at the point of attention. For (SEQ ...) it takes the point of attention as the leftmost point for the sequence of objects.

9. The perceptual system can add additional knowledge to a given perceptual object, either by increasing the detail of its given features <FTR> or by adding new dimensions to the perceptual object (for which added detail can then be obtained).

10. LOOK.AT requires a perceptual object. It adds an amount of additional knowledge as specified by the nature of the perceptual mechanism. (Currently it takes N steps of additional detail, N an externally settable parameter.) It does not create a new element in STM, except to indicate termination.

11. LOOK.FOR requires a perceptual object and a direction <EMD>. It looks for an object in the display along the given direction, taking the perceptual object as fixed and not adding more detail. It creates a new element in STM with the tag (OBS ...) if it finds the object and (NOBS ...) if it doesn't. It also creates a termination element (END LOOK.FOR) if there is not further to look in the given direction.

Figure 26: Summary of LKE

VI. BEHAVIOR OF THE SYSTEM

The system we have just created, consisting of PSG and LKE, is not in fact immensely complex compared (say) to many existing artificial intelligence systems. Still, we will only be able to afford the briefest look at its behavior, given the already extended character of this paper. We will not even be able to examine many aspects that are basic to its perceptual and cognitive behavior. In fact, we will set up a single simple system to illustrate how the two parts, the production system and the perceptual system, work together and to suggest some of the problems that exist.

Figure 27 presents the basic specification for behavior in the series completion task (SC3). It includes the various classes, the features and a display for a particular task. It also includes the basic goal manipulation system used for cryptarithmetic augmented by G12 and G13 to detect and execute perceptual instructions. For completeness, we have added definitions of the basic classes that are defined within LKE itself and are not specific to a task.

Figure 28 gives a short production system (SCP1) for the initial scan of the display. We assume that when the display is flashed on the screen an environment-initiated observation is produced:

(OBS NEW DISPLAY)

This is the trigger to scan the display and create the initial perceptual organization. This task is not goal directed in an explicit way, but is simply encoded in the set of productions as a direct reaction.

Production PD1 responds to the triggering stimulus and prepares for a left-to-right scan of the display by finding the left-hand edge. It is assumed that the subject has already oriented to the display and thus knows:

```
; SC3F: SERIES COMPLETION TASK (KLAHR)
;       REQUIRES LKEF, PSGF, U1F, DICTF, UTILF
;
DEFINE.SYMBOLS!
;
SC.CONTEXT SET.CONTEXT!
;
; MAKE NAMES AVAILABLE FOR USE IN SC.CONTEXT
RD* RD CHANGE.NAMES!
RT* RT CHANGE.NAMES!
;
; DEFINE CLASSES FOR USE IN PRODUCTION CONDITIONS
;
; DISPLAY ; CURRENT DISPLAY -- LIST OF OBJECTS
; BASIC CLASSES DEFINED IN LKEF, FOR REFERENCE
; <LKOPR>: (CLASS LOOK.AT LOOK.FOR) ; LOOK OPERATORS
; <EMD>: (CLASS LEFT RIGHT STAY) ; EYE MOVEMENT DIRECTIONS
; <OBS.TYPE>: (CLASS OBS OBS.AT NOBS) ;  OBSERVATION ELM TYPES
; <NEW.OBS>: (VAR) ; NAME FOR NEW OBSERVATION ELEMENT
; <END.OBS>: (VAR) ; NAME FOR END ELEMENT
; <POBJ.TYPE>: (CLASS OBJ SEQ) ; TYPES OF PERCEPTUAL OBJECTS
; <POBJ>: (<POBJ.TYPE>) ; PERCEPTUAL OBJECTS
; <NTC.TYPE>: (CLASS END MORE)
; LKT.ELM: (<NTC.TYPE> <LKOPR>)
; OBS.ELM: (<OBS.TYPE> <EMD> <POBJ>)
;
<COLOR>: (CLASS RD GN YL BL BK WH)
<SHAPE>: (CLASS SPC BTL)
<DIR>: (CLASS RT LF UP DN)
<G>:    (CLASS GOAL OLDG)
<SIG>: (CLASS * % + -)
<END>: (CLASS + -)
<COND>: (CLASS -COND +COND)
<OPR>: (CLASS)
<NTC>: (CLASS <LKOPR>)
<OBS>: (CLASS OBS OBS.AT)
;
DIM.LIST: (SHAPE COLOR DIR)
;
X1: (VAR)
X2: (VAR)
X3: (VAR)
X4: (VAR)
X5: (VAR)
;
RD: (FTR COLOR)
YL: (FTR COLOR)
BK: (FTR COLOR)
WH: (FTR COLOR)
BL: (FTR BG)
GN: (FTR BG)
BG: (FTR COLOR)
COLOR: (FTR)
;
UP: (FTR VT)
DN: (FTR VT)
VT: (FTR DIR)
LF: (FTR HZ)
RT: (FTR HZ)
HZ: (FTR DIR)
DIR: (FTR)
;
```

Figure 27: SC3F: Basic Specification of Series Completion Task.

```
BTL: (FTR SHAPE)
SPC: (FTR SHAPE)
SHAPE: (FTR)
EDGE: (SPC WH)
;
; DISPLAYS USED IN RUN WITH SUBJECT: LM, 20 OCT 70
;
DSP1: (EDGE (BTL RD RT)(BTL RD RT)(BTL RD RT)
        (BTL GN DN)(BTL GN DN)(BTL GN DN) EDGE)
;
; SEE FILE SCTF.AD8 FOR COMPLETE SET OF TASKS
;
;
G1: ((GOAL <END>) --> (GOAL ==> OLDG))
G2: ((GOAL *) ABS AND (GOAL %) --> (% ==> *))
G3: ((GOAL *) AND (GOAL *) --> (* ===> %))
G4: ((GOAL * <OPR>) --> <OPR>)
G5: ((GOAL * <COND>) AND (OLDG <END>) --> (<COND> ==>)
      (COND <COND> <END>))
G6: ((COND +COND +) AND (GOAL *) --> (COND ==> OLD COND)
      (* ===> +))
G7: ((COND -COND -) AND (GOAL *) --> (COND ==> OLD COND)
      (* ===> -))
G8: ((COND) AND (GOAL *) --> (COND ==> OLD COND))
G9: ((MORE) AND (GOAL *) --> (* ===> %))
G10: ((MORE <NTC>) AND (END <NTC>) --> (MORE ==> OLD MORE))
G11: ((GOAL *) ABS AND (GOAL <END> SOLVE) ABS -->
       (GOAL * SOLVE))
G12: ((<LKOPR>) --> <LKOPR>)
G13: ((<LKOPR>) AND (END <LKOPR>) --> (<LKOPR> ==> OLD <LKOPR>)
       (END ===> OLD END))
;
; PS1: TOTAL PRODUCTION SYSTEM
; PS2: PRODUCTION SYSTEM FOR TASK
; GS1: HIGH PRIORITY GOAL MANIPULATIONS
; GS2: BACK UP PRODUCTIONS
;
GS1: (G13 G1 G3 G10 G9 G5 G6 G7 G8 G4 G2)
GS2: (G11)
PS1: (GS1 PS2 GS2)
;
STM: (NIL NIL NIL NIL NIL NIL NIL NIL NIL)
;
"SC3F LOADED (NOTE: DIGITS ARE CHARS)" RETURN.TO.TTY!
```

