# PRODUCTIONS SYSTEMS: MODELS OF CONTROL STRUCTURES

Allen Newell

May, 1973

This paper is to appear in W. G. Chase (ed.) Visual Information Processing, New York: Academic Press. (in press) This research was supported in part by Research Grant MH-07732 from the National Institutes of Health and in part by the Advanced Research Projects Agency of the Office of the Secretary of Defense (F44620-70-C-0107) which is monitored by the Air Force Office of Scientific Research.

Carnegie-Mellon University
Pittsburgh, Pennsylvania

## ABSTRACT

An exposition of the potentiality of production systems as a model of the detailed control structure of humans. Contains a detailed treatment of the elementary Sternberg reaction time experiments in binary classification as a means of exhibiting the uses of production systems. Leads to a hypothesis for these experiments different from the usual one of exhaustive search, called the Decoding Hypothesis.

# PRODUCTION SYSTEMS: MODELS OF CONTROL STRUCTURES

*Allen Newell*
Carnegie-Mellon University

A production system is a scheme for specifying an information processing system. It consists of a set of productions, each production consisting of a condition and an action. It has also a collection of data structures: expressions that encode the information upon which the production system works--on which the actions operate and on which the conditions can be determined to be true or false.

A production system, starting with an initially given set of data structures, operates as follows. That production whose condition is true of the current data (assume there is only one) is executed, that is, the action is taken. The result is to modify the current data structures. This leads in the next instant to another (possibly the same) production being executed, leading to still further modification. So it goes, action after action being taken to carry out an entire program of processing, each evoked by its condition becoming true of the momentarily current collection of data structures. The entire process halts either when no condition is true (hence nothing is evoked) or when an action containing a stop operation occurs.

Much remains to be specified in the above scheme to yield a definite information processing system. What happens (a likely occurrence) if more than one production is satisfied at once? What is the actual scheme for encoding information? What sort of collection of data structures constitutes the current state of knowledge on which the system works? What sort of tests are expressible in the conditions of productions? What sort of primitive operations are performable on the data and what collections of these are expressible in the actions of productions? What sorts of additional memories are available and how are they accessed and written into? How is the production system itself modified from within, or is this possible? How much time (or effort) is taken by the various components of the system and how do they combine to yield a total time for an entire processing?

There are many questions which can be answered in many different ways. Each assemblage of answers yields a different production system with different properties from its siblings. Taken in all, they constitute a family of schemes for specifying information processing systems. Within this family can be found almost any process specification scheme one could like--though not in fact all possible schemes. There are other ways of specifying the information processing to be done. There are languages, such as Algol and Fortran, that take as their basis a specified sequence of operating-processes to be performed, punctuated by test-processes that explicitly direct processing to switch to another sequence. There are languages, such as SNOBOL, that use productions (conditions associating to actions), but each production explicitly switches the processing this way or that to other sequences of production.

Look at the situation a different way. Suppose you know about an information processing system: its memories, its encodings and its primitive operations (both tests and manipulations). What more would you require to obtain a complete picture? You need to know how the system organizes these primitives into an effective processing of its knowledge. This additional organization is called the *control structure*. Production systems are a type of control structure.

The purpose of this paper is to illustrate the possibility of having a theory of the control structure of human information processing. Gains seem possible in many forms: completeness of the microtheories of how various miniscule experimental tasks are performed; the ability to pose meaningfully the problem of what method a subject is using; the ability to suggest new mechanisms for accomplishing a task; the facilitation of comparing behavior on diverse tasks.

We illustrate by actually proposing a theory of the control structure. We are in earnest about the theory; in this respect we are being more than illustrative. However, to be taken seriously, a theory of control should encompass a substantially greater scope of experiments than we are able to deal with here. This also appears to be the first explicit model of the control structure at this level of detail. It would hardly seem that details of the structure are right--even if (as I currently believe) a production system of some sort appears to be a suitable model of the human control.

Our plan is to present a particular production system, noting its psychological properties, but with no attempt to defend it against variant schemes. Using this system we will conduct an analysis of the basic Sternberg paradigm, which underlies several of the experiments discussed in the present symposium. With this basic analysis in hand, we will then discuss in varying levels of detail the potentialites of production systems as models for human control and the issues raised thereby.

## PSG: A Particular Production System

The particular production system presented here, PSG (for production system version G), was developed as a continuation of work with problem solving in crypt-arithmetic (Newell & Simon, 1972, Chapters 5-7). The original data that PSG was designed to deal with were about an order of magnitude grosser than the reaction time data that currently seem most appropriate to defining the behavior of the immediate processor--i.e., it worked with freely produced phrases of a few seconds duration. A recent paper (Newell, 1972) describes PSG and begins the task of applying it to the more detailed situation, focussing on the problem of stimulus encoding.

The overall architecture of the system is shown in Figure 1. All of the action in the system takes

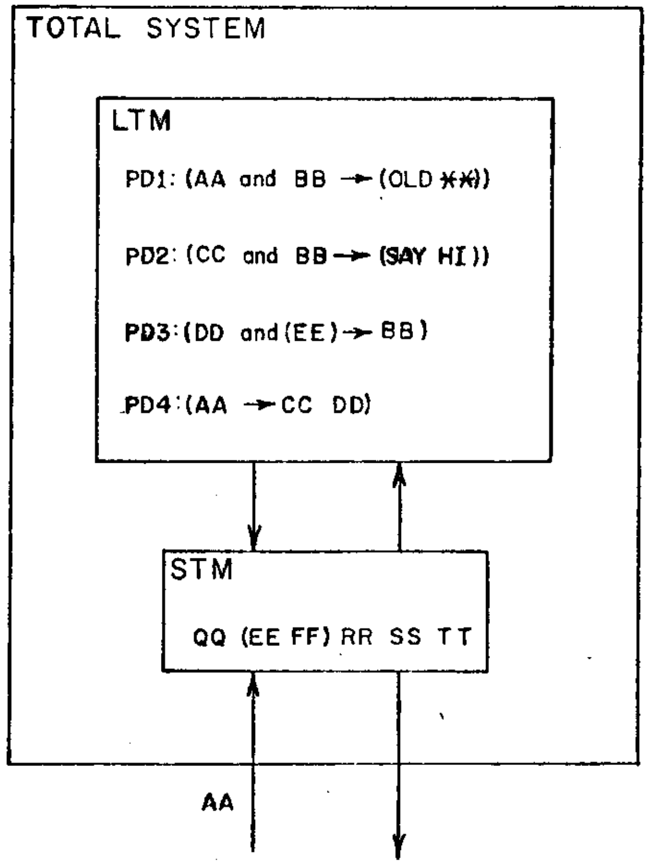


Fig. 1. Overall architecture of PSG.

place in the Short Term Memory (STM), which contains a set of symbolic expressions. STM is to be identified with the memory of Miller (1956) and Waugh and Norman (1965),[1] its size is some small number of chunks (proverbially 7 ± 2).

[1]We prefer not to use the terms primary and secondary memory introduced by Waugh and Norman, since the terms conflict directly with their use in computer science. There, primary memory is the memory that a processor can access for its program, secondary memory being more remote (e.g., a disk or magnetic tape, see Bell & Newell, 1971). What Waugh and Norman call primary memory would be called a scratchpad memory or a working memory. STM seems suitable as a name.

There is no direct representation in PSG of the various buffer memories that appear to be part of the immediate processor of the human: the visual icon of Sperling (1960), (possibly) the precategorical auditory store of Crowder and Morton (1969), and others. The interface to the senses is not represented as well, nor is the decoding on the motor side. Such deficiencies in the architectural model undoubtedly limit the scope and adequacy of the system, but will not be of first importance in this paper.

The STM holds an ordered set of symbolic expressions (i.e., chunks). The ordering shows up, as will be seen later, in that new expressions always enter STM at the front and that the conditions examine the expressions in order starting at the front (hence the frontal expressions may preempt later ones). As can be seen in Figure 1, a symbolic expression may be simply a symbol (e.g., CC) or it may consist of an ordered collection of symbolic expressions (e.g., (EE (AA DD)) ). Thus, symbolic expressions may be built up in a nested fashion, and we can represent them in the manner of algebraic expressions. STM may be taken as holding symbol tokens (i.e., pointers) to the expressions, or it may be taken as holding the expressions themselves. Operationally, there is no way of telling the difference. The degree to which an element in STM is opaque (Johnson, 1970) is determined by the conditions of the productions, which in essence are a description of what aspects of an expression can be responded to.

The Long Term Memory (LTM) consists entirely of an ordered set of productions. Each production is written with the condition on the left separated from the action on the right by an arrow. In Figure 1 only four productions are shown, PD1, PD2, PD3 and PD4. Some of the conditions (e.g., that of PD4) consist of only a single symbolic expression (e.g., PD4 has AA); others have a conjunction of two (e.g., PD1 has AA and BB). Some actions consist of a single symbolic expression (e.g., PD3 with BB), some have a sequence of expressions (e.g., PD4 with CC followed by DD), some have expressions that indicate operations to be performed (e.g., the SAY in PD2).

We will not, for the purposes of this paper, be considering either the question of other types of LTM or of storing new information (new productions) in LTM. This imposes a substantial restriction on the classes of experiments we can consider, but this class still includes many of those in the present symposium. Our assumption about LTM implies a form of homogeniety, but not one that precludes having essentially distinct memories for (say) distinct modalities--the distinctiveness arises from the content of the conditions, not from the structure of the memory itself. The creation of new expressions in STM is not to be taken as creating them in LTM as well. Thus chunking is separated from storing the chunks in LTM so they can be retrieved later.

As the system stands initially, none of the productions is satisfied by the contents of STM and nothing happens. However, we have shown an AA about to enter into STM from the external world. When it does so we get the situation of Figure 2. Here we have shifted to the representation of the system we will use from now on. All the essential elements in Figure 1 are represented, only the various enclosing boxes and input/output arrows are missing. STM now holds the AA and has lost the TT from the far right. (The STMI in the figure is the initial contents of STM.)

In Figure 3 we show the trace of the run, as it is produced by the system.[7] At each cycle the production that is true (i.e., the first whose condition is true) is noted, followed by each action when it is taken. Then the new state of STM is printed and the cycle repeats. The numbers to the left are a count of the number of actions that have occurred so far in the run.

```
PS.ONE: (PD1 PD2 PD3 PD4)
;
PD1: (AA AND BB --> (OLD **))
PD2: (CC AND BB --> (SAY HI))
PD3: (DD AND (EE) --> BB)
PD4: (AA --> CC DD)
;
STMI: (AA QQ (EE FF) RR SS)
;
```

Fig. 2. Example production system PS.ONE

```
VISUAL INFORMATION PROCESSING

00100  0.  STM: (AA QQ (EE FF) RR SS)
00200  PD4 TRUE
00300  0.  ACTION- CC
00400  1.  ACTION- DD
00500  2.  STM: (DD CC AA QQ (EE FF))
00600  PD3 TRUE
00700  2.  ACTION- BB
00800  3.  STM: (BB DD (EE FF) CC AA)
00900  PD1 TRUE
01000  3.  ACTION- (OLD **)
01100  4.  STM: ((OLD AA) BB DD (EE FF) CC)
01200  PD2 TRUE
01300  4.  ACTION- (SAY HI)
01400
01500  **********  HI
01600
01700  5.  STM: (CC BB (OLD AA) DD (EE FF))
01800  PD2 TRUE
01900  5.  ACTION- (SAY HI)
02000
02100  **********  HI
02200
02300  6.  STM: (CC BB (OLD AA) DD (EE FF))
02400  PD2 TRUE
02500
```

Fig. 3. Run of PS.ONE

Let us work through the trace, explaining how the conditions and actions operate. The only condition of the four productions satisfied is that of PD4, the AA on the left side of PD4 matching the AA in STM. This leads to the action of PD4 being evoked, first the CC then the DD. Notice that AA is still in STM but RR and SS have disappeared off the end. This can be seen in Figure 3 at Line 500 where the contents of STM are printed after all actions for production PD4 have been taken.

A production (PD4) having been successfully evoked, the system starts the cycle over. PD4 is of course still satisfied since AA is still in STM. But PD3 is also satisfied since the DD matches the DD in STM and the (EE) also matches the (EE (EE FF)) in STM. This

[2]PSG is a programming system coded in a system building language called L*(G) (see Newell, McCracken, Robertson and Freeman (1971) for an overview of L*(F), the immediate predecessor of L*(G)). PSG operates on a PDP10 and the runs in this paper were made on the PDP10 system of the CMU Computer Science Department.

latter follows from one of several matching rules in PSG. This one says that a match occurs if the condition matches completely, starting with the first symbol in the STM expression but optionally skipping some. Thus (EE) would also match (EE (FF GG)), but would not match an expression without EE at the front, e.g., (FF EE).

When two productions are simultaneously satisfied, the rule for resolving such conflicts is to take the first one in order--here PD3. The result of PD3's action is to put BB into STM as shown at Step 2.

Notice that when PD3 was evoked the two items in its condition moved up to the front of STM in the same order as in the condition. Thus, attended items stay current in STM, while the others drift down toward the end, ultimately to be lost. This mechanism provides a form of automatic rehearsal, though it does not preclude deliberate rehearsal. It also implies that the order of the items in STM does not remain fixed, but flops around with the details of processing.

At the next cycle PD1 is evoked, being the first of the productions satisfied, which includes PD2, PD3 and PD4. The action of PD1 introduces a basic encoding (i.e., construction) operation. (OLD**) is a new expression, which will go into STM like any other. But ** is a variable whose value is the front element in STM.[3] In the case in point the front element is AA, which was moved up by the automatic rehearsal when the condition of PD1 was satisfied. Hence the new element is (OLD AA). This element *replaces* the front element, rather than simply pushing onto the front. The net effect is to take the front element and embed it in a larger expression. Any expression may be written with **. For example, if the action of PD1 had been (XX ** (YY **)), then the new element replacing AA in STM

[3]The constructive operation using ** is an addition to PSG beyond Newell (1972). There we used a replacement operation to modify STM elements; here no modification is possible.

would have been (XX AA(YY AA)), creating a rather complex encoding. It is important that the AA no longer exist in STM (i.e., as the second element, after pushing in the code), since it is necessary to modify STM so AA cannot re-evoke a production.