FIGURE 27. SC3F: BASIC SPECIFICATION OF SERIES COMPLETION TASK

```
; SCP1: BASIC PRODUCTIONS FOR SERIES COMPLETION TASK (KLAHR)
;       REQUIRES SC3F, ETC.
;
;        (IDENTICAL TO SCPF.E03)
;
DEFINE.SYMBOLS!
;
PD1: ((OBS NEW DISPLAY) --> (LEFT (OBJ SPC)) LOOK.FOR)
;
PD2: ((OBS LEFT (OBJ SPC)) --> (OBS ==> OLD OBS)
      (LOOK.FOR RIGHT (OBJ BTL)))
;
PD3: ((OBS (OBJ BTL)) --> (OBS ==> OBS.AT) LOOK.AT)
;
PD4: ((<OBS> X1 == (<POBJ.TYPE>)) AND (<OBS> X1) -->
      (<OBS> ===> = <OBS>))
;
PD5: ((<OBS.TYPE>) AND (<OBS> X1 == (<POBJ.TYPE>)) AND
      (= <OBS> X1) AND (= <OBS> X1) ABS -->
      (<OBS> ===> OLD <OBS>)(= ====> OLD)(OBS (SEQ X1 X1)))
;
PD6: ((<OBS.TYPE>) AND (<OBS> X1 == (<POBJ.TYPE>)) AND
      (= <OBS> X1) AND (= <OBS> X1) AND (= <OBS> X1) ABS -->
      (<OBS> ===> OLD <OBS>)(= ====> OLD)(OBS (SEQ X1 X1 X1)))
;
PS2: (PD4 PD3 PD6 PD5 G12 PD2 PD1)
;
"SCPF.E03 LOADED" RETURN.TO.TTY!
```

FIGURE 28. SCP1:BASIC PRODUCTION SYSTEM FOR SERIES COMPLETION TASK

(1) the displ^r consists of sequences of bottles; (2) the field is bounded by the edge of the slide; (3) the relevant features are global aspects of the bottles; and (4) there is likely to be some sequential organization. This knowledge is embedded in the production system. How this was acquired as a function of instructions and preliminary examples is not touched here.

Production PD2 responds to the positioning of the eyes of the left-hand side by setting up an instruction to look for bottles by scanning to the right. This instruction defines the grain of the perceptual act.

Production PD3 responds to the detection of a bottle by looking at it somewhat closer. This will generate new detail about the bottle in the STM element that represents it. What detail is added is determined by the perceptual system itself and not by the instruction.

Production PD4 recognizes when two adjacent observed objects are the same and notes this fact by marking the second (the one that occurred earlier in time) with an equals (=). There must be a delay in actually organizing the perceived sequence, since subsequent objects have not yet been observed and they may effect the organization.

Productions PD5 and PD6 create perceptual organization by recognizing a sequence of perceived identical objects and encoding it as a SEQ. PD5 creates (SEQ XI XI) from a pair of identical objects; PD6 creates (SEQ XI XI XI) from a triple. The trigger for these actions is not only the requisite sequence of identical objects, but also that a distinct object has been perceived to bound the sequence. There is also a condition that no additional identical objects occur in STM, (<OBS> XI) ABS, which effectively provides a second boundary for the sequence.

In many of the productions (PD3, PD4, PD5, PD6) there is a modification of existing elements in STM by the replacement of one tag by another, e.g., (OBS ==> OLD OBS) or (OBS ==> OBS.AT). These modifications serve an essential control function to inhibit the repeated evocation of a production once a set of STM elements has sufficed to evoke it once. If a set of elements does evoke a production, then these same elements are capable of evoking it again (and again). What stops such repeated evocation in general is either (1) some change in these elements of (2) the new items created evoke a production prior in the ordering. Thus, many productions must take care to modify their evoking inputs.

Figure 29 gives a run of this system on P1, the first display. Tracing through the steps one can see each of the productions playing their role. For instance, G12 locates the first bottle (at 5), which is then examined (at 7) and seen to be red (RD). By 11 two red bottles have been seen whose identity can be noted by PD4. At 18 the observation of a bottle of a different color (BG) permits PD6 to create the sequence of three red bottles (at 21). A similar sequence now occurs with respect to the green bottles until the end of the sequence (NOBS) evokes PD6 at 32 to construct the second sequence. At 36 STM holds both sequences and there is nothing more to do.

Let us try this same system on some additional tasks. Figures 30 and 31 show the behavior of SCP1 on Problems P2 and P3. We give only the display and the final state of STM, from which can be inferred what must have happened. In P2 (Figure 30) we see that no organization at all developed. All elements were seen as the same, since only the color was perceived and that only at the level of BG. Contrariwise, the subject perceived this sequence as three vertical bottles followed by three bottles followed by three vertical ones (Figure 21).

```
DISPLAY: (EDGE (BTL RD RT) (BTL RD RT) (BTL RD RT) (BTL GN DN) (BTL GN DN) (BTL GN DN) EDGE)
0.  STM: ((OBS NEW DISPLAY) (GOAL * SOLVE) NIL NIL NIL NIL NIL NIL NIL)
PD1  TRUE

DISPLAY: (>> EDGE (BTL RD RT) (BTL RD RT) (BTL RD RT) (BTL GN DN) (BTL GN DN) (BTL GN DN) EDGE)
<NEW.OBS>: (OBS LEFT (OBJ SPC))
<END.OBS>: NIL
2.  STM: ((OBS LEFT (OBJ SPC)) (LEFT (OBJ SPC)) (OBS NEW DISPLAY) (GOAL * SOLVE) NIL NIL NIL
NIL NIL)
PD2  TRUE

4.  STM: ((LOOK.FOR RIGHT (OBJ BTL)) (OLD OBS LEFT (OBJ SPC)) (LEFT (OBJ SPC)) (OBS NEW
DISPLAY) (GOAL * SOLVE) NIL NIL NIL NIL)
G12  TRUE

DISPLAY: (EDGE >> (BTL RD RT) (BTL RD RT) (BTL RD RT) (BTL GN DN) (BTL GN DN) (BTL GN DN) EDGE)
<NEW.OBS>: (OBS RIGHT (OBJ BTL))
<END.OBS>: NIL
5.  STM: ((OBS RIGHT (OBJ BTL)) (LOOK.FOR RIGHT (OBJ BTL)) (OLD OBS LEFT (OBJ SPC)) (LEFT (OBJ
SPC)) (OBS NEW DISPLAY) (GOAL * SOLVE) NIL NIL NIL)
PD3  TRUE

DISPLAY: (EDGE >> (BTL RD RT) (BTL RD RT) (BTL RD RT) (BTL GN DN) (BTL GN DN) (BTL GN DN) EDGE)
<NEW.OBS>: NIL
<END.OBS>: NIL
7.  STM: ((OBS.AT RIGHT (OBJ BTL RD)) (LOOK.FOR RIGHT (OBJ BTL)) (OLD OBS LEFT (OBJ SPC))
(LEFT (OBJ SPC)) (OBS NEW DISPLAY) (GOAL * SOLVE) NIL NIL NIL)
G12  TRUE

DISPLAY: (EDGE (BTL RD RT) >> (BTL RD RT) (BTL RD RT) (BTL GN DN) (BTL GN DN) (BTL GN DN) EDGE)
<NEW.OBS>: (OBS RIGHT (OBJ BTL))
<END.OBS>: NIL
8.  STM: ((OBS RIGHT (OBJ BTL)) (LOOK.FOR RIGHT (OBJ BTL)) (OBS.AT RIGHT (OBJ BTL RD)) (OLD
OBS LEFT (OBJ SPC)) (LEFT (OBJ SPC)) (OBS NEW DISPLAY) (GOAL * SOLVE) NIL NIL)
PD3  TRUE

DISPLAY: (EDGE (BTL RD RT) >> (BTL RD RT) (BTL RD RT) (BTL GN DN) (BTL GN DN) (BTL GN DN) EDGE)
<NEW.OBS>: NIL
<END.OBS>: NIL
10.  STM: ((OBS.AT RIGHT (OBJ BTL RD)) (LOOK.FOR RIGHT (OBJ BTL)) (OBS.AT RIGHT (OBJ BTL RD))
(OLD OBS LEFT (OBJ SPC)) (LEFT (OBJ SPC)) (OBS NEW DISPLAY) (GOAL * SOLVE) NIL NIL)
PD4  TRUE

11.  STM: ((OBS.AT RIGHT (OBJ BTL RD)) (= OBS.AT RIGHT (OBJ BTL RD)) (LOOK.FOR RIGHT (OBJ
BTL)) (OLD OBS LEFT (OBJ SPC)) (LEFT (OBJ SPC)) (OBS NEW DISPLAY) (GOAL * SOLVE) NIL NIL)
G12  TRUE

DISPLAY: (EDGE (BTL RD RT) (BTL RD RT) >> (BTL RD RT) (BTL GN DN) (BTL GN DN) (BTL GN DN) EDGE)
<NEW.OBS>: (OBS RIGHT (OBJ BTL))
<END.OBS>: NIL
12.  STM: ((OBS RIGHT (OBJ BTL)) (LOOK.FOR RIGHT (OBJ BTL)) (OBS.AT RIGHT (OBJ BTL RD)) (=
OBS.AT RIGHT (OBJ BTL RD)) (OLD OBS LEFT (OBJ SPC)) (LEFT (OBJ SPC)) (OBS NEW DISPLAY) (GOAL *
SOLVE) NIL)
PD3  TRUE

DISPLAY: (EDGE (BTL RD RT) (BTL RD RT) >> (BTL RD RT) (BTL GN DN) (BTL GN DN) (BTL GN DN) EDGE)
<NEW.OBS>: NIL
<END.OBS>: NIL
14.  STM: ((OBS.AT RIGHT (OBJ BTL RD)) (LOOK.FOR RIGHT (OBJ BTL)) (OBS.AT RIGHT (OBJ BTL RD))
(= OBS.AT RIGHT (OBJ BTL RD)) (OLD OBS LEFT (OBJ SPC)) (LEFT (OBJ SPC)) (OBS NEW DISPLAY)
(GOAL * SOLVE) NIL)
PD4  TRUE
```