The import of PD1's action is that it deactivates the STM item able to evoke PD4 (and itself, as well). On the next cycle only PD2 is satisfied. Its action involves SAY, which is a primitive operation of the system that prints out the expression following it in the element, i.e., it prints HI (as shown in the figure).

We see from Figure 3 that the system continues to evoke PD2 and say HI. Nothing happens to modify STM so the condition of PD2 remains satisfied. If we had written:

```
PD2:  (CC AND BB --> (SAY HI) (OLD **))
```

then the production system would have turned off by marking CC as old.

We have indicated by illustration a number of details of PSG, enough to permit us to turn to the analysis of a substantive example. The details given so far are not sufficient. There is a somewhat wider array of primitive operations and many more details of the matching operation for conditions (Newell, 1972). We will introduce the additional aspects of this specification as required throughout the paper.

We can see, even at this stage, that many assumptions are required to specify a complete control structure. Some of them, such as the STM itself, its encoding, and the automatic rehearsal, constitute rather clear psychological postulates. Others, such as the details of matching have psychological implications (presumably every aspect of the system does), but it is hard to know how to state them directly as independent postulates.

## The Sternberg Paradigm

Let us consider the simplest of all binary classification tasks studied by Sternberg (1970). The subject memorizes a small set of symbols, say digits. This is called the *positive set.* In a trial of the experiment proper the subject is given a ready signal, followed by a digit after a short fixed delay. The subject responds "yes" if this so-called probe digit is a member of the positive set, "no" if it is not. The "yes" and "no" responses are usually encoded into button pressings. Many trials are given, so that the task becomes well practiced, the goal being to respond as quickly as possible while keeping a very low error rate. The positive set is varied in blocks, both as to size and composition. The measure taken is the response time (RT) from presentation of the signal to response, measured in milliseconds (ms).

The results of this experiment are well known and form a basis for a number of the experiments which are discussed by Posner (Chapter 2) and Hayes (Chapter 4) in the present symposium. Let us just summarize the basic findings:

(1) Response time is linear with the size of the positive set, the slope being in the range of 35-40 ms. The natural interpretation is that a search is made through the positive set.

(2) The intercept is of the order of 350 ms, but its absolute magnitude is never analysed in detail since it contains several unknown components (e.g., motor response time).

(3) The size of the positive set can be be up to the size normally associated with STM, i.e., around seven elements.

(4) The slope for negative responses (when the probe digit is not in the set) is the same as for positive responses (when the probe is in the set). This violates the results expected if the search is terminated whenever it found the probe in the set (which would make the positive set appear to be on the average only half as large in the case of positive responses). This gives rise to an interpretation of so-called exhaustive search (as opposed to so-called self-terminating search).

(5) There is essentially no serial position effect (the time it takes to respond to a positive probe as a function of where in the positive set the probe digit occurs). This agrees with the exhaustive search notion.

(6) The negative response can differ from the positive response by a constant amount (independent of set size, so the two linear curves lie parallel). The amount is usually about 50 ms, depending on experimental conditions.

Much more is known about this simple task, a full list including all the qualifications to the above would probably run to a hundred statements, rather than the six above. The basic results are highly reproducible and robust. The total set of results, however, is by no means easily seen to be consistent with any simple model.

We can use this paradigm to illustrate concretely what a model of the control system involves and how it makes contact with experimental data. Since we want to reveal the strengths and issues with respect to production systems we will not simply present a final system, but will proceed by a process of step-wise refinement.

We will work our way through a series of production systems until we arrive at one that seems appropriate to the task and the data.

## *PS.ST1: Immediate Recognition*

The obvious scheme, shown in Figure 4, is for STM to contain the positive set, whence the probe is introduced, leading to the attempt at identification. We cannot just have digits as the elements in STM, since we need to distinguish the probe digit from the positive set digits. Thus, we encode the digits of the set as (ELM <DIGIT>), where <DIGIT> means that any digit can go in that place, e.g., (ELM 5); likewise we encode the probe as (PROBE <DIGIT>). The class <DIGIT> is defined explicitly at the top of the figure. STM is initialized with a set of three elements and a ready signal (Line 1500). This latter simply controls the response to attend to the stimulus.

The labeling of items is responsive to a general issue. STM may contain various odd expressions from a diversity of sources. The subject must (normally) be able to distinguish the relevant items from the irrelevant. For instance, the positive set might consist of 2, 3, 4 and the subject (say) become aware of the digit 5 upon a final rehearsal, so that 5 is in STM upon presentation of the probe. We would not expect the subject simply to take 5 as a member of the positive

```
00100   <DIGIT>: (CLASS 0 1 2 3 4 5 6 7 8 9)
00200   ANY: (VAR)
00300   ;
00400   RESPOND: (ACTION (NTC (RESPONSE ANY)) (SAY ANY) (OLD **))
00500   ATTEND: (OPR CALL.TO.USER)
00600   ;
00700   PS.ST1: (PD1 PD2 PD3 PD4)
00800   ;
00900   PD1: ((PROBE) AND (OLD (RESPONSE)) --> (OLD **))
01000   PD2: ((PROBE <DIGIT>) AND (ELM <DIGIT>) --> (RESPONSE YES)
01100        RESPOND)
01200   PD3: ((PROBE) AND (ELM) --> (RESPONSE NO) RESPOND)
01300   PD4: (READY --> ATTEND)
01400   ;
01500   STMI: (READY (ELM 1) (ELM 4) (ELM 9) NIL NIL)
01600   ;
```

Fig. 4. PS.ST1: Immediate recognition.

set, though whether some additional processing would be called for is not clear. In any event, the general use of codes that declare the nature of the item seems to be appropriate and we will do it throughout, without making special arguments each time.

The production system, PS.ST1, consists of four productions. The performance of the task is accomplished by PD2 and PD3. PD2 is satisfied if there is an ELM and a PROBE both of which have the same digit. Thus the occurrence of the class name <DIGIT> in an expression operates as a variable to match against the actual items in STM. The action of PD2 is to put into STM a response expression (in this case to respond YES) and then to fire an operator, RESPOND. This operator, shown at the top of the figure at Line 400, consists of a sequence of actions, i.e., essentially the right side of a production.[4] There are three actions in RESPOND. The first action is to notice anywhere in STM an element of the form (RESPONSE ANY), where ANY is a variable that can take any symbolic expression as value (it is declared at the top of the figure at Line 200). NTC is a primitive operation, that performs a recognition of the same sort as is performed in the matching on the condition side. The second action is to say the value of ANY, which is accomplished by the SAY operation used in PS.ONE. Finally, RESPOND marks the RESPONSE element old, so that the system now knows (in some sense) that it has said the response.

Production PD3 is sensitive to the occurrence of any ELM and any PROBE, and will respond with NO.

---

[4]It thus behaves like a subroutine from a control point of view. However, it works with the same STM as do all other actions. That is, there is no isolation of its data, as there is for instance with a subroutine for computing the sine, SIN(X), which operates in an isolated environment where it knows only about the value of the passed operand, X. Whether or not subroutine control occurs and whether or not subroutine data isolation occurs are psychological questions about the human control system.

However, it sits behind PD2 and thus will only be evoked if PD2 is not, i.e., only if the probe is not a member of the positive set. Thus, PS.ST1 composes its response to the task out of a recognition of membership and a recognition that it is appropriate to respond.

The other two productions in PS.ST1 provide some of the additional control to make the system behave. PD4 responds to READY as does the operator ATTEND. Since we have no model of the external environment, we finesse the matter by having ATTEND call to the console of the user to obtain the input[5] (which will be described in Figure 5, coming up). PD1 is an analog to PD1 in PS.ONE, which serves to recognize that the task is done and to encode this by marking the PROBE element. The effect of this is to keep the system from saying YES YES YES ..., as PS.ONE keeps saying HI HI HI ...

Figure 5 shows a run of PS.ST1, from which it can be seen that the system performs correctly in both the positive and negative cases. The system was reinitialized for the second trial (Line 2600).[6] When ATTEND fires it prints a message to the user. The user puts in the expression after the prompt and then executes a ↑Z to return control to PSG.[7]

Variables occur in two places in PS.ST1: ANY in RESPOND and <DIGIT> in the condition of PD2. In both cases they are assigned a value during the course of a match, in order to satisfy the match. But they perform distinct functions.

---

[5]Thus PSG operates in an essentially interactive mode. The main gain, besides the usual one of flexibility, is that there is no need to program an outer environment.

[6]We might have simply put in another probe at the end of the first session, without reinitializing. However, it would not have behaved properly (why?).

[7]This ↑Z is necessary, since the system allows the user to do whatever he pleases after ATTEND sends its message, hence cannot know until told when the user is finished and wishes to return control to it.

```
VISUAL INFORMATION PROCESSING

PS.ST1 START!
0. STM: (READY (ELM 1) (ELM 4) (ELM 9) NIL NIL)
PD4 TRUE
0. ACTION- ATTEND
     ATTENDING - INPUT NEXT STIMULUS = (PROBE 4)
~
1. ACTION- (PROBE 4)
2. STM: ((PROBE 4) READY (ELM 1) (ELM 4) (ELM 9) NIL)
PD2 TRUE
2. ACTION- (RESPONSE YES)
3. ACTION- RESPOND
4. ACTION- (NTC (RESPONSE ANY))
5. ACTION- (SAY ANY)

********** YES

6. ACTION- (OLD **)
7. STM: ((OLD (RESPONSE YES)) (PROBE 4) (ELM 4) READY (ELM 1) (ELM 9))
PD1 TRUE
7. ACTION- (OLD **)
8. STM: ((OLD (PROBE 4)) (OLD (RESPONSE YES)) (ELM 4) READY (ELM 1) (ELM 9))
PD4 TRUE
8. ACTION- ATTEND
     ATTENDING - INPUT NEXT STIMULUS =

PS.ST1 START!
0. STM: (READY (ELM 1) (ELM 4) (ELM 9) NIL NIL)
PD4 TRUE
0. ACTION- ATTEND
     ATTENDING - INPUT NEXT STIMULUS = (PROBE 8)
~
1. ACTION- (PROBE 8)
2. STM: ((PROBE 8) READY (ELM 1) (ELM 4) (ELM 9) NIL)
PD3 TRUE
2. ACTION- (RESPONSE NO)
3. ACTION- RESPOND
4. ACTION- (NTC (RESPONSE ANY))
5. ACTION- (SAY ANY)

********** NO
```

**Fig. 5.** Run of PS.ST1 on positive and negative cases.

The ANY in RESPOND is used to communicate between one action, which sets the value of ANY, and another, which needs to use it. This communication of values from one action to another occurring later, or from a condition to its action, implies the existence of memory. By the nature of things, this memory cannot be STM (which would lead to an infinite regress). On the other hand, this memory occurs only over the scope of a single production. This is a short time, providing we restrict the time taken by an action. For instance, we should not permit an entire production system to be evoked by one action before going on to the next action.[8] Thus, our control system must posit a very short term buffer memory in addition to the STM working memory.

The <DIGIT> in PD2 serves to restrict the match to work on digits (so that e.g., (ELM BOAT) would not be recognized). No such restriction occurs with ANY. More important, it serves to enforce the equality between two occurrences of digits, since the value assigned at the first place will be used at the second and give a match only if the same digit recurs. Thus, the multiple occurrence is performing a major function of the task--the equality test of probe digit and member digit. Whether there can be multiple occurrences of variables in a condition is an independent psychological question. To replace it with the provision that a variable can occur but once on the condition side is tantamount to making only identification possible (including class membership). This would imply that a primitive operation of equality testing would be required, to be used in the action part. The processing implications of one assumption or the other is unclear, since what additional memory and control is required within the match to accomplish multiple occurrences depends on the mechanism used to implement the match (in particular the amount and kind of parallelism).[9]

How do we know this production system is the right sort of mechanism, given the results of experiments? We need to adopt an explicit timing model, so that we can compute the total time taken in performing the task. The central assumption we will make has three parts:

---

[8]Such facility represents good programming language design, in which one wants indefinite capabilities for recursion. However, we are trying to model the human control system, not construct a neat system.

[9]We state all these issues to show that the conventions of the production system, which may appear to be linguistic in nature, contain substantive psychological assumptions.

The time to evoke the next production is independent of:

(1) the number of productions in the system;

(2) the contents of STM;

(3) the condition of the evoked productions.

The assumptions are meant only as a first approximation. However, they do rule out time being proportional to the number of productions (the assumption that comes naturally from the definition of a production system and its implementation on a digital computer).

In favor of Part (1) is the circumstance that in writing a production system (PS.ST1 or any other) we only put down a few of the conditions to which the subject is presumably sensitive and could respond to if the situation (i.e., the contents of STM) warranted: a wasp lighting on the apparatus, the smell of smoke, an irrelevant remark in the background, turning off the lights and so on, any of which would surely evoke a noticing operation and subsequent alteration of the contents of STM. While reaction to such conditions might be somewhat longer, in no way could the subject be imagined to iterate through all such possible conditions taking an increment of time per possibility. Thus, the set of productions we work with bears no relation to the set of productions that we envision constituting the LTM. More generally, the basic control structure is to be viewed as one of a recognition followed by an action followed by a recognition again--the act of evoking the next action (or mini-sequence of actions) being the basic pulse of the system.[10]

Parts (2) and (3) of the assumption are not quite so compelling and alternatives can be imagined.[11] That

---

[10]This recognition-act cycle is to be contrasted with the basic fetch-decode-execute cycle which is the primitive control structure of the digital computer.

all the conditions are tested simultaneously implies that the time to determine the next production depends on all the unsatisfied conditions as well as the one that is chosen. Thus, no strong dependence could exist on the particular items, and in any event all the items in STM must be involved in the processing, not just those that enter into the selected production.

These three assumptions imply that it takes a constant amount of time, call it *T.evoke*, to determine the next production to be executed. Each production, of course, evokes a sequence of actions. The total time to accomplish the sequence may be variable, depending on the exact actions that occur. The simplest assumption is one of *seriality*: that each action takes a fixed amount of time and that the time for the sequence is the sum of the times for each action. Even simpler is the assumption that each action takes the same time, call it *T.action*. Under this assumption the time for a production with N actions can be written:

$$T.production = T.evoke + N * T.action$$

The special case of T.evoke = 0 is worth a moment's attention. The obvious interpretation is that it takes no time to evoke the production (i.e., to recognize what action sequence to perform) and all the time is taken by the performance of actions. An alternative interpretation is that only a single action can be evoked at a time. That is, writing of a sequence of N actions is simply a shorthand for writing N productions, each of which has a condition and a single action. We assert thereby that the conditions are so unique that only the production associated with the next action would fire. Under this assumption the total time of a production-as-written with N actions is:

[11]For instance, considering elements in order from the front of STM and evoking the first satisfied production would make the time dependent on the contents of STM.