Figure 29: Run of SCP1 on Task P1

```
15.  STM: ((OBS.AT RIGHT (OBJ BTL RD)) (- OBS.AT RIGHT (OBJ BTL RD)) (LOOK.FOR RIGHT (OBJ
BTL)) (- OBS.AT RIGHT (OBJ BTL RD)) (OLD OBS LEFT (OBJ SPC)) (LEFT (OBJ SPC)) (OBS NEW
DISPLAY) (GOAL ≠ SOLVE) NIL)
G12  TRUE

DISPLAY: (EDGE (BTL RD RT) (BTL RD RT) (BTL RD RT) >> (BTL GN DN) (BTL GN DN) (BTL GN DN) EDGE)
<NEW.OBS>: (OBS RIGHT (OBJ BTL))
<END.OBS>: NIL
16.  STM: ((OBS RIGHT (OBJ BTL)) (LOOK.FOR RIGHT (OBJ BTL)) (OBS.AT RIGHT (OBJ BTL RD)) (-
OBS.AT RIGHT (OBJ BTL RD)) (- OBS.AT RIGHT (OBJ BTL RD)) (OLD OBS LEFT (OBJ SPC)) (LEFT (OBJ
SPC)) (OBS NEW DISPLAY) (GOAL ≠ SOLVE))
PD3  TRUE

DISPLAY: (EDGE (BTL RD RT) (BTL RD RT) (BTL RD RT) >> (BTL GN DN) (BTL GN DN) (BTL GN DN) EDGE)
<NEW.OBS>: NIL
<END.OBS>: NIL
18.  STM: ((OBS.AT RIGHT (OBJ BTL BG)) (LOOK.FOR RIGHT (OBJ BTL)) (OBS.AT RIGHT (OBJ BTL RD))
(- OBS.AT RIGHT (OBJ BTL RD)) (- OBS.AT RIGHT (OBJ BTL RD)) (OLD OBS LEFT (OBJ SPC)) (LEFT
(OBJ SPC)) (OBS NEW DISPLAY) (GOAL ≠ SOLVE))
PD6  TRUE

21.  STM: ((OBS (SEQ (OBJ BTL RD) (OBJ BTL RD) (OBJ BTL RD))) (OBS.AT RIGHT (OBJ BTL BG)) (OLD
OBS.AT RIGHT (OBJ BTL RD)) (OLD OBS.AT RIGHT (OBJ BTL RD)) (- OBS.AT RIGHT (OBJ BTL RD))
(LOOK.FOR RIGHT (OBJ BTL)) (OLD OBS LEFT (OBJ SPC)) (LEFT (OBJ SPC)) (OBS NEW DISPLAY))
G12  TRUE

DISPLAY: (EDGE (BTL RD RT) (BTL RD RT) (BTL RD RT) (BTL GN DN) >> (BTL GN DN) (BTL GN DN) EDGE)
<NEW.OBS>: (OBS RIGHT (OBJ BTL))
<END.OBS>: NIL
22.  STM: ((OBS RIGHT (OBJ BTL)) (LOOK.FOR RIGHT (OBJ BTL)) (OBS (SEQ (OBJ BTL RD) (OBJ BTL
RD) (OBJ BTL RD))) (OBS.AT RIGHT (OBJ BTL BG)) (OLD OBS.AT RIGHT (OBJ BTL RD)) (OLD OBS.AT
RIGHT (OBJ BTL RD)) (- OBS.AT RIGHT (OBJ BTL RD)) (OLD OBS LEFT (OBJ SPC)) (LEFT (OBJ SPC)))
PD3  TRUE

DISPLAY: (EDGE (BTL RD RT) (BTL RD RT) (BTL RD RT) (BTL GN DN) >> (BTL GN DN) (BTL GN DN) EDGE)
<NEW.OBS>: NIL
<END.OBS>: NIL
24.  STM: ((OBS.AT RIGHT (OBJ BTL BG)) (LOOK.FOR RIGHT (OBJ BTL)) (OBS (SEQ (OBJ BTL RD) (OBJ
BTL RD) (OBJ BTL RD))) (OBS.AT RIGHT (OBJ BTL BG)) (OLD OBS.AT RIGHT (OBJ BTL RD)) (OLD OBS.AT
RIGHT (OBJ BTL RD)) (- OBS.AT RIGHT (OBJ BTL RD)) (OLD OBS LEFT (OBJ SPC)) (LEFT (OBJ SPC)))
PD4  TRUE

25.  STM: ((OBS.AT RIGHT (OBJ BTL BG)) (- OBS.AT RIGHT (OBJ BTL BG)) (LOOK.FOR RIGHT (OBJ
BTL)) (OBS (SEQ (OBJ BTL RD) (OBJ BTL RD) (OBJ BTL RD))) (OLD OBS.AT RIGHT (OBJ BTL RD)) (OLD
OBS.AT RIGHT (OBJ BTL RD)) (- OBS.AT RIGHT (OBJ BTL RD)) (OLD OBS LEFT (OBJ SPC)) (LEFT (OBJ
SPC)))
G12  TRUE

DISPLAY: (EDGE (BTL RD RT) (BTL RD RT) (BTL RD RT) (BTL GN DN) (BTL GN DN) >> (BTL GN DN) EDGE)
<NEW.OBS>: (OBS RIGHT (OBJ BTL))
<END.OBS>: NIL
26.  STM: ((OBS RIGHT (OBJ BTL)) (LOOK.FOR RIGHT (OBJ BTL)) (OBS.AT RIGHT (OBJ BTL BG)) (-
OBS.AT RIGHT (OBJ BTL BG)) (OBS (SEQ (OBJ BTL RD) (OBJ BTL RD) (OBJ BTL RD))) (OLD OBS.AT
RIGHT (OBJ BTL RD)) (OLD OBS.AT RIGHT (OBJ BTL RD)) (- OBS.AT RIGHT (OBJ BTL RD)) (OLD OBS
LEFT (OBJ SPC)))
PD3  TRUE

DISPLAY: (EDGE (BTL RD RT) (BTL RD RT) (BTL RD RT) (BTL GN DN) (BTL GN DN) >> (BTL GN DN) EDGE)
<NEW.OBS>: NIL
<END.OBS>: NIL
28.  STM: ((OBS.AT RIGHT (OBJ BTL BG)) (LOOK.FOR RIGHT (OBJ BTL)) (OBS.AT RIGHT (OBJ BTL BG))
```