T.production = N * (time-to-evoke + time-for-action)

The two times coalesce to form the T.action in the top formula.

The simplicity of these assumptions should not be disturbing. Their complication can be left to the impact of specific data. Even in this simple form they offer guidance in the analysis of a production system. Notice, by the way, that the production system has a built in seriality in the sequence of production evocations, independent of whether we make the serial assumption for performing a sequence of actions for a given production. Roughly speaking, the time to do a task is proportional to the number of productions evoked to do the task.

Given this much of a timing model, it can be seen from Figure 5 that PS.ST1 produces an answer in a time that is independent of the size of the positive set (essentially, T.evoke + 5*T.action). Thus PS.ST1 disagrees fundamentally with the empirical results. Consequently, let us explore other methods for the task (putting to one side for the moment what is implied by not using a scheme of action that seems possible a priori).

## *PS.ST2: Terminating Search*

Figure 6 shows a production system, PS.ST2, that performs the task by explicitly searching through each of the members of the positive set. PD2 in Figure 6 looks very similar to PD2 in Figure 4. However, there is a critical difference. In Figure 4 the digit selected by <DIGIT> is defined by the probe; thus this seeks out an element in STM that has the same digit. In

```
PS.ST2: (PD1 PD2 PD3 PD4 PD5)
;
PD1: ((PROBE) AND (OLD (RESPONSE)) --> (OLD **))
PD2: ((ELM <DIGIT>) AND (PROBE <DIGIT>) --> (RESPONSE YES)
     RESPOND)
PD3: ((ELM) AND (PROBE) --> (OLD **))
PD4: ((PROBE) AND (ELM) ABS --> (RESPONSE NO) RESPOND)
PD5: (READY --> ATTEND)
;
```

Fig. 6. PS.ST2: Linear terminating search.

Figure 6, the digit is selected by the first (ELM...) in STM; only if this has the same digit as the probe, will there be a match. If this doesn't occur, the next production (PD3) then modifies the first element so that it will not be sensed again by PD2. Thus, these two productions work through the positive set and will find a match if it exists. Only if no more elements exist, will PD4 be evoked and say NO. (PD1 and PD5 are identical to PD1 and PD4 respectively of PS.ST1.)

The condition of PD4 involves detecting the absence of an element in STM, indicated by the ABS following the element. Thus PD4 will not be evoked if there is an item in STM of form (ELM...). This happens not to be strictly necessary for PS.ST2 to work, but somehow providing a production that could be triggered to say NO on the occurrence of the probe alone seems risky. Suppose, for instance, the probe arrived simultaneously with the ready signal. PS.ST2 would behave right; a system with only (PROBE) in the condition of PD4 would not, producing NO immediately.

We have now introduced all but one of the ingredients of matching: (1) the matching of items in STM; (2) the conjunction of condition elements, either for presence or absence; (3) the use of variables and classes (which operate as variables with restricted domains); and (4) the rules for matching an element (or subelement) of the condition with an element (or subelement) of the STM, namely subelement by subelement, working from the front, but allowing the tail of the STM element to not be matched (e.g., (EE) matches (EE FF)). The one addition (to occur in the next example) is (5) permitting a variable to have an associated domain locally. An example of this is:

```
(A X1 == (B C) D)        where X1:  (VAR)
```

This says that X1 must match (B C). Thus the entire condition element matches (A (B C) D), but not (A B C D), ((B C) D) or (A (C B) D).

Examination of the logic of PS.ST2 shows that the time is indeed proportional to the size of the set

VISUAL INFORMATION PROCESSING

```
0. STM: (READY (ELM 1) (ELM 4) (ELM 9) NIL NIL)
PD5 TRUE
0. ACTION- ATTEND
     ATTENDING - INPUT NEXT STIMULUS = (PROBE 4)
~
1. ACTION- (PROBE 4)
2. STM: ((PROBE 4) READY (ELM 1) (ELM 4) (ELM 9) NIL)
PD3 TRUE
2. ACTION- (OLD **)
3. STM: ((OLD (ELM 1)) (PROBE 4) READY (ELM 4) (ELM 9) NIL)
PD2 TRUE
3. ACTION- (RESPONSE YES)
4. ACTION- RESPOND
5. ACTION- (NTC (RESPONSE ANY))
6. ACTION- (SAY ANY)

********** YES


```

Fig. 7. Run of PS.ST2 on positive case.

```
0. STM: (READY (ELM 1) (ELM 4) (ELM 9) NIL NIL)
PD5 TRUE
0. ACTION- ATTEND
     ATTENDING - INPUT NEXT STIMULUS = (PROBE 8)
~
1. ACTION- (PROBE 8)
2. STM: ((PROBE 8) READY (ELM 1) (ELM 4) (ELM 9) NIL)
PD3 TRUE
2. ACTION- (OLD **)
3. STM: ((OLD (ELM 1)) (PROBE 8) READY (ELM 4) (ELM 9) NIL)
PD3 TRUE
3. ACTION- (OLD **)
4. STM: ((OLD (ELM 4)) (PROBE 8) (OLD (ELM 1)) READY (ELM 9) NIL)
PD3 TRUE
4. ACTION- (OLD **)
5. STM: ((OLD (ELM 9)) (PROBE 8) (OLD (ELM 4)) (OLD (ELM 1)) READY NIL)
PD4 TRUE
5. ACTION- (RESPONSE NO)
6. ACTION- RESPOND
7. ACTION- (NTC (RESPONSE ANY))
8. ACTION- (SAY ANY)

********** NO


```

Fig. 8. Run of PS.ST2 on negative case.

searched requiring one evocation and one action for each element examined that is not the probe and then one more (PD2 if positive, PD4 if negative) to generate the response. However, as demonstrated in Figures 7 and 8, PS.ST2 does a self-terminating search. It looks at all the elements in the set in the negative case (Figure 8), but only half the elements (on the average) in the positive case (Figure 7), thus making the slope of the positive case appear only half of what it is in the negative case. But, as noted earlier, the evidence is unequivocal that the slopes are the same for the positive and negative cases. Furthermore, there is no serial position effect (as there would be in PS.ST2). Thus, we have not yet found a method for doing the task that has the right characteristics.

### *PS.ST3 and PS.ST4: Encoded Representations*

The system of Figure 9, PS.ST3, introduces the notion that the set is actually held in an encoded representation (i.e., as a chunk). Thus, we have changed the STM to hold some irrelevant items prior to the start of the trial. At the READY signal the encoded positive set is brought into STM (PD5).

The positive set is encoded as a nested set, as can be seen in the action side of PD5 for a set of three elements. A set of five would have the form: (X (X (X (X X)))). This means that a single production, PD2, can perform the decoding by repeated application. The point of introducing the decoding is that the entire set must be decoded before any further processing is done on it. Thus, the time to decode will be independent of whether the result is to be positive or negative. Thus, PS.ST3 satisfies the experimental results that lead to the inference of the exhaustive search. It does so, however, by attributing the time, not to search (which is done in constant time by PD3 and PD4), but to a linear time to decode the expression. Figures 10 and 11 show runs on PS.ST3 in the positive and negative case that illustrate this. It can be seen that the time to do the task is:

$$T.total = 2*T.action + N*(T.evoke + 3*T.action)$$

Examination of PS.ST3 shows that what enforces the compulsive decoding before testing is that PD2, the decoding production, occurs before PD3 and PD4, the comparison and response productions. Why don't we simply reverse the order? Then we should catch the elements as they are being decoded, and reinstitute a termination search. Figure 12 shows the result, using PS.ST3X which is simply a reordered version of PS.ST3.

VISUAL INFORMATION PROCESSING

```
X1: (VAR)
X2: (VAR)
;
PS.ST3: (PD1 PD2 PD3 PD4 PD5 PD6)
;
PD1: ((PROBE) AND (OLD (RESPONSE)) --> (OLD **))
PD2: ((SET X1 X2) AND (PROBE) --> (OLD **) X2 X1)
PD3: ((PROBE <DIGIT>) AND (ELM <DIGIT>) --> (RESPONSE YES)
     RESPOND)
PD4: ((PROBE) AND (ELM) --> (RESPONSE NO) RESPOND)
PD5: (READY AND (SET) ABS -->
     (SET (ELM 1) (SET (ELM 4) (ELM 9))) )
PD6: (ANY --> ATTEND)
;
STM: (JUNK NIL NIL NIL NIL NIL NIL)
;
```

Fig. 9. PS.ST3: Nested representation.

```
0. STM: (JUNK NIL NIL NIL NIL NIL NIL)
PD6 TRUE
0. ACTION- ATTEND
     ATTENDING - INPUT NEXT STIMULUS = READY
~
1. ACTION- READY
2. STM: (READY JUNK NIL NIL NIL NIL NIL)
PD5 TRUE
2. ACTION- (SET (ELM 1) (SET (ELM 4) (ELM 9)))
3. STM: ((SET (ELM 1) (SET (ELM 4) (ELM 9))) READY JUNK NIL NIL NIL NIL)
PD6 TRUE
3. ACTION- ATTEND
     ATTENDING - INPUT NEXT STIMULUS = (PROBE 4)
~
4. ACTION- (PROBE 4)
5. STM: ((PROBE 4) (SET (ELM 1) (SET (ELM 4) (ELM 9))) READY JUNK NIL NIL NIL)
PD2 TRUE
5. ACTION- (OLD **)
6. ACTION- X2
7. ACTION- X1
8. STM: ((ELM 1) (SET (ELM 4) (ELM 9)) (OLD (SET (ELM 1) (SET (ELM 4) (ELM 9))))
       (PROBE 4) READY JUNK NIL)
PD2 TRUE
8. ACTION- (OLD **)
9. ACTION- X2
10. ACTION- X1
11. STM: ((ELM 4) (ELM 9) (OLD (SET (ELM 4) (ELM 9))) (PROBE 4) (ELM 1)
        (OLD (SET (ELM 1) (SET (ELM 4) (ELM 9)))) READY)
PD3 TRUE
11. ACTION- (RESPONSE YES)
12. ACTION- RESPOND
13. ACTION- (NTC (RESPONSE ANY))
14. ACTION- (SAY ANY)

********** YES


```

Fig. 10. Run of PS.ST3 on positive case.

VISUAL INFORMATION PROCESSING

```
0. STM: (JUNK NIL NIL NIL NIL NIL NIL)
PD6 TRUE
0. ACTION- ATTEND
    ATTENDING - INPUT NEXT STIMULUS = READY
~
1. ACTION- READY
2. STM: (READY JUNK NIL NIL NIL NIL NIL)
PD5 TRUE
2. ACTION- (SET (ELM 1) (SET (ELM 4) (ELM 9)))
3. STM: ((SET (ELM 1) (SET (ELM 4) (ELM 9))) READY JUNK NIL NIL NIL NIL)
PD6 TRUE
3. ACTION- ATTEND
    ATTENDING - INPUT NEXT STIMULUS = (PROBE 8)
~
4. ACTION- (PROBE 8)
5. STM: ((PROBE 8) (SET (ELM 1) (SET (ELM 4) (ELM 9))) READY JUNK NIL NIL NIL)
PD2 TRUE
5. ACTION- (OLD **)
6. ACTION- X2
7. ACTION- X1
8. STM: ((ELM 1) (SET (ELM 4) (ELM 9)) (OLD (SET (ELM 1) (SET (ELM 4) (ELM 9))))
       (PROBE 8) READY JUNK NIL)
PD2 TRUE
8. ACTION- (OLD **)
9. ACTION- X2
10. ACTION- X1
11. STM: ((ELM 4) (ELM 9) (OLD (SET (ELM 4) (ELM 9))) (PROBE 8) (ELM 1)
        (OLD (SET (ELM 1) (SET (ELM 4) (ELM 9)))) READY)
PD4 TRUE
11. ACTION- (RESPONSE NO)
12. ACTION- RESPOND
13. ACTION- (NTC (RESPONSE ANY))
14. ACTION- (SAY ANY)

********** NO
```

Fig. 11. Run of PS.ST3 on negative case.

Trouble results, as we see, since PD4 responds to the non-satisfaction of PD3 by declaring NO immediately, thus causing an error.