Figure 29: (continued)

```
(= OBS.AT RIGHT (OBJ BTL BG)) (OBS (SEQ (OBJ BTL RD) (OBJ BTL RD) (OBJ BTL RD))) (OLD OBS.AT
RIGHT (OBJ BTL RD)) (OLD OBS.AT RIGHT (OBJ BTL RD)) (= OBS.AT RIGHT (OBJ BTL RD)) (OLD OBS
LEFT (OBJ SPC)))
PD4  TRUE

29.  STM: ((OBS.AT RIGHT (OBJ BTL BG)) (= OBS.AT RIGHT (OBJ BTL BG)) (LOOK.FOR RIGHT (OBJ
BTL)) (= OBS.AT RIGHT (OBJ BTL BG)) (OBS (SEQ (OBJ BTL RD) (OBJ BTL RD) (OBJ BTL RD))) (OLD
OBS.AT RIGHT (OBJ BTL RD)) (OLD OBS.AT RIGHT (OBJ BTL RD)) (= OBS.AT RIGHT (OBJ BTL RD)) (OLD
OBS LEFT (OBJ SPC)))
G12  TRUE

DISPLAY: (EDGE (BTL RD RT) (BTL RD RT) (BTL RD RT) (BTL GN DN) (BTL GN DN) (BTL GN DN) >> EDGE)
<NEW.OBS>: (NOBS RIGHT (OBJ BTL))
<END.OBS>: (END LOOK.FOR)
30.  STM: ((END LOOK.FOR) (NOBS RIGHT (OBJ BTL)) (LOOK.FOR RIGHT (OBJ BTL)) (OBS.AT RIGHT (OBJ
BTL BG)) (= OBS.AT RIGHT (OBJ BTL BG)) (= OBS.AT RIGHT (OBJ BTL BG)) (OBS (SEQ (OBJ BTL RD)
(OBJ BTL RD) (OBJ BTL RD))) (OLD OBS.AT RIGHT (OBJ BTL RD)) (OLD OBS.AT RIGHT (OBJ BTL RD)))
G13  TRUE

32.  STM: ((OLD LOOK.FOR RIGHT (OBJ BTL)) (OLD END LOOK.FOR) (NOBS RIGHT (OBJ BTL)) (OBS.AT
RIGHT (OBJ BTL BG)) (= OBS.AT RIGHT (OBJ BTL BG)) (= OBS.AT RIGHT (OBJ BTL BG)) (OBS (SEQ (OBJ
BTL RD) (OBJ BTL RD) (OBJ BTL RD))) (OLD OBS.AT RIGHT (OBJ BTL RD)) (OLD OBS.AT RIGHT (OBJ BTL
RD)))
PD6  TRUE

35.  STM: ((OBS (SEQ (OBJ BTL BG) (OBJ BTL BG) (OBJ BTL BG))) (NOBS RIGHT (OBJ BTL)) (OLD
OBS.AT RIGHT (OBJ BTL BG)) (OLD OBS.AT RIGHT (OBJ BTL BG)) (= OBS.AT RIGHT (OBJ BTL BG)) (OLD
LOOK.FOR RIGHT (OBJ BTL)) (OLD END LOOK.FOR) (OBS (SEQ (OBJ BTL RD) (OBJ BTL RD) (OBJ BTL
RD))) (OLD OBS.AT RIGHT (OBJ BTL RD)))
G11  TRUE

36.  STM: ((GOAL * SOLVE) (OBS (SEQ (OBJ BTL BG) (OBJ BTL BG) (OBJ BTL BG))) (NOBS RIGHT (OBJ
BTL)) (OLD OBS.AT RIGHT (OBJ BTL BG)) (OLD OBS.AT RIGHT (OBJ BTL BG)) (= OBS.AT RIGHT (OBJ BTL
BG)) (OLD LOOK.FOR RIGHT (OBJ BTL)) (OLD END LOOK.FOR) (OBS (SEQ (OBJ BTL RD) (OBJ BTL RD)
(OBJ BTL RD))))
END: NO PD TRUE
```

FIGURE 29. RUN OF SCP1 ON TASK P1

DISPLAY: (EDGE (BTL GN UP) (BTL BL UP) (BTL GN UP) (BTL BL LF) (BTL GN LF) (BTL BL LF) EDGE)

31. STM: ((GOAL * SOLVE) (OLD LOOK.FOR RIGHT (OBJ BTL)) (OLD END LOOK.FOR) (NOBS RIGHT (OBJ BTL)) (OBS.AT RIGHT (OBJ BTL BG)) (= OBS.AT RIGHT (OBJ BTL BG)) (= OBS.AT RIGHT (OBJ BTL BG)) (= OBS.AT RIGHT (OBJ BTL BG)) (= OBS.AT RIGHT (OBJ BTL BG)))

FIGURE 30. RUN OF SCP1 ON TASK P2

DISPLAY: (EDGE (BTL YL DN) (BTL BL RT) (BTL YL DN) (BTL BL RT) (BTL YL DN) (BTL BL RT) EDGE)

34. STM: ((GOAL * SOLVE) (OLD LOOK.FOR RIGHT (OBJ BTL)) (OLD END LOOK.FOR) (NOBS RIGHT (OBJ BTL)) (OBS.AT RIGHT (OBJ BTL BG)) (OBS.AT RIGHT (OBJ BTL YL)) (OBS (SEQ (OBJ BTL BG) (OBJ BTL BG))) (OBS (SEQ (OBJ BTL YL) (OBJ BTL YL))) (OLD OBS.AT RIGHT (OBJ BTL BG)))

FIGURE 31. RUN OF SCP1 ON TASK P3

In P3 a quite different departure occurred: the system put some yellows together and some blues together, thus constructing an organization that violated the sequential order of the objects. The subject, on the other hand, perceived P3 as a sequence of three pairs, [VT + HZ] (Figure 21).

The sources of these difficulties are not hard to spot. The perceptual system only observes a single additional dimension, whereas the subject obviously is aware of both dimensions of variation. Selection on dimensions of perception is always necessary, and ultimately the relevant dimensions for a task series must become encoded into the STM element that gets formed to look at the display (as provided in SCP1 by PD3). The inappropriate grouping in problem P3 arises simply because SCP1 has no productions that are sensitive to forms other than runs of identical elements.

In addition to these two discrepancies, some other aspects of the system's behavior should be noted. First, we are not having the system actually produce an output (as we did, for example, in the Neal Johnson task) and the encoding of the perceptual objects for output is not given. Thus, in Figure 29, the conversion from:

(SEQ (OBJ BTL BG) (OBJ BTL BG) (OBJ BTL BG))

to a statement of a sequence of three green bottles is still to be made. The productions to do this are not difficult to envision, but it should be noted that they require an additional look at the stimulus (with LOOK.AT) in order to disambiguate BG into GN. A second feature to notice is that the subsequences are simply left in STM at the end (in both PI and P3). The subject organizes these into a single perception of the stimulus. Again, this is due to the lack of productions that are sensitive to this final need for organization.

Figure 32 shows a modified production system (SCP2) that attempts to respond to a number of these considerations. We have changed the number of dimensions looked at when adding detail (by LOOK.AT) from one to two. This does not show up in the production system, since it is a feature of the perceptual system. We have added productions PD7 and PD8 to be sensitive to alternations. PD7 recognizes the repetition of an element. Thus, it notes X Y X as indicating an organization into X (Y X). PD8 uses an existing organization to build up additional ones, so that it sees Y X (Y X) as (Y X) (Y X). Normally the occurrence of Y X would appear to be simply two distinct elements.

It might be thought that PD8 was not needed, since X Y X (Y X) would get transformed to X (Y X) (Y X) in any event by PD7. Indeed this is true -- until the last pair occurs, when there is no following X to force the organization. Basically, there must be some reason why Y X looks like a group. Initially it is the fact that following elements repeat (PD7); but eventually it must be that previous elements repeat (PD8). Thus some form of expectation must occur.

We have also added productions PD9 and PD10 in SCP2 to group together whatever organization has occurred by the end of the stimulus. However, we have not introduced the second layer of responding, given the perceived organization, e.g., to say "3 green." Thus, the output of interest of the system is simply the final state of STM.

Figures 33, 34 and 35 show the results of these modification on P1, P2 and P3 respectively. P1 and P2 now look fine. However, we failed to obtain the intended result in P3. It did obtain the subsequences, as desired, but it then put two of them together into a higher sequence, rather than all three; and then followed this by the use of PD9 to create an organization of the form:

(((YX)(YX))(YX))

The reason for this is interesting. The strategy of the SCP1-SCP2 system is to detect organization by delaying until a boundary occurs. The productions PD5 and PD6 respond to a general boundary (<OBS.TYPE>), since what is important is that

```
; SCP2; MODIFICATION OF SCP1
;       REQUIRES SC3F, ETC. (I.E., REPLACES SCP1)
;
;       (IDENTICAL TO SCPF.E04)
;       ADDS P7, P8 FOR ALTERNATIONS
;       ADDS P9, P10 FOR FINAL GROUPING
;       GOES TO 2 DIMENSIONS OF ADDED DETAIL PER TRY
;
DEFINE.SYMBOLS!
;
PD1: ((OBS NEW DISPLAY) --> (LEFT (OBJ SPC)) LOOK.FOR)
;
PD2: ((OBS LEFT (OBJ SPC)) --> (OBS ==> OLD OBS)
       (LOOK.FOR RIGHT (OBJ BTL)))
;
PD3: ((OBS (OBJ BTL)) --> (OBS ==> OBS.AT) LOOK.AT)
;
PD4: ((<OBS> X1 == (<POBJ.TYPE>)) AND (<OBS> X1) -->
       (<OBS> ===> = <OBS>))
;
PD5: ((<OBS.TYPE>) AND (<OBS> X1 == (<POBJ.TYPE>)) AND
       (= <OBS> X1) AND (= <OBS> X1) ABS -->
       (<OBS> ===> OLD <OBS>)(= ====> OLD)(OBS (SEQ X1 X1)))
;
PD6: ((<OBS.TYPE>) AND (<OBS> X1 == (<POBJ.TYPE>)) AND
       (= <OBS> X1) AND (= <OBS> X1) AND (= <OBS> X1) ABS -->
       (<OBS> ===> OLD <OBS>)(= ====> OLD)(OBS (SEQ X1 X1 X1)))
;
PD7: ((<OBS> X1 == (<POBJ.TYPE>)) AND
       (<OBS> X2 == (<POBJ.TYPE>)) AND (<OBS> X1) -->
       (<OBS> ===> OLD <OBS>)(<OBS> ====> OLD <OBS>)
       (OBS (SEQ X1 X2)))
;
PD8: ((<OBS> X1 == (<POBJ.TYPE>)) AND
       (<OBS> X2 == (<POBJ.TYPE>)) AND (OBS (SEQ X2 X1)) -->
       (<OBS> ==> OLD <OBS>)(<OBS> ===> OLD <OBS>)
       (OBS (SEQ X2 X1)))
;
PD9: ((NOBS) AND (<OBS> X1 == (<POBJ.TYPE>)) AND
       (<OBS> X2 == (<POBJ.TYPE>)) --> (<OBS> ===> OLD <OBS>)
       (<OBS> ====> OLD <OBS>)(OBS (SEQ X2 X1)))
;
PD10: ((NOBS) AND (<OBS> X1 == (<POBJ.TYPE>)) AND
        (<OBS> X2 == (<POBJ.TYPE>)) AND
        (<OBS> X3 == (<POBJ.TYPE>)) --> (<OBS> ===> OLD <OBS>)
        (<OBS> ====> OLD <OBS>)(OBS (SEQ X3 X2 X1)))
;
PS2: (PD7 PD4 PD3 PD8 PD6 PD5 C12 PD10 PD9 PD2 PD1)
;
"SCPF.E04 LOADED" RETURN.TO.TTY!
```

FIGURE 32. SCP2: MODIFIED SYSTEM FOR SERIES COMPLETION TASK

DISPLAY: (EDGE (BTL RD RT) (BTL RD RT) (BTL RD RT) (BTL GN DN) (BTL GN DN) (BTL GN DN) EDGE)

39. STM: ((GOAL * SOLVE) (OBS (SEQ (SEQ (OBJ BTL RD HZ) (OBJ BTL RD HZ) (OBJ BTL RD HZ)) (SEQ (OBJ BTL BG VT) (OBJ BTL BG VT) (OBJ BTL BG VT)))) (NOBS RIGHT (OBJ BTL)) (OLD OBS (SEQ (OBJ BTL BG VT) (OBJ BTL BG VT) (OBJ BTL BG VT))) (OLD OBS (SEQ (OBJ BTL RD HZ) (OBJ BTL RD HZ) (OBJ BTL RD HZ))) (OLD OBS.AT RIGHT (OBJ BTL BG VT)) (OLD OBS.AT RIGHT (OBJ BTL BG VT)) (* OBS.AT RIGHT (OBJ BTL BG VT)) (OLD LOOK.FOR RIGHT (OBJ BTL)))

FIGURE 33. RUN OF SCP2 ON TASK P1

DISPLAY: (EDGE (BTL GN UP) (BTL BL UP) (BTL GN UP) (BTL BL LF) (BTL GN LF) (BTL BL LF) EDGE)

39. STM: ((GOAL * SOLVE) (OBS (SEQ (SEQ (OBJ BTL BG VT) (OBJ BTL BG VT) (OBJ BTL BG VT)) (SEQ (OBJ BTL BG HZ) (OBJ BTL BG HZ) (OBJ BTL BG HZ)))) (NOBS RIGHT (OBJ BTL)) (OLD OBS (SEQ (OBJ BTL BG HZ) (OBJ BTL BG HZ) (OBJ BTL BG HZ))) (OLD OBS (SEQ (OBJ BTL BG VT) (OBJ BTL BG VT) (OBJ BTL BG VT))) (OLD OBS.AT RIGHT (OBJ BTL BG HZ)) (OLD OBS.AT RIGHT (OBJ BTL BG HZ)) (* OBS.AT RIGHT (OBJ BTL BG HZ)) (OLD LOOK.FOR RIGHT (OBJ BTL)))

FIGURE 34. RUN OF SCP2 ON TASK P2

DISPLAY: (EDGE (BTL YL DN) (BTL BL RT) (BTL YL DN) (BTL BL RT) (BTL YL DN) (BTL BL RT) EDGE)

42. STM: ((GOAL * SOLVE) (OBS (SEQ (SEQ (SEQ (OBJ BTL YL VT) (OBJ BTL BG HZ)) (SEQ (OBJ BTL YL VT) (OBJ BTL BG HZ))) (SEQ (OBJ BTL YL VT) (OBJ BTL BG HZ)))) (NOBS RIGHT (OBJ BTL)) (OLD OBS (SEQ (OBJ BTL YL VT) (OBJ BTL BG HZ))) (OLD OBS (SEQ (SEQ (OBJ BTL YL VT) (OBJ BTL BG HZ)) (SEQ (OBJ BTL YL VT) (OBJ BTL BG HZ)))) (OLD OBS.AT RIGHT (OBJ BTL BG HZ)) (OLD OBS.AT RIGHT (OBJ BTL YL VT)) (OLD LOOK.FOR RIGHT (OBJ BTL)) (OLD END LOOK.FOR))

FIGURE 35. RUN OF SCP2 ON TASK P3

the boundary element is different from the existing sequence of elements (the ones marked by =). For instance, PD5 and PD6 need to respond to the occurrence of a NOBS as a boundary. The difficulty this produces can be seen in Figure 36, which shows the critical moment (26) in the run of Figure 35. The occurrence of a new observed object in STM (OBS (OBJ YL VT)) triggers the grouping of the two sequences, since it acts as a perfectly good boundary for PD5. What we want is for the system to delay to see if another subsequence will build up, so that a group of three can be put together. For that to happen the system must either distinguish different kinds of boundaries or (not exclusively) have a more definite expectation of the organization that is coming (i.e., better than PD8).

An unsatisfactory solution, but one that gets the right result in the short run is shown in Figure 37, where alternative versions of PD5 and PD6 are given that restrict the boundaries acceptable to agree with the grouping that is to be done (e.g., all OBJs or all SEQs). Then something must be added to permit the the final act of organization at the end. This is provided by PD11, which constructs a boundary element of whatever type is necessary. Figure 38 shows the result.