What ways exist of patching up the system so it avoids the difficulty of Figure 12, while preserving the self-terminating features? PD4 must be inhibited while decoding goes on, whereas PD3 must not be. The simplest solution is to split the two productions, putting PD3 ahead of PD2 and PD4 afterward. This works just fine. Other alternatives involve making PD4 conditional upon the set being completely decoded. This can be done, for instance, by changing PD4 to:

PD4: ((PROBE) AND (SET) ABS --> (RESPONSE NO) RESPOND)

Thus, although introducing the idea of decoding permitted us to produce a version with the correct timing

```
                    VISUAL INFORMATION PROCESSING

PS.ST3X: (PD1 PD2 PD3 PD4 PD5 PD6)
PD1: ((PROBE) AND (OLD (RESPONSE)) --> (OLD **))
PD3: ((PROBE <DIGIT>) AND (ELM <DIGIT>) --> (RESPONSE YES) RESPOND)
PD4: ((PROBE) AND (ELM) --> (RESPONSE NO) RESPOND)
PD2: ((SET X1 X2) AND (PROBE) --> (OLD **) X2 X1)
PD5: (READY AND (SET) ABS --> (SET (ELM 1) (SET (ELM 4) (ELM 9))))
PD6: (ANY --> ATTEND)

PS.ST3X START!
0.  STM: (JUNK NIL NIL NIL NIL NIL NIL)
PD6 TRUE
0.  ACTION- ATTEND
      ATTENDING - INPUT NEXT STIMULUS = READY
~
1.  ACTION- READY
2.  STM: (READY JUNK NIL NIL NIL NIL NIL)
PD5 TRUE
2.  ACTION- (SET (ELM 1) (SET (ELM 4) (ELM 9)))
3.  STM: ((SET (ELM 1) (SET (ELM 4) (ELM 9))) READY JUNK NIL NIL NIL NIL)
PD6 TRUE
3.  ACTION- ATTEND
      ATTENDING - INPUT NEXT STIMULUS = (PROBE 4)
~
4.  ACTION- (PROBE 4)
5.  STM: ((PROBE 4) (SET (ELM 1) (SET (ELM 4) (ELM 9))) READY JUNK NIL NIL NIL)
PD2 TRUE
5.  ACTION- (OLD **)
6.  ACTION- X2
7.  ACTION- X1
8.  STM: ((ELM 1) (SET (ELM 4) (ELM 9)) (OLD (SET (ELM 1) (SET (ELM 4) (ELM 9))))
         (PROBE 4) READY JUNK NIL)
PD4 TRUE
8.  ACTION- (RESPONSE NO)
9.  ACTION- RESPOND
10.  ACTION- (NTC (RESPONSE ANY))
11.  ACTION- (SAY ANY)

***********  NO
```

Fig. 12.  Run of PS.ST3X showing error.

```
X1: (VAR)
X2: (VAR)
X3: (VAR)
X4: (VAR)
;
PS.ST4: (PD1 PD2 PD3 PD4 PD5 PD6 PD7 PD8 PD9)
;
PD1: ((PROBE) AND (OLD (RESPONSE)) --> (OLD **))
PD2: ((SET X1 X2 X3 X4) AND (PROBE) --> (OLD **) X4 X3 X2 X1)
PD3: ((SET X1 X2 X3) AND (PROBE) --> (OLD **) X3 X2 X1)
PD4: ((SET X1 X2) AND (PROBE) --> (OLD **) X2 X1)
PD5: ((SET X1) AND (PROBE) --> (OLD **) X1)
PD6: ((PROBE <DIGIT>) AND (ELM <DIGIT>) --> (RESPONSE YES)
     RESPOND)
PD7: ((PROBE) AND (ELM) --> (RESPONSE NO) RESPOND)
PD8: (READY AND (SET) ABS -->
      (SET (ELM 1) (ELM 4) (ELM 9)) )
PD9: (ANY --> ATTEND)
;
```

Fig. 13.  PS.ST4:  Linear representation.

properties, we have found minor variations of the same scheme that re-instate the terminating condition, and appear to be somewhat more efficient than the exhaustive scheme.

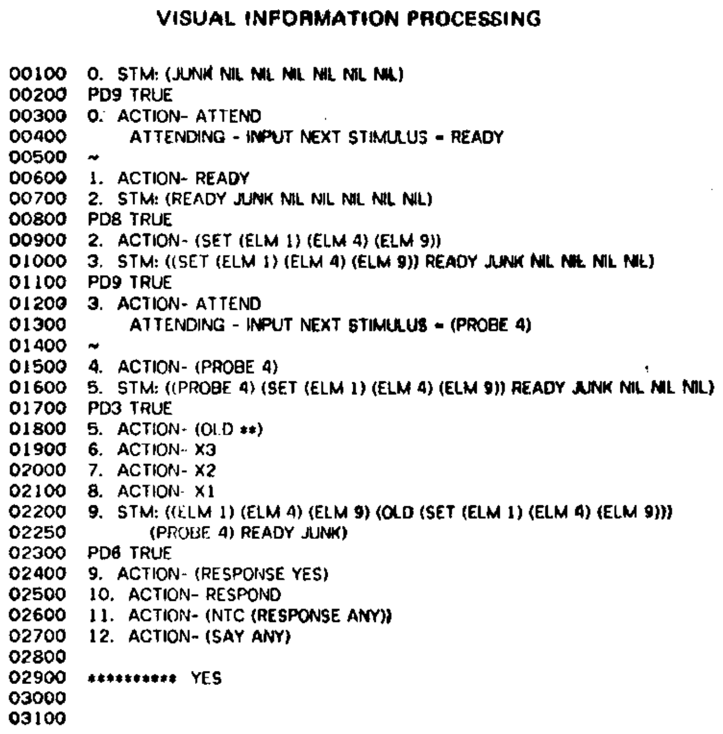
Figure 13 shows an alternative form of encoded representation that appears to overcome these difficulties. A set is now represented by a linear expression, e.g., (SET A B C D). Such sets cannot be decoded recursively, but require a set of productions, one member for each set size. Thus, PD2 to PD5 in PS.ST4 accomplish jointly the decoding of a set in STM into its elements. The recognition of the larger sets occur before smaller ones, since by the matching rules of PSG the productions for smaller sets would also be satisfied by larger sets. The maximum size set admitted in PS.ST4 is four elements; it could be extended to any specific upper limit.[12]

The decoding now occurs within the action sequence of a single production. Thus, it takes minimal time (N*T.action) and there is no opportunity to slip in the evocation of a production (i.e., PD6) that would terminate the search. The rest of PS.ST4 is the same as in PS.ST3. Figure 14 shows a run on a positive case that illustrates how the decoding goes.

Throughout the discussion we have ignored where the positive set came from. In the first examples (PS.ST1 and PS.ST2) we simply posited the elements in STM initially. In the later examples (PS.ST3 and PS.ST4) we posited a set in LTM already assimilated into a production and in the encoded form we wished to work with. We have set to one side the way new productions are created in LTM (i.e., the question of LTM acquisition as it shows up in our system), but the mechanics of encoding are within our purview.

Figure 15 shows PS.ST5, which is an augmentation of PS.ST4 to encode a sequence of incoming elements

[12]The capacity of STM would appear to limit the size of the sets that could be successfully decoded; so also could the capacity of the variable buffer store.



VISUAL INFORMATION PROCESSING

```
0.  STM: (JUNK NIL NIL NIL NIL NIL NIL)
PD9 TRUE
0.  ACTION- ATTEND
      ATTENDING - INPUT NEXT STIMULUS = READY
~
1.  ACTION- READY
2.  STM: (READY JUNK NIL NIL NIL NIL NIL)
PD8 TRUE
2.  ACTION- (SET (ELM 1) (ELM 4) (ELM 9))
3.  STM: ((SET (ELM 1) (ELM 4) (ELM 9)) READY JUNK NIL NIL NIL NIL)
PD9 TRUE
3.  ACTION- ATTEND
      ATTENDING - INPUT NEXT STIMULUS = (PROBE 4)
~
4.  ACTION- (PROBE 4)
5.  STM: ((PROBE 4) (SET (ELM 1) (ELM 4) (ELM 9)) READY JUNK NIL NIL NIL)
PD3 TRUE
5.  ACTION- (OLD **)
6.  ACTION- X3
7.  ACTION- X2
8.  ACTION- X1
9.  STM: ((ELM 1) (ELM 4) (ELM 9) (OLD (SET (ELM 1) (ELM 4) (ELM 9)))
        (PROBE 4) READY JUNK)
PD6 TRUE
9.  ACTION- (RESPONSE YES)
10.  ACTION- RESPOND
11.  ACTION- (NTC (RESPONSE ANY))
12.  ACTION- (SAY ANY)

**********  YES


```

Fig. 14. Run of PS.ST4 on positive case.

into a set with the linear encoding. Figure 16 shows a run where this encoding occurs, stopping at the point where one would go into the rest of the Sternberg task with a READY and a (PROBE). Again, there has to be a separate production for each set size, since each item of the set has to be acquired (with a variable) and then the new set created. A similar program can be written to construct sets in the nested representation. In this case, only a pair of productions is needed (as shown in Figure 17, which gives only the encoding part of the complete system). This pair has the property that it can construct indefinitely large sets, though of course the sets must still be decoded step by step.

We have attended primarily to the equality between the slope of the response time for positive responses and negative responses, when response time is plotted against the size of the positive set. However the negative response can differ from the positive response (Point 6 in our list of empirical properties). This

```
PS.ST5: (PD1 PD2 PD3 PD4 PD5 PD6 PD7 PD8 PD9 PD10 PD11 PD12)
;
PD1: ((PROBE) AND (OLD (RESPONSE)) --> (OLD **))
PD2: ((SET X1 X2 X3 X4) AND (PROBE) --> (OLD **) X4 X3 X2 X1)
PD3: ((SET X1 X2 X3) AND (PROBE) --> (OLD **) X3 X2 X1)
PD4: ((SET X1 X2) AND (PROGE) --> (OLD **) X2 X1)
PD5: ((SET X1) AND (PROBE) --> (OLD **) X1)
PD6: ((PROBE <DIGIT>) AND (ELM <DIGIT>) --> (RESPONSE YES)
     RESPOND)
PD7: ((PROBE) AND (ELM) --> (RESPONSE NO) RESPOND)
PD8: (X1 == (ELM) AND X2 == (ELM) AND READY -->
     (OLD **) (NTC (ELM)) (OLD **) (SET X2 X1))
PD9: (X1 == (ELM) AND (SET X2 X3 X4) AND PEADY -->
     (OLD **) (NTC (SET)) (OLD **) (SET X2 X3 X4 X1))
PD10: (X1 == (ELM) AND (SET X2 X3) AND READY -->
      (OLD **) (NTC (SET)) (OLD **) (SET X2 X3 X1))
PD11: (X1 == (ELM) AND (SET X2) AND READY -->
      (OLD **) (NTC (SET)) (OLD **) (SET X2 X1))
PD12: (ANY --> ATTEND)
;
```

**Fig. 15. PS.ST5: Linear representation, encoding and decoding.**

effect can be attributed to a response bias--that is, the subject sets himself to respond one way, e.g., YES so that the expected response occurs more rapidly than the unexpected one. Such a bias could presumably be adopted in either direction, which is in accord with the empirical findings. (For instance, if there is an appreciable frequency difference between the occurrences of positive and negative instances, then the response is quicker to the more frequent.)

Given a system such as we have been considering, we can ask how, or whether, a response bias can be programmed to permit a more rapid response in one or the other case. Figure 18 shows a solution, PS.ST7, that puts the (RESPONSE YES) element in STM in advance, so it does not have to be done by the positive response production (PD6). We do not show what determines which way the bias goes; from the structure of the production system it could be either way. The actual size of the bias depends on the difference between PD6, which now simply executes RESPOND, and PD7, which has the burden of changing the response to NO. We have shown three different productions, PD7A, PD7B and PD7C. The first does not bother to neutralize (RESPOND YES), but simply puts a (RESPOND NO) ahead of it in STM. Presumably this raises some problems about a freely wandering (RESPONSE YES), but perhaps this could be neutralized

```
VISUAL INFORMATION PROCESSING
0.  STM: (JUNK NIL NIL NIL NIL NIL NIL)
PD12 TRUE
0.  ACTION- ATTEND
     ATTENDING - INPUT NEXT STIMULUS = READY
~
1.  ACTION- READY
2.  STM: (READY JUNK NIL NIL NIL NIL NIL)
PD12 TRUE
2.  ACTION- ATTEND
     ATTENDING - INPUT NEXT STIMULUS = (ELM 1)
~
3.  ACTION- (ELM 1)
4.  STM: ((ELM 1) READY JUNK NIL NIL NIL NIL)
PD12 TRUE
4.  ACTION- ATTEND
     ATTENDING - INPUT NEXT STIMULUS = (ELM 2)
~
5.  ACTION- (ELM 2)
6.  STM: ((ELM 2) (ELM 1) READY JUNK NIL NIL NIL)
PD8 TRUE
6.  ACTION- (OLD **)
7.  ACTION- (NTC (ELM))
8.  ACTION- (OLD **)
9.  ACTION- (SET X2 X1)
10. STM: ((SET (ELM 1) (ELM 2)) (OLD (ELM 1)) (OLD (ELM 2)) READY JUNK NIL NIL)
PD12 TRUE
10. ACTION- ATTEND
     ATTENDING - INPUT NEXT STIMULUS = (ELM 3)
~
11. ACTION- (ELM 3)
12. STM: ((ELM 3) (SET (ELM 1) (ELM 2)) (OLD (ELM 1)) (OLD (ELM 2)) READY JUNK NIL)
PD10 TRUE
12. ACTION- (OLD **)
13. ACTION- (NTC (SET))
14. ACTION- (OLD **)
15. ACTION- (SET X2 X3 X1)
16. STM: ((SET (ELM 1) (ELM 2) (ELM 3)) (OLD (SET (ELM 1) (ELM 2))) (OLD (ELM 3))
       READY (OLD (ELM 1)) (OLD (ELM 2)) JUNK)
PD12 TRUE
16. ACTION- ATTEND
     ATTENDING - INPUT NEXT STIMULUS = (PROBE 1)
```

Fig. 16. Run of PS.ST5 on encoding part only.

```
PD5: (X1 == (ELM) AND X2 == (ELM) AND READY -->
    (OLD **) (NTC (ELM)) (OLD **) (SET X2 X1))
PD6: (X1 == (ELM) AND X2 == (SET) AND READY -->
    (OLD **) (NTC (SET)) (OLD **) (SET X2 X1))
;
```

Fig. 17. Encoding productions for nested representation.

after the response was actually made. PD7B and PD7C both mark the YES respond OLD. PD7B does so by locating the response element in its condition part; PD7C takes an extra NTC action to locate it. Thus, we have a range of time differences depending on which mechanism we opt for.

VISUAL INFORMATION PROCESSING

```
PS.ST7: (PD1 PD2 PD3 PD4 PD5 PD6 PD7X PD8 PD9 PD10 PD11 PD12
        PD13)
;
PD1: ((PROBF) AND (OLD (RESPONSE)) --> (OLD **))
PD2: ((SET X1 X2 X3 X4) AND (PROBE) --> (OLD **) X4 X3 X2 X1)
PD3: ((SET X1 X2 X3) AND (PROBE) --> (OLD **) X3 X2 X1)
PD4: ((SET X1 X2) AND (PROBE) --> (OLD **) X2 X1)
PD5: ((SET X1) AND (PROBE) --> (OLD **) X1)
PD6: ((PROBE <DIGIT>) AND (ELM <DIGIT>) --> RESPOND)
PD7A: ((PROBE) AND (ELM) --> (RESPONSE NO) RESPOND)
PD7B: ((RESPONSE) AND (PROBE) AND (ELM) --> (OLD **)
     (RESPONSE NO) RESPOND)
PD7C: ((PROBE) AND (ELM) --> (NTC (RESPOND)) (OLD **)
     (RESPONSE NO) RESPOND)
PD8: (X1 == (ELM) AND X2 == (ELM) AND READY -->
    (OLD **) (NTC (ELM)) (OLD **) (SET X2 X1))
PD9: (X1 == (ELM) AND (SET X2 X3 X4) AND READY -->
    (OLD **) (NTC (SET)) (OLD **) (SET X2 X3 X4 X1))
PD10: (X1 == (ELM) AND (SET X2 X3) AND READY -->
     (OLD **) (NTC (SET)) (OLD **) (SET X2 X3 X1))
PD11: (X1 == (ELM) AND (SET X2) AND READY -->
     (OLD **) (NTC (SET)) (OLD **) (SET X2 X1))
PD12: (READY AND (RESPONSE) ABS --> (RESPONSE YES))
PD13: (ANY --> ATTEND)
;
```

Fig. 18. PS.ST7: PS.ST5 with response bias.

*Summary*

The final production system, PS.ST7, comes close to satisfying the several empirical propositions listed earlier: the linear dependence on set size, the equality of slope for positive and negative cases, the constant difference between positive and negative cases, and the lack of a serial position effect.

However, the situation is not perfect. We can write the total response time as:

$$\begin{aligned} T = {} & \text{T.external} + 3*\text{T.evoke} \\ & + (6 + X)*\text{T.action} + N*\text{T.action} \end{aligned}$$

$$\begin{aligned} \text{where } X &= 0 \text{ for the positive case} \\ X &= 1, 2, 3 \text{ for the negative case} \\ & \quad \text{for PD7A, B, C respectively.} \end{aligned}$$

Actually, this equation contains a small addition to the constant part. If the system is actually run through both the encoding and decoding stages then (RESPONSE) gets lost from STM before it is called by (PROBE) after decoding. This can be avoided by the addition of another production that brings (RESPONSE) to the front when (PROBE) is first detected:

PDX: (READY AND (PROBE) AND (RESPONSE) --> (OLD **))

This production goes right after PD1. It marks READY as old to avoid repetition of PDX itself; READY has in fact done its job of controlling the encoding and initiating the response when (PROBE) occurs. PDX adds one T.evoke and one T.action to the constant part of T above, since it is evoked on every occasion.

The experimental value of the slope of time against set size is around 35 ms. Hence from the equation above, T.action must be around 35 ms. The difference between positive and negative cases is either 1, 2, or 3 times T.action, which is to say, either about 35, 70, or 105 ms. Empirically this difference is often found to be around 50 ms, which lies halfway between the two values for A and B. Notice that both the slope and the positive-negative difference are determined solely by T.action. T.evoke enters the equation only as part of the total ordinate. since this also contains various peripheral perception and motor response times (here symbolized by T.external), there is no way to derive any independent information about T.evoke. The best we can do is make a check of reasonableness. Since the total ordinate is around 350 ms, there is about 140 ms available for T.external + 3*T.evoke, which does not seem out of bounds if T.evoke is not too large.

There is little point in attempting to assay the seriousness of the discrepancy between the theoretical and empirical values for the positive-negative difference or to explore various potential explanations. The model is still enough within the ball park to remain worth considering. Other more pressing issues need exposing.

Let us note what the control structure has accomplished for us so far. First, we have been able to approach the task of binary classification in the Sternberg paradigm as a programming task. We could tell when an arrangement accomplished the task and when it did not.[13] Once a viable production system was discovered, all of its properties were fixed, to the extent that we had settled on an explicit timing model. Thus, explicit predictions follow for the entire range of inputs.

In this view PSG represents the basic structure of the human information processing system. It follows that *any* program written in PSG should be a viable program for the subject. Only such an assumption permits us simply to program the task in PSG. However, nothing has been provided to determine which of all the feasible production systems will come to govern the subject's behavior. Our example makes clear that the multiple production systems are possible. Without a theory of which system is selected the total view remains essentially incomplete.

General considerations of the adaptiveness of human behavior lead one to adopt the following:

> *Principle of adaptation*: Other things equal, the subject will adopt that production system that more closely obtains his goals.

It is, after all, a principle of this sort that leads us to believe that the subject will come to perform the task at all, once instructed. For we do not believe that the subject comes equipped with a preformed organization for doing the Sternberg task (before encountering it for the first time). This organization is composed in response to the demands of the task, i.e., the subject himself selects this organization, presumably from among others that he could adopt that would not solve the task. That he should also be able, say, to use one organization that takes less time than another is simply another application of the same principle.

Why then does not a subject use the more efficient

[13]We do face verifying that the program does in fact work, i.e., debugging the program. While simple for the task at hand, it can become a serious problem.

schemes, such as PS.ST1 which recognizes the action in a time independent of set size and (importantly) less than for the other systems? Resolution can be sought in several directions. Possibly the timing model is wrong, or the particular structure of PSG, or the general structure of production systems. A different sort of possibility is that additional constraints exist that limit the production systems that are possible or selected. For example, if the subject can't learn a given type of production system or assemble it on demand, then it can be excluded from the feasible set. Something of this sort, perhaps, makes us hesitate at splitting the response productions on both sides of the decoding productions in PS.ST3 (Figure 13). We have reason to be leery of the linear ordering of productions, since we do not interpret a production system as considering productions serially, but rather in parallel. If productions are not completely independent, but are developed in subsystems, arbitrary ordering may not be possible.

Notice that the set of all production systems plays a somewhat different role here than does, say, the set of all Markov processes in mathematical learning theory. In both cases the set in question is indeed the set of all theories under consideration. But with the Markov process the problem of selection is one of descriptive adequacy (i.e., of the fit to the data). In the present case, since the selection is ascribed to the subject (by a not yet formulated process, unfortunately) we must confront the issue of why psychologically one rather than another production system occurs--in addition to the question of whether it fits the data.

Leaving to one side for the moment the major issue just raised, working with the production systems has in fact led us down a somewhat new path in theorizing about the basic phenomena in the Sternberg paradigm. The basic linear effect is ascribed not to a search process but to a decoding process. This solution was discovered in the attempt to find a production system that fit the basic phenomena. One can find in the literature some suggestions that encoding may be involved (e.g., Sternberg, 1970), but no genuine presentation of such a theory is known to me. This at least illustrates that the additional level of detail of a control system theory serves to generate new hypotheses about the mechanisms involved.

This assumption about decoding is sufficiently novel and sufficiently central to the model, that it rates additional investigation. This will let us explore additional aspects of what a detailed theory of control can provide.

## The Decoding Hypothesis

We wish to explore the decoding hypothesis and attempt to discover whether it is reasonable, or whether (as introduced) it is to be viewed as a *deus ex machina* to permit the construction of a production system that happens to fit the empirical data. There are two directions (at least) in which to look. First, we can search for basic theoretical reasons why the decoding should exist. Second, we can look at other tasks to see whether they too seem to require the decoding hypothesis.

### *Why Decode?*

The argument starts from the generally accepted view (within an information processing theory of human behavior) that subjects encode stimuli ubiquitously. Hence, the argument goes, the system is simply unable to pick a production system that does not do the encoding, hence the decoding.

The argument has perhaps some force, though it is better when kept rather general. In detail, it would not seem to rule out the decoding of the set upon receipt of the ready signal, rather than the probe, so that by the time the probe came along only the instantaneous matching productions would need to be evoked. This would not be possible in the dynamic versions of the task where the set is given sequentially right up to the problem. But we know that the behavior in the static task (the positive set in LTM) and the dynamic task (the positive set given each time) are essentially the same. Thus we must still face the issue: Why not decode the positive set into STM at the ready signal?

Let us return to the question of adaptive behavior raised in the prior section in a more pointed way: Why should the subject encode and decode a set rather than leave it in STM where the task can be performed in a single recognition (as in PS.ST1)? Consider the following assumption:

> *Assumption of Unreliable STM*: The contents of STM are sufficiently variable, noisy and unreliable that the subject will adopt production systems with lower risk from STM unreliability.

Unreliability of STM could be the case because it fades rapidly or because it is the confluence of uncontrolled input from many sources, both from LTM and from perception. The production system itself is consonant with such a view. Imagine, as argued earlier, that the small production system that we use to describe the program of the subject is really embedded in a very large system. From time to time other productions may be evoked instead of the ones in our set. The only effect of these, mostly, may be to add junk to the memory and to add some time to performance (a few T.evokes and T.actions). From a control point of view the process looks like cycle-stealing (as it goes on in most computers today for input/output). From a data point of view it makes the STM unreliable.

Given such a situation the rational way to obtain reliable behavior is to work with programs that are as safe as possible--in which the parts of the program are positively coupled. In the case at hand, if the total organization (our PS.ST7) both dumps the elements into STM and then tests for a match, then the test production can operate with the knowledge that the elements of the set are all there. It is a reliable method for solving the problem. If the system (PS.ST1) simply scans whatever is in STM at the probe signal when the set was dumped earlier at the ready signal, then it is not safe. The chance of a spurious NO is appreciable and even the chance of a spurious YES increases. What if the subject thinks about some possible element during the interval between READY and PROBE--he has no way of guaranteeing that he will be able to distinguish it from a true element. Note that he cannot process such a stray thought, since processing conflicts with being prepared to react to the PROBE when it comes.

This argument essentially introduces a second criterion, reliability, in addition to speed as a governor of the production system that the subject will construct. We have thereby preserved the principle of adaptation. Against this we have only a qualitative notion so far of how to assess the reliability (as seen by the subject) of a proposed production system. In the case at hand, an *ad hoc* argument goes some ways toward establishing that the speedier production system is less reliable than the slower one (which is also the empirically correct one). We should at least package this assumption in a principle:

> *Principle of Coupled Systems*: When attempting to behave reliably the subject uses production systems where early evoked productions produce guarantees on the contents of STM that can be utilized by later productions (thereby coupling the productions together).

The argument above leads directly to two qualitative hypotheses, one rather easy to verify, another much harder. First, if the selection of PS.ST7 over PS.ST1 is due to a requirement for reliability, then releasing that requirement should move subjects to adopt PS.ST1. As mentioned at the beginning of the paper, the conditions for the Sternberg paradigm are a low error rate (of the order of a few percent). If one permitted much higher error rates and paid off for speed only, one should see the slope disappear. It is unknown of course, how much the error rate would go up, since selection of the reliable system is based on a choice of the subject in the face of a task demand, not on demonstrated failure of the faster algorithm. This experiment should be rather easy to carry out and indeed the essential facts may already be known (though I don't know them).

The second hypothesis comes from noting that we have an instance of the speed-accuracy trade-off, which is a general phenomenon much studied in the literature. One of the features of that literature (which we cannot review here) is that no mechanisms are proposed as to how a speed-accuracy trade-off is possible. One often proposes to represent such a trade-off by a criterion parameter which can be changed. But (to my knowledge) this never is embedded within a model for how such a parameter effects a shift to greater speed at the expense of accuracy or vice versa. The hypothesis then is that the space of feasible programs is indeed relatively large and that selection (construction) of different production systems with slightly different speeds and reliabilities provides the underlying ability of the subject to trade off speed for accuracy. Within this hypothesis, the freedom of programmability of production systems, far from being a disturbing theoretical feature (reflecting a preference that a unique production system exist for a task), is an essential aspect of the human information processing system.

We state these two hypotheses to point out how having a specific theory of the control system is able to generate hypotheses of the rather global nature long favored by experimental psychology.

## *Memory Span*

A major advantage of a theory of the control system is the applicability of the theory to a wide range of tasks. One should be able to test an hypothesis, such as the decoding hypothesis, against its indicated use in other tasks. A particularly transparent task from this viewpoint is the standard auditory memory span test.

We can take the task as receiving a sequence of elements, each of which can be perceived as a chunk. When the signal to repeat occurs, the subject is to repeat the sequence exactly.

Figure 19 gives a production system PS.MS1, for performing the memory span test in the most obvious way. The subject lets the elements accumulate in STM and then, upon REPEAT, proceeds to respond with each one. It keeps from repeating an element by marking each element used. Thus, we get a production system of only three productions: PD1 to emit the response and mark old; PD2 to terminate the trial by deactivating REPEAT when no more elements are left; and PD3 to attend to the environment. We do not include an initial ready signal in this simple version.

Figure 20 gives a run of PS.MS1 on a sequence of three elements. We have modified the executive structure so that the ATTEND operator goes to a list, STIMULUS (given at the top of the figure), and attends to each symbol successively. Although all members of the sequence are emitted, the system does not obtain them in the correct order. A moment's consideration shows that this is not a fluke. The STM is indeed a stack-like memory which performs generally in a last-in first-out manner.

How can this order be reversed? There are two directions to explore: reversing at response time; and reversing at input time so that the response process works off something in the right order. Let's consider each in turn.

Simply tring to pick up the last element of a given type in STM appears difficult. The subject (i.e., the production system) knows about the elements only that they belong to the same type (e.g., are marked ELM). The nature of the match is such that the more

```
PS.MS1: (PD1 PD2 PD3)
;
PD1: ((ELM X1) AND REPEAT --> (OLD **) (RESPONSE X1) RESPOND)
PD2: (REPEAT AND (ELM) ABS --> (OLD **))
PD3: (ANY --> ATTEND)
;
```

Fig. 19. PS.MS1: Simple PS for memory span.