Although we don't show it, SCP3 continues to operate satisfactorily on P1 and P2. Figures 39, 40, 41 and 42 show the terminal behavior on displays P4, P5, P6 and P7 respectively. The result P7 is satisfactory. In fact, P7 represents a case where the subject does not initially create any organization on the sequence, similar to the performance of SCP1 on P2. Thus, in modifying the program to work more appropriately on P2, it was important not to go so far as to prohibit similar behavior on other displays. Behavior

```
DISPLAY: (EDGE (BTL YL DN) (BTL BL RT) (BTL YL DN) (BTL BL RT) >> (BTL YL DN) (BTL BL RT) EDGE)
<NEW.OBS>: NIL
<END.OBS>: NIL
26.  STM: ((OBS.AT RIGHT (OBJ BTL YL VT)) (LOOK.FOR RIGHT (OBJ BTL)) (OBS (SEQ (OBJ BTL YL VT)
(OBJ BTL BG HZ))) (= OBS (SEQ (OBJ BTL YL VT) (OBJ BTL BG HZ))) (OLD OBS.AT RIGHT (OBJ BTL BG
HZ)) (OLD OBS.AT RIGHT (OBJ BTL YL VT)) (OLD OBS.AT RIGHT (OBJ BTL BG HZ)) (OLD OBS.AT RIGHT
(OBJ BTL YL VT)) (OLD OBS LEFT (OBJ SPC)))
PD5  TRUE

29.  STM: ((OBS (SEQ (SEQ (OBJ BTL YL VT) (OBJ BTL BG HZ)) (SEQ (OBJ BTL YL VT) (OBJ BTL BG
HZ)))) (OBS.AT RIGHT (OBJ BTL YL VT)) (OLD OBS (SEQ (OBJ BTL YL VT) (OBJ BTL BG HZ))) (OLD OBS
(SEQ (OBJ BTL YL VT) (OBJ BTL BG HZ))) (LOOK.FOR RIGHT (OBJ BTL)) (OLD OBS.AT RIGHT (OBJ BTL
BG HZ)) (OLD OBS.AT RIGHT (OBJ BTL YL VT)) (OLD OBS.AT RIGHT (OBJ BTL BG HZ)) (OLD OBS.AT
RIGHT (OBJ BTL YL VT)))
```

FIGURE 36. CRITICAL PART OF RUN OF FIGURE 35 WHERE EVOKED PD5

```
; SCP3: MODIFICATION OF SCP2
;       AUGMENTATION TO SCP2
;
;       (THUS THE PART OF SCPF.E05 THAT IS DIFFERENT)
;       ADDS P11 TO PROVIDE BOUNDARY FROM NOBS
;       MODIFIES P5, P6 TO RESTRICT BOUNDARY TO <OBS>
;
DEFINE.SYMBOLS!
;
PD5: ((<OBS>) AND (<OBS> X1 == (<POBJ.TYPE>)) AND
       (= <OBS> X1) AND (= <OBS> X1) ABS -->
       (<OBS> ===> OLD <OBS>)(= ====> OLD)(OBS (SEQ X1 X1)))
;
PD6: ((<OBS>) AND (<OBS> X1 == (<POBJ.TYPE>)) AND
       (= <OBS> X1) AND (= <OBS> X1) AND (= <OBS> X1) ABS -->
       (<OBS> ===> OLD <OBS>)(= ====> OLD)(OBS (SEQ X1 X1 X1)))
;
PD11: ((NOBS) AND (<OBS> X1 == (<POBJ.TYPE>)) AND
        (<OBS> NOBS) ABS --> (<OBS> NOBS))
;
PS2: (PD7 PD4 PD3 PD8 PD6 PD5 G12 PD11 PD10 PD9 PD2 PD1)
;
"SCPF.E05 ADDITION LOADED" RETURN.TO.TTY!
```

FIGURE 37. SCP3: MODIFIED SYSTEM FOR SERIES COMPLETION TASK TO AVOID WRONG GROUPING

```
DISPLAY: (EDGE (BTL YL DN) (BTL BL RT) (BTL YL DN) (BTL BL RT) (BTL YL DN) (BTL BL RT) EDGE)


41.  STM: ((GOAL * SOLVE) (OBS (SEQ (SEQ (OBJ BTL YL VT) (OBJ BTL BG HZ)) (SEQ (OBJ BTL YL VT)
(OBJ BTL BG HZ)) (SEQ (OBJ BTL YL VT) (OBJ BTL BG HZ)))) (OBS NOBS) (OLD OBS (SEQ (OBJ BTL YL
VT) (OBJ BTL BG HZ))) (OLD OBS (SEQ (OBJ BTL YL VT) (OBJ BTL BG HZ))) (= OBS (SEQ (OBJ BTL YL
VT) (OBJ BTL BG HZ))) (NOBS RIGHT (OBJ BTL)) (OLD LOOK.FOR RIGHT (OBJ BTL)) (OLD END LOOK.FOR))
```

FIGURE 38. RUN OF SCP3 ON TASK P3

DISPLAY: (EDGE (BTL BL UP) (BTL BL UP) (BTL YL UP) (BTL YL RT) (BTL BL RT) (BTL BL RT) EDGE)

39. STM: ((GOAL * SOLVE) (OBS (SEQ (SEQ (OBJ BTL BG VT) (OBJ BTL BG VT)) (SEQ (OBJ BTL BG HZ) (OBJ BTL BG HZ)))) (NOBS RIGHT (OBJ BTL)) (OLD OBS (SEQ (OBJ BTL BG HZ) (OBJ BTL BG HZ))) (OLD OBS (SEQ (OBJ BTL BG VT) (OBJ BTL BG VT))) (OBS NOBS) (OBS.AT NOBS) (OLD OBS.AT RIGHT (OBJ BTL BG HZ)) (OLD OBS.AT RIGHT (OBJ BTL BG HZ)) (OLD LOOK.FOR RIGHT (OBJ BTL)) (OLD END LOOK.FOR))

FIGURE 39. RUN OF SCP3 ON TASK P4

DISPLAY: (EDGE (BTL RD LF) (BTL RD RT) (BTL BL RT) (BTL BL LF) (BTL RD RT) (BTL RT LF) EDGE)

42. STM: ((GOAL * SOLVE) (OBS (SEQ (SEQ (OBJ BTL BG HZ) (OBJ BTL BG HZ)) (SEQ (OBJ BTL RD HZ) (OBJ BTL COLOR)))) (NOBS RIGHT (OBJ BTL)) (OLD OBS (SEQ (OBJ BTL RD HZ) (OBJ BTL COLOR))) (OLD OBS (SEQ (OBJ BTL BG HZ) (OBJ BTL BG HZ))) (OBS NOBS) (OLD OBS.AT RIGHT (OBJ BTL COLOR)) (OLD OBS.AT RIGHT (OBJ BTL RD HZ)) (OBS.AT NOBS))

FIGURE 40. RUN OF SCP3 ON TASK P5

DISPLAY: (EDGE (BTL YL RT) (BTL YL RT) (BTL GN DN) (BTL GN DN) (BTL YL RT) (BTL YL RT) EDGE)

46. STM: ((GOAL * SOLVE) (OBS (SEQ (SEQ (OBJ BTL YL HZ) (OBJ BTL YL HZ)) (SEQ (SEQ (OBJ BTL YL HZ) (OBJ BTL YL HZ)) (SEQ (OBJ BTL BG VT) (OBJ BTL BG VT))))) (NOBS RIGHT (OBJ BTL)) (OLD OBS (SEQ (SEQ (OBJ BTL YL HZ) (OBJ BTL YL HZ)) (SEQ (OBJ BTL BG VT) (OBJ BTL BG VT)))) (OLD OBS (SEQ (OBJ BTL YL HZ) (OBJ BTL YL HZ))) (OBS NOBS) (OLD OBS (SEQ (OBJ BTL BG VT) (OBJ BTL BG VT))) (OLD OBS (SEQ (OBJ BTL YL HZ) (OBJ BTL YL HZ))) (OBS.AT NOBS) (OLD OBS.AT RIGHT (OBJ BTL YL HZ)) (OLD OBS.AT RIGHT (OBJ BTL YL HZ)))

FIGURE 41. RUN OF SCP3 ON TASK P6

DISPLAY: (EDGE (BTL BL UP) (BTL GN UP) (BTL BL UP) (BTL BL DN) (BTL GN DN) (BTL BL DN) EDGE)

32. STM: ((GOAL * SOLVE) (OBS.AT NOBS) (NOBS RIGHT (OBJ BTL)) (OBS.AT RIGHT (OBJ BTL BG VT)) (OLD LOOK.FOR RIGHT (OBJ BTL)) (OLD END LOOK.FOR) (= OBS.AT RIGHT (OBJ BTL BG VT)) (= OBS.AT RIGHT (OBJ BTL BG VT)) (= OBS.AT RIGHT (OBJ BTL BG VT)))

FIGURE 42. RUN OF SCP3 ON TASK P7

on P6 is partially satisfactory. The system does not have the concept of surrounding, so it cannot obtain the same concept as the subject. It does however, pick up some of the underlying regularity. Behavior on P4 is also partially satisfactory. The production system has no mechanism for breaking off the scan and the behavior of the subject indicates a much stronger expectation for organization than our system provides. However, SCP3 does pick up the first pair and then fails to pick up the pair (say on just color) in the middle. Since it continues (whereas the subject breaks off) it also picks up the second blue pair; and then it puts the two sequences together at the end.* The subject's response on task P5 is not within the range of our program, since it does not have the additional direction concepts to permit it to see the first two as a unit in terms of direction as well as color.

* The careful reader will note that additional cells have been added to STM for the P6 and P4 runs. The exact size of this STM cannot yet be determined, since it holds much control information not accounted for in the usual models. Hence we have set it at whatever size seemed appropriate.

VII. CONCLUSION

Let us summarize very briefly where this exploration has taken us. We started with the desire to obtain an explicit control structure for a system that was able to perform tasks involving stimulus encoding. Rather than start fresh we chose to adapt a system that had been developed for describing behavior in problem solving situations, which already came equipped with an explicit control structure.

At the level that has been called sufficiency analysis, the enterprise has been moderately successful. The system developed (PSG + LKE + SC3 + SCP3) does not violate seriously the general characteristics of human cognitive and perceptual organization as we currently understand them. It does encode stimuli and in not unreasonable ways. It does have an explicit control structure and control interface between the perceptual structure and the more central cognitive structure. Furthermore, the control structure plays a significant role in producing behavior. For example, in the Neal Johnson task, it forced us to recode while responding; and in the series completion task it forced us to give up generality on the grouping productions (PD5 and PD6) and to make the system explicitly recognize the end of the sequence.

All the above lends support to the enterprise. On the other hand it is apparent that we hardly understand at all the nature of the system created. Within the confines of this paper we have not even exhibited the behavior of the system along many important dimensions. For example, we have not shown its capability to perceive sequences directly. We might have exhibited it by trying a different processing strategy in place of PD8. It could take the formed sequence as a new instruction for how to look at the display. For instance, we might have labeled sequences as NEW when first created and then used a production such as:

```
(NEW OBS XI « (SEQ)) AND (LOOK.FOR X2 == «POBJ.TYPE>) ) -->
(NEW ==» (X2 ==> XI))
```

We did not follow this path, mostly because — like the path we did follow it simply raises a large number of issues and adjustments in the system before it produces appropriate behavior*

The example above is only one form of unexamined behavior. Others Include the ability to adjust the level of detail upward again, after it has been once seen; the ability to match perceived objects **SQ** as to create knowledge of their differences; the ability to use a complex perceived object to guide re-perception of the display (as occurs during the remainder of each of the protocols from which our Initial utterances were taken); and even the final form of a production system that would do the full gamut of perceptual organization showed by the subject (Figure 21)•

In all of the above it is not obvious to me (and, 1 presume, to the reader as well) just what are the capabilities and characteristics of the system. The system does have the power to produce some sorts of performance in all these areas, without further basic modification or augmentation. But experience with even the existing small fragment of its behavior shows it is not easy to arrange to produce a given performance. Although the system has many aspects of a general programming system, it also has definite characteristics of its own that do not permit one simply to state to it in clear terms (so to speak) what is desired. Indeed, it is the very control structure that frustrates this, compared to the sorts of control structures in user-oriented programming languages, which permit absolute local control and protection from unwanted side effects.

To offset the pessimism of the above remarks, one can conclude something about the psychological character of these production systems, even from the small amount of experience that is available. For instance, the natural way to write productions that encode sequences is recursively: from X (SEQ XX) to construct (SEQ X X X). In fact, an earlier production system was constructed this way. This appears to violate the sort of rule that Neal Johnson was attempting to establish, in which one could not peek inside the coded expression. More important, such a production is indeed recursive and there is no way to keep it from constructing coded groups that are as large as you please, e.g.,

X (SEQ X X X X X X X) --> (SEQ X X X X X X X X)

This clearly violates the extensive experience on the use of small encodings that is apparent throughout the data on human encoding. Thus, the present production system admits only finite encodings of two or three. While slightly less elegant, it appear to match more closely what we know of human behavior.

However, despite the above, it would appear that statements about the inadequacies of the system in the light of current psychological knowledge are somewhat premature. My own feelings, upon creating the LKE version, was that the model was psychologically false in a number of obvious ways and that its main excuse for living was that it would at least turn over. I still believe that judgment, but I am no longer prepared to modify the basic structure until more evidence becomes available about the inadequacies of its behavior and whether they are due to not understanding processing strategies, or whether they represent inherent structural features of the system.

Consequently, this paper must end on a note of incompleteness, though one that is hopefully appropriate to a theoretical exploration.

REFERENCES

1. R. C. Atkinson and R. M. Shiffrin, "Human memory: A proposed system and its control processes," in K. W. Spence and J. T. Spence (eds.) The Psychology of Learning and Motivation, vol. 2, Academic Press, 1968.

2. G. H. Bower, "Mental imagery and associative learning," in L. Gregg (ed.) Cognition in Learning and Memory, Wiley (in press).

3. D. E. Broadbent, "Psychological aspects of short-term and long-term memory," Proc. Royal Society of London B, 175, 333-350, 1970.

4. W. G. Chase and H. A. Simon, "Perception in chess," Cognitive Psychology, forthcoming.

5. E. A. Feigenbaum, "The simulation of verbal learning behavior," Proc. Western Joint Computer Conference, 19, 121-132, 1961.

6. J. Feldman, "Simulation of behavior in the binary choice experiment," Proc. Western Joint Computer Conference, 19, 133-144, 1961.

7. J.Feldman, F. M. Tonge and H. Kanter, "Empirical explorations of a hypothesis-testing model of binary choice behavior," in A. C. Hoggatt and F. E. Balderston (eds.) Symposium on Simulation Models, South-Western Publishing Company, 1963.

8. J. E. Hopcroft and J. D. Ullman, Formal Languages and their Relation to Automata, Addison-Wesley, 1969.

9. E. B. Hunt, Concept Formation, Wiley, 1962.

10. E. S. Johnson, "An information-processing model of one kind of problem solving," Psychological Monographs, Whole No. 581, 1964.

11. N. F. Johnson, "The role of chunking and organization in the process of recall," in G. Bower (ed.) The Psychology of Learning and Motivation, vol. 4, Academic Press, 1970.

12. D. Klahr and J. G. Wallace, "The development of serial completion strategies: The information processing approach," British Journal of Psychology, 61, 243-257, 1970.

13. E. L. L. Leewenberg, "Quantitative specification of information in sequential patterns," Psychological Review, 76, 216-220, 1969.

14. R. S. McLean and L. Gregg, "Effects of induced chunking on temporal aspects of serial recitation," J. Exp. Psychology, 74, 455-459, 1967.