VISUAL INFORMATION PROCESSING

```
STIMULUS: ((ELM A) (ELM B) (ELM C) REPEAT)

PS.MS1 START!
0.  STM: (JUNK NIL NIL NIL NIL NIL NIL)
PD3 TRUE
0.  ACTION- ATTEND
ATTEND TO: (ELM A)
1.  ACTION- (ELM A)
2.  STM: ((ELM A) JUNK NIL NIL NIL NIL NIL)
PD3 TRUE
2.  ACTION- ATTEND
ATTEND TO: (ELM B)
3.  ACTION- (ELM B)
4.  STM: ((ELM B) (ELM A) JUNK NIL NIL NIL NIL)
PD3 TRUE
4.  ACTION- ATTEND
ATTEND TO: (ELM C)
5.  ACTION- (ELM C)
6.  STM: ((ELM C) (ELM B) (ELM A) JUNK NIL NIL NIL)
PD3 TRUE
6.  ACTION- ATTEND
ATTEND TO: REPEAT
7.  ACTION- REPEAT
8.  STM: (REPEAT (ELM C) (ELM B) (ELM A) JUNK NIL NIL)
PD1 TRUE
8.  ACTION- (OLD **)
9.  ACTION- (RESPONSE X1)
10.  ACTION- RESPOND
11.  ACTION- (NTC (RESPONSE ANY))
12.  ACTION- (SAY ANY)

********** C

13.  ACTION- (OLD **)
14.  STM: ((OLD (RESPONSE C)) (OLD (ELM C)) REPEAT (ELM B) (ELM A) JUNK NIL)
PD1 TRUE
14.  ACTION- (OLD **)
15.  ACTION- (RESPONSE X1)
16.  ACTION- RESPOND
17.  ACTION- (NTC (RESPONSE ANY))
18.  ACTION- (SAY ANY)

********** B

19.  ACTION- (OLD **)
20.  STM: ((OLD (RESPONSE B)) (OLD (ELM B)) REPEAT (OLD (RESPONSE C))
        (OLD (ELM C)) (ELM A) JUNK)
PD1 TRUE
20.  ACTION- (OLD **)
21.  ACTION- (RESPONSE X1)
22.  ACTION- RESPOND
23.  ACTION- (NTC (RESPONSE ANY))
24.  ACTION- (SAY ANY)

********** A


```

Fig. 20.  Run of PS.MS1.

recent elements will be selected first. Thus, the only way to get the last element is by brute force--by productions that latch onto all preceding elements. One needs a set of productions of the form:

```
X1 AND X2 AND X3 AND X4 --> ...
X1 AND X2 AND X3 --> ...
X1 AND X2 --> ...
X1 --> ...
```

While this bears some resemblance to the encoding productions, it still seems like an uncomfortable way to do business.

An alternative strategy is to mark each element as it enters in a unique way so that that production system can know about the first one. This essentially produces an STM paired-associate structure, e.g.,

```
STM:  (... (ELM3 C) ... (ELM2 B) ...(ELM1 A) ...)
```

With this arrangement the response productions have to be an explicit set, knowing first to respond with (ELM1), then with (ELM2), etc. Again, it seems a possible, but awkward strategy. However, an attempt on the part of a subject to use the 1-BUN, 2-SHOE, ... mnemonic on the memory span test would be an application of this. (General experience is that presentation rates of 1 symbol/sec are too fast for this.)

As a final example of the reverse-while-responding strategy, the system could respond internally as in Figure 20, which reverses the order, and then respond again externally, thus emitting them in the right order. This is also a conceivable strategy and in slightly different circumstances can be detected (e.g., in reciting an alphabet backwards, McLean & Gregg, 1967). It seems an unlikely strategy in the simple memory span. It should produce a substantial delay before the first response; further, the task of repeating the set backwards should be easier than repeating it forwards and should not have the delay. Empirically these seem not to be the case.

Turning to strategies of reversing on input, the attempt to do this for each element at each moment of input creates a fair amount of thrashing, in which the set of already ordered elements must be brought in front of each new element and still left in the same order.

A second scheme is to encode the elements on input, just as we have done for the Sternberg task. This leaves a single chunk in STM which is decoded in the right order at response time. Figure 21 gives a production system, PS.MS2, for this encoding. To show the relationship to the Sternberg task we have labeled the productions with the ones they correspond to in PS.ST7 (Figure 18), the final production system for the Sternberg task. Productions PD1 and PD1.1 are the response productions and are unique to the task. Production PD1.1 is the response production for the memory span task, and takes the place of PD6 and PD7 in the Sternberg task. PD12 in the Sternberg task sets the response bias. This is not a feature of the memory span task, so it is missing as well. Corresponding productions are not all identical. The encoding productions (PD8 - PD11) are the same. However, the decoding productions (PD2 - PD5) are responsive to REPEAT rather than to (PROBE). To make them identical would require another level of indirectness--one that might be expected perhaps in the early stages of performance (when the subject, in effect, must interpret

```
PS.MS2: (PD1 PD1.1 PD2 PD3 PD4 PD5 PD8 PD9 PD10 PD11 PD13)
;
PD1: (REPEAT AND (ELM) ABS AND (SET) ABS --> (OLD **))
PD1.1: ((ELM X1) AND REPEAT --> (OLD **) (RESPONSE X1) RESPOND)
PD2: ((SET X1 X2 X3 X4) AND REPEAT --> (OLD **) X4 X3 X2 X1)
PD3: ((SET X1 X2 X3) AND REPEAT --> (OLD **) X3 X2 X1)
PD4: ((SET X1 X2) AND REPEAT --> (OLD **) X2 X1)
PD5: ((SET X1) AND REPEAT --> (OLD **) X1)
PD8: (X1 == (ELM) AND X2 == (ELM) AND READY -->
     (OLD **) (NTC (ELM)) (OLD **) (SET X2 X1))
PD9: (X1 == (ELM) AND (SET X2 X3 X4) AND READY -->
     (OLD **) (NTC (SET)) (OLD **) (SET X2 X3 X4 X1))
PD10: (X1 == (ELM) AND (SET X2 X3) AND READY -->
      (OLD **) (NTC (SET)) (OLD **) (SET X2 X3 X1))
PD11: (X1 == (ELM) AND (SET X2) AND READY -->
      (OLD **) (NTC (SET)) (OLD **) (SET X2 X1))
PD13: (ANY --> ATTEND)
;
```

Fig. 21. PS.MS2: PS for memory span, with encoding.

the signal in terms of a common meaning--to decode), but would presumably be adapted out with practice. Finally, PD1, which recognizes the end of the task, is responsive to different features in the two tasks. Figure 22 shows a run of PS.MS2 on a three element sequence, which can be seen to perform appropriately.

Let us summarize. Substantively, we have found that the encoding hypothesis is not only consistent with behavior in another distinct task, but provides an appropriate solution to a difficulty (the ordering) that arises from the application of a naive formulation. We showed, however, that it was not the only way to overcome the difficulty. Some of the alternatives, despite our disparagement, clearly represent alternatives to be considered further. We indicated some other tasks in which they appear to operate. Nevertheless, the encoding hypothesis comes through appearing substantially less *ad hoc*.

Methodologically, we say that it was relatively easy to move to a new task and to construct a theory that had substantial contact with the initial one. With a little care one could insist that exactly the same theory (i.e., the same total production system) be able to perform both tasks. To be sure, some of the productions will be unique to each task. Indeed, they must be if the unique aspects of a task are to be represented.

In seeking support for the decoding hypothesis in the phenomenon of response order we have taken the structure of the STM to be fixed. As we observed earlier, it is the last-in first-out character of the STM that creates this problem and makes it a fundamental one. Alternatively, the solution might lie in changing the structure of the underlying system. One can certainly construct STM models that have a first-in first-out character and thus make the response order identical to input order. However, such systems must ultimately have other problems. For the underlying empirical reality is that humans appear to behave in positive time order (first-in first-out) in the short run and in inverse time order (last-in first-out) in the long run.

VISUAL INFORMATION PROCESSING

```
STIMULUS: (READY (ELM A) (ELM B) (ELM C) REPEAT)

PS.MS2 START!
0.  STM: (JUNK NIL NIL NIL NIL NIL NIL)
PD13 TRUE
0.  ACTION- ATTEND
ATTEND TO: READY
1.  ACTION- READY
2.  STM: (READY JUNK NIL NIL NIL NIL NIL)
PD13 TRUE
2.  ACTION- ATTEND
ATTEND TO: (ELM A)
3.  ACTION- (ELM A)
4.  STM: ((ELM A) READY JUNK NIL NIL NIL NIL)
PD13 TRUE
4.  ACTION- ATTEND
ATTEND TO: (ELM B)
5.  ACTION- (ELM B)
6.  STM: ((ELM B) (ELM A) READY JUNK NIL NIL NIL)
PD8 TRUE
6.  ACTION- (OLD **)
7.  ACTION- (NTC (ELM))
8.  ACTION- (OLD **)
9.  ACTION- (SET X2 X1)
10.  STM: ((SET (ELM A) (ELM B)) (OLD (ELM A)) (OLD (ELM B)) READY JUNK NIL NIL)
PD13 TRUE
10.  ACTION- ATTEND
ATTEND TO: (ELM C)
11.  ACTION- (ELM C)
12.  STM: ((ELM C) (SET (ELM A) (ELM B)) (OLD (ELM A)) (OLD (ELM B)) READY JUNK NIL)
PD10 TRUE
12.  ACTION- (OLD **)
13.  ACTION- (NTC (SET))
14.  ACTION- (OLD **)
15.  ACTION- (SET X2 X3 X1)
16.  STM: ((SET (ELM A) (ELM B) (ELM C)) (OLD (SET (ELM A) (ELM B))) (OLD (ELM C))
          READY (OLD (ELM A)) (OLD (ELM B)) JUNK)
PD13 TRUE
16.  ACTION- ATTEND
ATTEND TO: REPEAT
17.  ACTION- REPEAT
18.  STM: (REPEAT (SET (ELM A) (ELM B) (ELM C)) (OLD (SET (ELM A) (ELM B)))
          (OLD (ELM C)) READY (OLD (ELM A)) (OLD (ELM B)))
PD3 TRUE
18.  ACTION- (OLD **)
19.  ACTION- X3
20.  ACTION- X2
21.  ACTION- X1
22.  STM: ((ELM A) (ELM B) (ELM C) (OLD (SET (ELM A) (ELM B) (ELM C)))
          REPEAT (OLD (SET (ELM A) (ELM B))) (OLD (ELM C)))
PD1.1 TRUE
22.  ACTION- (OLD **)
23.  ACTION- (RESPONSE X1)
24.  ACTION- RESPOND
25.  ACTION- (NTC (RESPONSE ANY))
```

Fig. 22.  Run of PS.MS2.

VISUAL INFORMATION PROCESSING

```
26. ACTION- (SAY ANY)

********** A

27. ACTION- (OLD **)
28. STM: ((OLD (RESPONSE A)) (OLD (ELM A)) REPEAT (ELM B) (ELM C)
        (OLD (SET (ELM A) (ELM B) (ELM C))) (OLD (SET (ELM A) (ELM B))))
PD1.1 TRUE
28. ACTION- (OLD **)
29. ACTION- (RESPONSE X1)
30. ACTION- RESPOND
31. ACTION- (NTC (RESPONSE ANY))
32. ACTION- (SAY ANY)

********** B

33. ACTION- (OLD **)
34. STM: ((OLD (RESPONSE B)) (OLD (ELM B)) REPEAT (OLD (RESPONSE A))
        (OLD (ELM A)) (ELM C) (OLD (SET (ELM A) (ELM B) (ELM C))))
PD1.1 TRUE
34. ACTION- (OLD **)
35. ACTION- (RESPONSE X1)
36. ACTION- RESPOND
37. ACTION- (NTC (RESPONSE ANY))
38. ACTION- (SAY ANY)

********** C

39. ACTION- (OLD **)
40. STM: ((OLD (RESPONSE C)) (OLD (ELM C)) REPEAT (OLD (RESPONSE B))
        (OLD (ELM B)) (OLD (RESPONSE A)) (OLD (ELM A)))
PD1 TRUE
40. ACTION- (OLD **)
41. STM: ((OLD REPEAT) (OLD (RESPONSE C)) (OLD (ELM C)) (OLD (RESPONSE B))
        (OLD (ELM B)) (OLD (RESPONSE A)) (OLD (ELM A)))
PD13 TRUE
41. ACTION- ATTEND
END: NO PD TRUE
```

Fig. 22 (continued).

Thus, there is a reversal at some stage (from primacy to recency, if you like to think of it that way) and the structure of the system must account for both aspects.

## Applications of the Theory

We have now developed a theory of the simple Sternberg binary classification task that has modest standing. It should be possible to apply it to the experiments discussed in this symposium that make use of similar task situations. To do this properly requires that we extend the theory to these variant situations, much as we did to the memory span task, keeping as much communality with the original situation as possible. However, there is a limit to an introductory paper and to go into the results of Posner (Chapter 2) and Hayes (Chapter 4) in detail exceeds those limits. Thus, we must be content with a cursory examination of a few aspects. Methodologically, we can make a virtue of this restriction, since it provides the opportunity to apply the theory in a qualitative way, thereby illustrating how such applications might go.

## *Perceptual Enhancement*

The brief discussion in Posner's paper on the phenomenological experience of perceptual enhancement of the successful item in a Neisser paradigm offers a simple example. He observes that Cavanagh and Chase (1971) found that in a Sternberg task with two probes (one positive, one negative) the positive one only was enhanced. Posner's argument was that this controverted the use of the enhancement as an indicator of the boundary between pre-attentive and attentive processes, since much attentive processing (i.e., the search) went on prior to the enhancement and did so for both probes.

The present model offers a somewhat different characterization. Presenting two probes rather than one has no effect on the linear-time component, which is the decoding time. It might have an effect on the intercept if the two probes are themselves encoded in some way, or enter STM serially. One and only one of the probes evokes the positive production (PD6). The other probe simply does not evoke anything. Thus a single decoding operates for both probes.[14]

Examination of the production system puts the

---

[14]The actual slopes are somewhat higher than the usual 35 ms. This complicates the interpretation. It suggests (as only one among several alternatives) that some subjects may have processed each probe separately and that the data represent a mixture of methods.

enhancement effect on PD6, which is to say on the multiple occurrence of a variable in the matching. This offers a clue about how one might explore the details of the match processes. However, the present model does not offer a clear interpretation of pre-attentive versus attentive processes. First of all, the model does not include a perceptual component so that one can determine whether the match is or is not part of the same apparatus that carries out perception. No matter how one determines the latter question, the match (the selection of the next production), and hence the enhancement, is involved intimately with whatever can be called attentive processes.[15]

Having gone this far, it is tempting to state a hypothesis about the locus of conscious experience. It is not to be associated with the content of any memory, not even of STM which defines in an operational sense what the subject is momentarily aware of, i.e., to what he can respond to in the next tens of milliseconds. Rather, phenomenal consciousness is to be associated with the *act* of matching, and its content is given by the set of STM items extracted by the matched condition. Thus, it is an ephemeral fleeting thing that never stays quite put and never seems to have clearly defined edges (the never-step-into-the-same-river-twice phenomenon). It seems like an interesting hypothesis. That the hypothesis can be stated in such a precise form is attributable to having a detailed model of the control structure.