15. G. A. Miller, "The magical number seven, plus or minus two," Psychological Review, 63, 81-97, 1956.

16. M. Minsky, Computation: Finite and Infinite Machines, Prentice-Hall, 1967.

17. J. L. Myers, "Sequential choice behavior," in G. Bower (ed.) The Psychology of Learning and Motivation, vol. 4, Academic Press, 1970.

18. A. Newell, Studies in Problem Solving: Subject S3 on the cryptarithmetic task DONALD+GERALD=ROBERT, Carnegie-Mellon University, 1967.

19. A. Newell, "On the analysis of human problem solving protocols," in J. C. Gardin and B. Jaulin (eds.) Calcul et Formalisation dans les Sciences de L'Homme, Centre National de la Recherche Scientifique, 1968.

20. A. Newell, D. McCracken, G. Robertson and P. Freeman, The Kernel Approach to Building Software Systems, Computer Science Research Review, Carnegie-Mellon University, 1971.

21. A. Newell and H. A. Simon, Human Problem Solving, Prentice-Hall, 1972.

22. A. Pavio, "Mental imagery in associative learning and memory," Psychological Review, 76, 241-263, 1969.

23. F. Restle, "Theory of serial pattern learning: structural trees," Psychological Review, 77, 481-495, 1970.

24. H. A. Simon, Complexity and the Representation of Patterned Sequences of Symbols, CIP Working Paper 203, Carnegie-Mellon University, 1972.

25. H. A. Simon and E. A. Feigenbaum, "An information-processing theory of some effects of similarity, familiarization and meaningfulness in verbal learning," J. Verbal Learning and Verbal Behavior, 3, 385-396, 1964.

26. H. A. Simon and K. Kotvsky, "Human acquisition of concepts for sequential patterns," Psychological Review, 70, 534-546, 1963.

27. H. A. Simon and A. Newell, "Human problem solving: The state of the theory in 1970," American Psychologist, 26, 145-159, 1971.

28. P. C. Vitz and T. C. Todd, "A coded element model of the perceptual processing of sequential stimuli," Psychological Review, 76, 433-449, 1969.

29. N. Waugh and D. A. Norman, "Primary memory," Psychological Review, 76, 89-104, 1964.

30. W. A. Wickelgren, "Multitrace strength theory," in D. A. Norman (ed.) Models of Human Memory, Academic Press, 1970.

APPENDIX I: INTERPRETATION RULES FOR PRODUCTION SYSTEM PSF

Executing a production system (1 - 7)

1. A list of productions and production systems is considered a single linear list of productions.

2. Each production is considered in order.

3. Each production constitutes an independent context with respect to assignment of values for variables and class names, all communication between successive evocations of productions occurring via STM.

4. The condition of a production is matched to STM, and the actions elements of the production are executed if the match succeeds.

5. If a production is successfully matched then productions are considered again starting with the first production.

6. Starting over occurs independently of the actions of the successful production, including termination of the action sequence by a FAIL. The exception is a STOP.PS action, which terminates the production system.

7. If no production is satisfied, then the production system terminates.

Matching a production condition (8 - 12)

8. Each condition element is considered in order.

9. Each condition element is matched against each STM element in order.

10. A condition element matches a memory element if:

    10.1 Each symbol in the condition element matches some symbol in the memory element.

    10.2 The symbols in the condition element are considered in order.

    10.3 Memory elements are also considered in order.

    10.4 However, memory elements may be skipped, except the first.

    10.5 If a symbol has a proper name, then the match is on the name of the symbol.

    10.6 Otherwise the symbol is taken as designating another element and the match is executed recursively.

    10.7 A variable can be matched by being assigned, as value, the symbol to which it is being matched, provided that the symbol is in the domain of the variable (if it has one).

10.8 A class name can be matched by being assigned, as value, the symbol to which it is being matched, provided that the symbol is a member of the class.

10.9 A variable or class name that has already been assigned a value takes on that value during the remainder of the match.

11. A memory element that has been matched by a condition element is not considered in matching the remainder of the elements.

12. Whether the entire condition matches is determined by considering each condition element in accordance with connectives:

12.1 C1 AND C2 matches if C1 matches and C2 matches.

12.2 C1 OR C2 matches if C1 matches or C2 matches or both.

12.3 C1 ABS matches if C1 is absent, i.e., does not match.

12.4 Any single level sequence of the above connectives is legal, but embedded expressions are not.

E.g., C1 AND C2 AND C3 OR C4 AND C5 ABS is legal, but (C1 AND C2) OR (C3 AND C4) is not legal.

Executing actions after successful matching (13 - 16)

13. All STM elements participating in the match are moved to the front of STM in the order of the condition elements to which they correspond. This happens prior to any of the actions.

E.g., if (C AND B --> A1) matches STM:(A B C D), then STM is reorganized as STM:(C B A D) before action A1 is executed.

14. Each action element is considered in order.

15. Values of variables and class names assigned prior (in the production) to an action element hold during the execution of an action element.

16. The processing that occurs with an action element depends on what action connective it contains:

16.1 ACTION: FAIL Terminates the execution of action elements, thus ending the production.

16.2 ACTION: STOP.PS Terminates production system.

16.3 ACTION: (OPR ...) The action is an operator and will be executed as a program (which might be a production system).

16.4 ACTION: (X1 == X2) X1 is either a variable or a class name; it is assigned (or reassigned) the value X2.

16.5 ACTION: (X1 ##) X1 is either a variable or a class name; its value (if it exists) is unassigned.

16.6 ACTION: (X1 X2 ... ==> Y1 Y2 ...) The first element in STM is modified by replacing the sequence X1 X2 ... by the sequence Y1 Y2 ... . The identification is only on the first symbol (i.e., on X1), the other symbols (i.e., X2 ...) being in effect simply a way to define an interval of N symbols. If X1 does not exist in the STM element, nothing happens.

16.7 ACTION: (X1 X2 ... ===> Y1 Y2 ...) The second element in STM is modified analogously to ==>.

16.8 ACTION: (X1 X2 ... ====> Y1 Y2 ...) The third element in STM is modified analogously to ==>.

16.9 ACTION: (NTC X1) X1 is noticed in STM and moved to the front. The match used to identify X1 is the same as that used in the match of condition elements. If X1 is not found in STM, then nothing happens.

16.10 ACTION: (...) In all cases when a specific action connective (as enumerated above) does not exist the action element is taken to be a form for the creation of a new element to go into STM (at the front). A copy of the element is made and all values of variables are replaced by their assigned values. If there are subelements (indicated by symbols that do not have proper names), they too are copied.

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Computer Science Department Carnegie-Mellon University Pittsburgh, Pa. 15213	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED 2b. GROUP

3. REPORT TITLE

A THEORETICAL EXPLORATION OF MECHANISMS FOR CODING THE STIMULUS

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Scientific Final

5. AUTHOR(S) (First name, middle initial, last name)

Allen Newell

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
April, 1972	109	30

8a. CONTRACT OR GRANT NO. F44620-70-C-0107 b. PROJECT NO. 9769 c. 61102F d. 681304	9a. ORIGINATOR'S REPORT NUMBER(S) 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
TECH OTHER	Air Force Office of Scientific Research 1400 Wilson Blvd. Arlington, Va. 22209

13. ABSTRACT

This paper explores an information processing model of how stimuli are perceived and encoded. The model is an extension of recent work on human problem solving, which has yielded an explicit programming structure (a production system) as a representation of time course of human behavior in some relatively simple discrete symbolic tasks. The emphasis in the present paper is on obtaining an explicit representation of the control structure in the immediate processor and on the communication between the immediate processor and the perceptual system. The internal structure of the perceptual system is not explored in detail. The paper presents the original production system for problem solving and illustrates its structure and behavior. It then discusses the nature of stimulus encoding and what is provided by the model as it stands. This leads to the introduction of a task to guide the extension of the model. A model of the perceptual system is then presented and its behavior in conjunction with the main system illustrated.

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6801

Security Classification

Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
human cognition						
cognitive simulation						
production system						
perception						
information processing psychology						
cryptarithmetic						
artificial intelligence						