## *Recency Effects*

Posner's paper discusses several Sternberg-like tasks in detail. A prominent feature of his data is

[15]The diffuseness of this discussion only shows that each theory puts its own classification on phenomena and one cannot easily discuss one in terms of the other (attentive versus pre-attentive derive from a certain rough model of the total machinery).

the non-linear relation to positive set size. This leads him to plot all of his graphs against the logarithm of set size, since this tends to linearize the curves somewhat. This decision of how to display the data makes me uncomfortable, I confess, since it seems not to be theoretically motivated. In fact it serves to obscure, rather than clarify the explanation Posner provides in passing. He notes that the effect may be a recency effect on the first item, namely, that subjects respond more quickly to sets of size one than to larger sets. If this is so, then the curves should be linear for set sizes greater than one. However, all the data are limited to three sizes, 1, 2 and 4, and thus no direct empirical test of this is possible.

This recency phenomenon appears to be not unknown elsewhere in the literature on the Sternberg task and seems to be associated with dynamic presentation--defining the set just prior to test--with a relatively short delay between set definition and probe. Posner's experiments fit this format, since they run from set to probe continuously (at half second pacing) and without warning.

An explanation is not far to seek within the present theory, consisting of both the production system framework and the decoding hypothesis. With set size of one the system delays encoding until the second element arrives. If instead the probe arrives, then there is no decoding step; rather, the system simply responds. In fact, if one runs the full range of set sizes one finds the recency effect. From the formula given earlier, which expresses the correct linear growth,[16] one gets:

T(1) = 3*T.evoke + 6*T.action + 1*T.action

= 3*T.evoke + 7*T.action

[16]In deriving that formula we simply did not reflect the special circumstances of the special case. A careful enough analysis would have revealed it, of course, and perhaps the perspicacious reader in deriving it independently detected the flaw.

The measured value is:

T(1)' = 2*T.evoke + 5*T.action

This provides a difference of T.evoke + 2*T.action, which is something in excess of 70 ms, taking the 35 ms figure for T.action. This is somewhat high for the measured values, which run 40 - 60 ms. As with the discrepancy on the response bias, we do not know whether or not to be disturbed by the approximate fit. Basically, the ambiguity of interpretation arises because the experimental numbers are averages over trials and over subjects. This means they are undoubtedly generated by mixtures of strategies to some unknown extent.

Posner's Figure 2 shows a strong serial position effect for a set size of four. This is a recency effect in which the last item (the fourth) is processed about 50 ms faster than the other three, which are reasonably constant. Our theory as it stands does not handle this, since it produces the recency phenomenon only for sets of one. We can extend it to the new situation, however, if we assume that the subject can react to the last element directly, even though he has also encoded it. The size of the effect indicates that this happens sometimes, but not always, so that the data would be a mixture of two ways of doing the task. If this is the explanation, we should also find recency effects for the other set sizes.

In general terms, such an explanation is consistent with the nature of production systems. There is no reason why the responding production (PD6) should not pick up the data of the unencoded element directly. In fact the ability to short circuit a longer process and to mix methods would seem to be a major point in favor of production systems, providing a detailed explanation for variety and lability of behavior. However, as our experience on the several production systems should indicate, it may not be trivial to construct the production system to get the recency result. We may find that it works just as well on all members of the set, if we fix it up to work on the most recent.

Whereas recency seems consistent with the unreliability assumption of STM, so that the subject might trust the most recent one but not the older ones, the system may not be able to tell the two situations apart. We mention these potential difficulties to indicate the gap between having the right sort of theory and having it deliver the right predictions in detail.

*Continuous Sternberg Experiment*

Enough work has been done with the Sternberg paradigm to accumulate a number of experiments whose interpretation appears to pose extreme difficulties. One of these is an experiment by Sternberg and Scarborough (1969). Unfortunately it has not been replicated nor extended, but it is still worth attempting an explanation in terms of the present theory.

Briefly, a subject was given a fixed positive set. Then he was tested with 20 probes in sequence. Exactly one probe was positive or,none was. The time between probes was 70 ms, so the entire set of 20 probes went by in under 1.5 seconds. The subject was to react to the positive probe in the usual way. The result: the reaction time was identical to that in the basic task, being a linear function measured from the time of the probe, with a slope of about 35 ms and an intercept of about 350 ms.

This result is extremely difficult for search theories to deal with. Sternberg and Scarborough erect an *ad hoc* pipeline processing system with stages for each probe. The present theory produces the essential result on the assumption that the probes trigger the decoding of the set, thus filling STM with both probes and elements. Due to the unreliability of STM, if a hit gets made, the set is decoded again to confirm the hit.

Figure 23 gives a production system, PS.CST1, for the continuous Sternberg task. It differs somewhat, as it must, from PS.ST7, the production system for the basic task. We have kept the names of productions the same, so that the correspondence is evident. Mostly,

**VISUAL INFORMATION PROCESSING**

```
PS.CST1: (PD1 PD1.1 PD2 PD3 PD4 PD5 PD6 PD12 PD13)
;
PD1: ((MARK) AND (OLD (RESPONSE)) --> (OLD **))
PD1.1: ((PROBE <DIGIT>) AND (ELM <DIGIT>) AND (RESPONSE) ABS -->
        (MARK **) (RESPONSE YES) POSITIVE.SET)
PD2: ((SET X1 X2 X3 X4) AND (PROBE) --> (OLD **) X4 X3 X2 X1)
PD3: ((SET X1 X2 X3) AND (PROBE) --> (OLD **) X3 X2 X1)
PD4: ((SET X1 X2) AND (PROBE) --> (OLD **) X2 X1)
PD5: ((SET X1) AND (PROBE) --> (OLD **) X1)
PD6: ((MARK (PROBE <DIGIT>)) AND (ELM <DIGIT>) --> RESPOND)
PD12: (READY AND (SET) ABS (OLD (SET)) ABS --> POSITIVE.SET)
PD13: (ANY --> WAIT)
;
```

Fig. 23. PS.CST1: PS for continuous Sternberg task.

productions drop out. Since the subject has the set in LTM, no encoding productions are needed (though they could have been left in the system). Instead, PD12 is modified to put the positive set into STM, either on the ready signal or whenever there is an indication that some elements might be lost from STM. The cues to this are there not being any set in STM, either undecoded--(SET) ABS--or decoded--(OLD (SET)) ABS.[17] Thus, the system dumps sets into STM at every indication, so to speak, in an attempt to avoid losing some elements of the positive set from STM.

Decoding of a set takes place whenever there is a set in STM to be decoded and a probe to initiate it. Since there is a continuous stream of probes (once they start), decoding takes place immediately (and produces small refractory periods). The task itself dictates the removal of the negative response production (PD7), since the test is only for presence. (Actually, the production system could have been expanded to say NO at the end of the sequence.) The positive response production (PD6) is modified to only sense an identical probe and set element with a marked probe (with MARK). The key production is PD1.1, which responds to an

[17]The vigilant reader will notice an error in the figure, namely the AND missing between two condition elements of PD12. The interpreter does not in fact require the AND. Thus it behaved correctly, so that the error was not noticed until later.

VISUAL INFORMATION PROCESSING

```
POSITIVE.SET: (SET (ELM 4) (ELM A))

PS.CST1 START!
0.  STM: (JUNK NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL)
PD13 TRUE
0.  ACTION- WAIT
      INPUT FORCED STIMULUS (IF ANY) = READY
1.  STM: (READY WAIT JUNK NIL NIL NIL NIL NIL NIL NIL NIL)
PD12 TRUE
1.  ACTION- POSITIVE.SET
2.  STM: (POSITIVE.SET READY WAIT JUNK NIL NIL NIL NIL NIL NIL NIL)
PD13 TRUE
2.  ACTION- WAIT
3.  STM: (WAIT POSITIVE.SET READY WAIT JUNK NIL NIL NIL NIL NIL NIL)
PD13 TRUE
3.  ACTION- WAIT
      INPUT FORCED STIMULUS (IF ANY) = (PROBE 1)
4.  STM: ((PROBE 1) WAIT WAIT POSITIVE.SET READY WAIT JUNK NIL NIL NIL NIL)
PD4 TRUE
4.  ACTION- (OLD **)
5.  ACTION- X2
      INPUT FORCED STIMULUS (IF ANY) = (PROBE 2)
6.  ACTION- X1
7.  STM: ((ELM 4) (PROBE 2) (ELM A) (OLD POSITIVE.SET) (PROBE 1)
         WAIT WAIT READY WAIT JUNK NIL)
PD13 TRUE
7.  ACTION- WAIT
      INPUT FORCED STIMULUS (IF ANY) = (PROBE 3)
8.  STM: ((PROBE 3) WAIT (ELM 4) (PROBE 2) (ELM A) (OLD POSITIVE.SET) (PROBE 1)
         WAIT WAIT READY WAIT)
PD13 TRUE
8.  ACTION- WAIT
9.  STM: (WAIT (PROBE 3) WAIT (ELM 4) (PROBE 2) (ELM A) (OLD POSITIVE.SET) (PROBE 1)
         WAIT WAIT READY)
PD13 TRUE
9.  ACTION- WAIT
      INPUT FORCED STIMULUS (IF ANY) = (PROBE 4)
10.  STM: ((PROBE 4) WAIT WAIT (PROBE 3) WAIT (ELM 4) (PROBE 2) (ELM A)
          (OLD POSITIVE.SET) (PROBE 1) WAIT)
PD1.1 TRUE
10.  ACTION- (MARK **)
11.  ACTION- (RESPONSE YES)
      INPUT FORCED STIMULUS (IF ANY) = (PROBE 5)
12.  ACTION- POSITIVE.SET
13.  STM: (POSITIVE.SET (PROBE 5) (RESPONSE YES) (MARK (PROBE 4)) (ELM 4)
          WAIT WAIT (PROBE 3) WAIT (PROBE 2) (ELM A))
PD4 TRUE
13.  ACTION- (OLD **)
      INPUT FORCED STIMULUS (IF ANY) = (PROBE 6)
14.  ACTION- X2
15.  ACTION- X1
      INPUT FORCED STIMULUS (IF ANY) = (PROBE 7)
16.  STM: ((PROBE 7) (ELM 4) (ELM A) (PROBE 6) (OLD POSITIVE.SET) (PROBE 5)
          (RESPONSE YES) (MARK (PROBE 4)) (ELM 4) WAIT WAIT)
PD6 TRUE
16.  ACTION- RESPOND
17.  ACTION- (NTC (RESPONSE ANY))
18.  ACTION- (SAY ANY)

**********  YES
```

Fig. 24.  Run of PS.CST1.

identical probe and set element by marking the probe and reinitializing the positive set. This realizes the checking assumption.

Figure 24 shows a run of PS.CST1 with a two element set, consisting of (ELM 4), to be matched to the probe, and (ELM A), the irrelevant one. The executive for the run was modified so that it came to the console on almost every other action. At 35 ms per action, this approximated a 70 ms interstimulus duration. The experimenter forced an element into STM at each of these times, starting with READY and then, after a slight wait, a sequence of probes. Examination of the run shows that it reacts to (PROBE 4) appropriately, marking it, going through another decode and responding YES, despite the fact that other probes are being entered throughout.

The system deals with the main effect in an appropriate way. It would appear to have a slightly higher intercept, which was not found in the experiment. However, this is an uncertain measure, since the absolute value of the intercept is always contaminated. Also, a somewhat higher error rate might be expected, due to the chances of missing the match with PD1.1 if the probe arrives and STM has just lost the key set element. However, experimentally the error rate remained low. It is possible that the scheme of PS.CST1 is in fact relatively reliable, but it requires more exploration than has been done.

## *A Difficult Experiment*

The impression should not be that the theory is unchallenged. The total set of Sternberg-like experiments is too diverse for that. For instance, the theory appears to have great difficulty with another experiment reported by Sternberg (1970). The positive set (digits) is stored in LTM and its transmission into STM is held in abeyance by an auxiliary STM task of remembering a set of letters. Sometimes the subject gets a probe digit to classify as in the positive set or not. Sometimes he gets a signal to repeat the letter set, which helps to assure that he attends to the letter set prior to the signal. The result is a slope of about twice that of the normal paradigm (which was run as a control)--namely, 80 ms versus 40. The intercept is also higher by about 100 ms in the experimental situation.

Sternberg interprets the higher slope as being due to the time to transmit the positive set from LTM to STM, which is a close analog of the decoding hypothesis. The difficulty for the present theory is that, if this is a decoding, then the slope should be exactly the same as in the control case, since both have involved one act of decoding. Alternative interpretations are always possible, but none has occurred that comes close to resolving this experimental result.

## Conclusion

Let us sum up what we have done in this paper. (1) We introduced the notion of a control structure. (2) We introduced a general class of systems--production systems--that could serve as models of the human control system. (3) We developed in detail a specific production system--PSG--which incorporated assumptions about the structure of the human information processor. (4) We exercised the theory on the basic Sternberg binary classification experiment, which led to an additional psychological assumption--the decoding hypothesis. (5) We pursued in lesser detail some other applications--the memory span and some aspects of the experiments in Posner's paper.

Our intent throughout has been jointly substantive and methodological and we have mixed the two thoroughly. In the remainder of the conclusion we will attempt to sort out the main points and issues.

### *Production Systems as Theories*

Production systems offer an explanation of human behavior at the information processing level (Newell & Simon, 1972). They are only one of many forms of programming system that can be used to describe behavior in information processing terms. As we have seen in PSG, the production system itself has become the carrier of the basic psychological assumptions--the system architecture of PSG is taken to be the system architecture of the human information processing system. In this respect these systems represent an evolution beyond programming language systems, such as LISP, IPL, SNOBOL (and even more, ALGOL and FORTRAN). In these earlier systems the programming language was an essentially neutral affair, designed for the user to write his specific systems. In production systems, as represented by PSG, any particular set of productions represents a possible momentary performance organization of a human subject.

The evolution to a theory-laden programming language, to use a term of Pylyshyn, appears to me a major advance. By the same coin, however, the language is not neutral, so that variations in the psychological theory imply variations in the programming system. A moment's reflection will show how wide is the potential variation in system architecture. The STM can be run according to many disciplines: last-in first-out, as now; first-in first-out, which preserves order; random replacement in a fixed set of addressable cells; a circulating loop, which provides another form of rehearsal, etc. The matching rules can be varied: no multiple variables in the condition; only single levels in the condition (not nested expressions); no recognition of absence; etc. The operations can be varied: a decoding operation that simply dumps the contents into STM, rather than the encoding operation as now; etc. The selection of productions can be varied: more than one satisfied production producing a psychologically meaningful conflict state; evocation of a production leading to an automatic refractory state that inhibits re-evocation immediately; etc. The timing model can be varied: parallel processing in the action sequence; matching time dependent on the elements in the satisfied condition.

Listing many alternatives emphasizes that PSG is only one member of the class of psychologically relevant production systems. Despite this variety, production systems as a class incorporate some psychological assumptions that seem highly plausible. One is the recognize-act cycle of activity in which the human continually recognizes some features in the situation and acts accordingly. Another is making the locus of the condition correspond to those aspects of the situation that the subject is momentarily aware of, and the identification of this as the relevant short term memory. Yet another, though it applies to a somewhat narrower class of systems, is the incorporation of encoding into all STM processing, not simply as an added mechanism.

The structure of production system models, as we have described them here, are seriously deficient in several respects. They do not model the perceptual component, including the various buffer memories and the control interface between perceptual structures and the contents of STM (see Newell, 1972). They do not model LTM, especially the acquisition of new information. We took the contents of LTM as consisting of productions, but never defined the way new productions were to be created. They do not model the motor apparatus, including the control interface to the contents of STM and the actions of productions. These missing aspects cripple the model with respect to many phenomena, though there is no reason why the model should not be extended appropriately.

## *Completeness*

Production systems, like other programming systems and mathematical theories, are complete in the sense of producing theoretical consequences that are deductions from the theory. We are interested also in completeness of another sort. Is the theory complete for the phenomena of interest? Does it provide a vehicle of sufficient richness and scope to model what appears to need modeling? Production system models, like other so-called simulation models, seem to have this completeness. This is often expressed by saying that they perform what they model. Thus PS.ST7 not only is a theory of binary classification; it can *do* binary classification. As long as the interest of the psychologist remains focussed on the performance of the task, including its behavioral details, a production theory claims theoretical coverage (though of course it can be dead wrong in its predictions).

It is useful to compare this situation with some of the other techniques we currently use for describing our processing theories. As commented upon in the companion paper (Newell, this volume, Chapter 6), the theoretical structure of work on the immediate processor has been dominated by the classification of mechanisms. We have serial versus parallel, exhaustive versus self-terminating, attentive versus preattentive, and so on. Such terms hold low-level generalizations resulting from the experimental studies. Suppose PS.ST7 were the actual mechanism. Is the human, then, a serial or a parallel system? It appears to be parallel on selecting productions, serial on executing micro-sequences of actions, parallel on examining STM, serial on the order of that parallel examination as revealed by shielding of one STM element by another. Is its search exhaustive or self-terminating? Within a given task there are production systems of each type. Slightly more complex systems would yield strategies that mix the type of search conditionally within a given trial. Is something pre-attentive or attentive? We found it hard to ascertain that as well. The point is not that a given system does not give rise to classifications. The present system has sharp distinctions, e.g., between the use of STM and of the variable memory, or between sequences of actions and the evocation of a sequence of recognitions on STM. The point is that the existing classifications don't seem to help much in describing more complete systems.

Flow diagrams have become a primary vehicle for expressing theories of processing, and they represent a substantial advance on the simple classification of mechanisms. There is an example in the paper by Cooper and Shepard (Chapter 3) in the present symposium, which summarizes well a processing structure that might give rise to their experimental results.

## VISUAL INFORMATION PROCESSING

What is the relationship between production systems and flow diagrams as they are used in the psychological literature? The flow diagram provides a precise model of control flow--of what follows what.[18] It provides a frame within which informal specification of operations can be made (the little descriptive phrases that go in the boxes). It does not provide any way of disciplining the structures so built up. As noted, the operations themselves are informal. Sometimes, as in some of the diagrams in Sternberg (1970), the boxes appear so elementary as to be well-defined (e.g., a comparator, a match register, etc.), but in fact the flow diagram still remains informal.

More important from the present view, there is no discipline on the control structure. There are neither primitives of control, nor ways of determining that additional apparatus or processing must occur to effect control. The effect of this is to make the flow diagram unique to each task. It must of course be unique in some way since the tasks are different. But there is then no way to assert when two different flow diagrams represent the same processing mechanism.

The production system, on the other hand, provides a complete set of primitives and determines what auxiliary control processing is necessary to perform a task. This comparison between tasks is possible. This is not a peculiar property of production systems, of course, but is true of any programming system. Writing programs in SNOBOL or FORTRAN would do as well, methodologically, except that their underlying structure does not mirror reasonable psychological assumptions about the human system architecture.

The virtue of the flow diagram is that it expresses clearly the independence and ordering of stages derived experimentally by careful design (e.g., Sternberg, 1969). Flow diagrams, by their very incompleteness, do not

---

[18]Besides flow diagrams, which show control flow, block diagrams, which show data flow, are also used. The remarks of this section apply equally well to both.

over-commit their user to more than what the data say. Thus they are good for summarizing experimental data, at the same time that they are weak for constructing theory.

*The Problem of Methods*

Variability over subjects comes in large part from the variation in the methods (strategy, program, ...) they use for a task. This is conjectural, of course, but much evidence supports it. A major contribution of a detailed theory of control is to make possible the proper posing of the question of what method a subject used for a given task. It does this by providing the space of all methods (based on the constants of system architecture and the primitive operations) for a subject. Thus, the problem of discovering the method takes the form of a programming problem. As we illustrated, there are often many solutions, i.e., many production systems that perform the task, but these can be generated and analysed, and scientific reasons found for selecting one over another within the limited set. This is a quite different situation than currently, where anything seems possible in discussing what might go in a subject's performance.

This formulation of the problem of methods comes not just from the use of a precise language (e.g., a simulation language). It comes from the identification of the space of all programs defined by the system with the space of all programs feasible for the subject.

A theory of control is more important to analyzing methods than just another aspect of the total system necessary to complete specification. Much of what goes on in information processing is control. Almost every operation in a large complex program does nothing except arrange things so something else can do something. This appears to hold for both humans and computers. For instance, Dansereau (1969) found it to be true of humans doing mental multiplication (e.g., 36 x 152). The times for the additions and multiplications--the productive part of the process, so to speak--played a small role compared to the times for fixation, operand positioning, etc. The same is certainly true of the theory as developed in this paper. The decoding hypothesis is in fact a form of the same magicians trick, in which the actions that take time are not the apparently productive part (the iterated test for identity), but a preparatory piece of housekeeping. In short, methods are mostly control, so that any theory of methods must operate within an explicit theory of control.

## *The Problem of Scope*

How to construct theories that range over a wide diversity of tasks is a major issue for psychology. To do so would seem to require a theory that was specific about those aspects of structure and content that in fact were used in common in diverse tasks. A detailed theory of the control structure would seem to offer this, since it specifies the common architecture and the boundaries within which a task-specific method can be sought.

The evidence we have presented that production systems will indeed make a major contribution to this issue is still meager. In this paper we applied the theory only to a couple of tasks. The original production system was applied to a puzzle, a much vaster task than any discussed here, and there are some other applications in Newell (1972). The PSG production system by Klahr (Chapter 11) in this volume provides one more example.

All these efforts provide evidence only about half the issue. They show that it is relatively easy to construct a theory in a new task environment that is responsive to the empirical issues in that environment. One obtains, as well, strong comparability. For instance, Klahr's counting production system can be examined in conjunction with the Sternberg one here. In an important sense they are the same system, since they both use PSG and therefore make the same assumptions about underlying structure. However, the constants of the time model differ. Klahr also uses replacement operators--(X ==> Y) replaces the symbol X in an element with the symbol Y--whereas the model here uses only the encode operator, (**). This leads to a quite different style of programming. Some of his conditions are very long and raise questions about whether constraints should exist on the size or complexity of conditions.

This collection of production systems does not constitute a coherent theory for the set of tasks involved. To do so, they must be melded together into a single production system that performs all the tasks, corresponding to the total organization of a single human. Such a production system will have productions that are unique to each task. But it must face scrutiny about using disparate mechanisms for common operations. It must also handle the instructional problem, since something in the environment must select out the performance relevant to the task at hand. The interaction of the instructions with the task performance program is as much central to control as the internal part of the performance program. It is predictable that a full fledged theory of task instruction will be required.

I stress the creation of a single production system to represent the unified performance on a set of tasks. This seems to me the only way to validate a theory of control. We saw in the discussion of the basic Sternberg paradigm that many degrees of freedom were available, though they showed up as alternatives in method, rather than freedom of parameter settings. This arises primarily because the datum taken from a single trial is so small (i.e., overall reaction time) compared to the complexity of the system that generates it. To compensate, behavior in many disparate tasks must be obtained, so that finally the mechanisms and methods being used become uniquely identified. My own personal estimate is that a model of the control structure should claim to handle some dozens of diverse experiments before it is a genuine contender. The present theory, though promising, still has a ways to go.

It should be noted in passing that the theory refers to individual performance with a specific method. Thus all forms of aggregation raise the spectre of averaging over disparate methods, hence producing mixed estimates. Thus one is driven towards collecting and reporting data only on individual subjects, and even there not averaging disparate performances.

## *The Prospects for this Particular Theory*

As noted, the present theory is only nascent. A few words might be said about its prospects. Missing from the model as it stands is a theory of error. The theory makes only time predictions. Errors are indeed possible in the system, due to incorrect programs and to limited STM. Both of these sources are important in some task environments. Neither of them appears to provide the errors that occur, say, in a Sternberg paradigm. The current theory has implicit in it a model of error, but whether it will work out is not yet clear. It is worth stating because it transforms the theory in an interesting way.

Take STM as having indefinite length but being sufficiently unreliable so that there is an increasing probability of an element disappearing entirely. Whether this is decay with time, with activity or what not is secondary. The fate of each element is somewhat independent so that early ones can disappear before later ones. This is the primary error source, from which error propagates to all tasks according to the strategy with which the subject operates. Such a strengthening of the unreliability assumption will reinforce the encoding hypothesis, so that all tasks must be dealt with by encoding. The role of STM becomes one of holding a few items after decoding (dumping into STM) to be picked up quickly by coupled productions, and of holding a few items strung out prior to encoding into a new chunk. Thus the short term capacity is not the length (or expected length) of STM, but is composed from the size of codes and the space for their decoding. For example, a short term capacity of seven might occur via a chunk of three and four, with the STM holding four items reliably enough to get them decoded and emitted. Thus, no memory structure exists in the system that has a capacity of seven. In particular the STM would appear to be misnamed.

As we have already mentioned, the theory is missing perceptual mechanisms, effector mechanisms and a good theory of LTM acquisition. All of these are serious. The question of how to acquire new productions seems to me the most serious of all. In part this is because we know it to be a hard problem, whereas the others appear to be simply aspects that have not received their share of attention.

All existing theory is delightfully vague on the mechanism of LTM acquisition. It is tied somehow to amount of residence in STM, measured either by time or by rehearsals. But what is stored is left unspecified. Proposing to create a new production makes clear that decisions (by the system) must be made about both conditions and actions. The condition is essentially the access path. The action is essentially the content, though it consists of both passive content (elements to STM) and active content (operators). Since there is good, though indirect, evidence that humans do not have voluntary control of the acquisition process (i.e., operators for constructing productions, which can be part of actions), there must be some more automatic process for learning. Its structure is a puzzle.

The fate of the decoding hypothesis is extremely uncertain. The appeal of an indirect non-obvious explanation of a major regularity in behavior must be resisted. There are an immense number of studies whose interpretation seem straightforward in terms of linear search. Until the decoding hypothesis is shown to be compatible with many more of these than the present paper has considered, the hypothesis should be taken as a strictly secondary challenger. However, the emphasis that it gives to the processes of coding and decoding seems certainly on the right track.

## VISUAL INFORMATION PROCESSING

Bell, C. G., & Newell, A. *Computer structures: Readings and examples.* New York: McGraw Hill, 1971.

Cavanagh, J. P., & Chase, W. G. The equivalence of target and nontarget processing in visual search. *Perception & Psychophysics,* 1971, 9, 493-495.

Crowder, R. G., & Morton, J. Precategorical acoustic storage (PAS). *Perception & Psychophysics,* 1969, 5, 365-373.

Dansereau, D. An information processing model of mental multiplication. Unpublished doctoral dissertation, Carnegie-Mellon University, 1969.

Johnson, N. F. The role of chunking and organization in the process of recall. In G. H. Bower (Ed.), *The psychology of learning and motivation,* Vol. 4. New York: Academic Press, 1970.

McLean, R. S., & Gregg, L. W. Effects of induced chunking on temporal aspects of serial recitation. *Journal of Experimental Psychology,* 1967, 74, 455-459.

Miller, G. A. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review,* 1956, 63, 81-97.

Newell, A. A theoretical exploration of mechanisms for coding the stimulus. In A. W. Melton & E. Martin (Eds.), *Coding processes in human memory.* Washington D. C.: Winston, 1972.

Newell A., McCracken, D., Robertson, G., & Freeman, P. The kernel approach to building software systems. *Computer Science Research Review,* Carnegie-Mellon University, 1971.

Newell, A., & Simon, H. A. *Human problem solving.* Englewood Cliffs, N. J.: Prentice-Hall, 1972.

Sperling, G. The information available in brief visual presentations. *Psychological Monographs,* 1960, 74, Whole No. 11.

Sternberg, S. The discovery of processing stages: Extensions of Donders' method. *Acta Psychologica,* 1969, 30, 276-315.

Sternberg, S. Mental scanning: mental processes revealed by reaction time experiments. In J. S. Antrobus (Ed.), *Cognition and affect.* Boston: Little Brown, 1970.

Sternberg, S., & Scarborough, D. L. Parallel testing of stimuli in visual search. Visual information processing and control of motor activity, *Proceedings of the international symposium,* Bulgarian Academy of Sciences, Sofia, July 1969.

Waugh, N. C, & Norman, D. A. Primary memory. *Psychological Review,* 1965, 72, 89-104.

## Acknowledgments

This research was supported in part by Public Health Service Research Grant MH-07722 from the National Institute of Mental Health and in part by the Advanced Research Projects Agency of the Office of the Secretary of Defense (F4620-70-C-0107) which is monitored by the Air Force Office of Scientific Research.