CIP Working Paper #211

PRELIMINARY RESULTS WITH A SYSTEM FOR AUTOMATIC PROTOCOL ANALYSIS

D. A. Waterman and A. Newell

May, 1972

Departments of Psychology and Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania

ABSTRACT

A computer program for automatic protocol analysis (PAS-I) is described in detail. The task of protocol analysis is that of inferring from the verbalizations given by a human while solving a problem the time course of his states of knowledge about the task. PAS-I works only with the task domain of cryptarithmetic puzzles and incorporates a specific theory of problem solving (as described in Newell and Simon, Human Problem Solving). The input to PAS-I is the transcription of the human's verbalizations (as tape recorded), segmented into topics. The program does a linguistic analysis from the input text to produce a sequence of semantic elements; these are then subjected to several stages of processing, including hypothesizing the information processing operations that the subject performed to produce the semantic knowledge that he appears to have at each moment of time. The final output is a problem behavior graph (PBG) which is the trajectory of the subject through the problem space of possible knowledge states. The performance of PAS-I on three examples of behavior is presented and analysed: 100 topic segments of subject S1 on the cryptarithmetic task, DONALD+GERALD=ROBERT; 43 further topic segments on S1 on the same task; and 128 topic segments (the entire session) of subject S4 on DONALD+GERALD.

PRELIMINARY RESULTS WITH A SYSTEM FOR AUTOMATIC PROTOCOL ANALYSIS*

D. A. Waterman and A. Newell

May, 1972

The study of the information processing that occurs in human problem solving often involves a form of data analysis called protocol analysis.** The subject is instructed to talk during a problem solving session, uttering whatever is concerning him at each moment. His verbal behavior is recorded on audio tape, together with appropriate indicators of relevant non-verbal behavior, and a transcript is made which becomes the primary source data. Inferences, based on the content of the utterances, are then made about the information processes which must have occurred. When the information processing of the subject is used as a basis for proposing a formal model (usually a program), the adequacy of the model can be assessed by comparing the trace of the behavior of the model with the protocol and the information inferred from it.

This use of the term protocol is within the general use of the term in psychology, where it designates the time course of an experimental session, usually described qualitatively. But the emphasis on verbal behavior and on

* 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.

** There have been many studies, not even counting the early ones such as Dunker's (1945). A sample: Bartlett, 1958; Baylor, 1971; Bree, 1968; DeGroot, 1965; Eastman, 1969; Feldman, 1963; Frijda and Meertens, 1969; Johnson, 1964; Laughery and Gregg, 1962; Newell, 1968; Newell and Simon, 1961, 1963, 1965, 1972; Paige and Simon, 1968; Reitman, 1965; Wagner and Scurrah, 1971 and Winikoff, 1968.

the analysis of content is somewhat idiosyncratic to the use of the term in cognitive psychology. Protocol analysis has not been formalized, and the question of its essential nature and the extent of its biases and inadequacies have been discussed only to a limited extent (e.g., DeGroot, 1965). Intertwined in this discussion has been the issue of the relation of protocol analysis to the so-called introspective method, used extensively in an earlier era of psychology and linked most strongly to the Wurzburg school (Humphrey, 1951).

In this paper we describe a system (called PAS-I) for the automatic analysis of protocols and give initial results of our work with this system. The analysis we use is patterned after a particular style of protocol analysis developed for the task of cryptarithmetic, and to a lesser extent for symbolic logic and chess (Newell and Simon, 1972). PAS-I is limited to working within a single task domain, cryptarithmetic. A cryptarithmetic problem is a type of puzzle, simple enough to be solved in a short period of time, yet complex enough to require interesting problem solving activity to effect its solution.

Throughout this paper we refer to a system for automatic protocol analysis, although we realize the system will eventually involve extensive man-computer interaction. We deem it inappropriate to view our problem primarily as one of creating a man-machine system (Waterman and Newell, 1971). We prefer to approach the task as one of building a system for fully automatic analysis, realizing that the end product will be an extremely sophisticated tool for the scientist to use.

Reasons for Automation

There exist several quite distinct reasons for automating protocol analysis, of which the following three seem the most basic.

Efficiency. A large amount of effort is involved in making an analysis by hand. This has amounted to hundreds of prime scientist hours for tens of minutes of behavior. The upshot is that relatively few analyses of any detail yet exist. Automatization would make it possible to produce many analyses, including numerous variations, and this appears to be an essential condition for protocol analysis to become a generally useful tool.

Objectivity. Automatic analysis may help attain objectivity of inference. Currently, a human makes the transcription from tape, utilizing an unidentified amount of intelligence -- that involved in understanding natural language. Then, in making each inference from the content of the transcribed utterance further inferential processes are used, their extent and depth depending on the intelligence and care of the analyst. Thus, the auxiliary information that goes into the analysis is essentially uncontrolled.

A basic strategy of science to cope with complex systems is to create measuring instruments that map system behavior into highly constrained and simple spaces (usually numerical). Psychology, in attempting this strategy, has tended to use human beings as instruments, e.g., the use of panels of judges to quantify concepts such as hostility, tension, cohension, neuroticism, etc. In all of these attempts, including current protocol analysis, the instrument-cum-human remains an unanalyzable system, hence a fundamental source of error. The tendency in psychology has been to accept this state of affairs and to focus on the reliability of the human instrument by replication. Automatization might offer an alternative route to objectification, by building mechanical (hence repeatable and determinable) schemes for going directly from the uncontrolled environment to final summary judgments of scientific importance.

Representation of support. Another reason for automating protocol analysis is related to objectivity, but distinct from it. The nature of the support that a protocol offers to a proposed model (or hypothesis) is given by a complex web of arguments, which simultaneously demonstrate the adequacy of the model and the inadequacy of various plausible alternatives. Currently, there exists no way of representing the total state of support (or non-support). Working with protocols is somewhat analogous to a trial at law in which various arguments are brought forth sequentially, without ever confronting the whole. Automatization, by forcing formalization, may make possible a representation of the total evidence marshalled by a protocol in support of a given model or hypothesis.

It is premature to worry about the extent to which these three goals can be achieved, or are in process of being achieved, by the results to be given here. Enumerating them serves, rather, to make explicit the context in which the present effort is being conducted.

Area of Application

There are at least three vantage points from which the present work can be viewed. The first, explicated above, is the attempt to formalize protocol analysis in the service of better cognitive psychology. The second is artificial intelligence. Automating protocol analysis requires capturing in mechanism heretofore exclusively human intellectual functions. The programs that accomplish this will be efforts in artificial intelligence, presumably as interesting as chess playing and theorem proving programs. The third vantage point is the study of language. A program for automating protocol analysis must, perforce, extract information from natural language. In linguistics, although something approaching adequate analysis currently exists for phonemic and syntactic structure, the same can hardly be said for the full arc from utterance to understanding. Thus, a program for understanding the information conveyed in a protocol, may be of linguistic interest.

The present paper is addressed exclusively to the first concern -- the psychological one. A companion paper (Waterman and Newell, 1971) addresses itself to the question of artificial intelligence,* though in equally preliminary form. The work is not yet so far advanced nor so firm in its approach that it is time for any linguistic observations.

The paper is organized as follows. Section II describes in outline form the theory of human problem solving that operates as a theoretical substructure to the system of analysis. Not surprisingly, the power of the analysis stems primarily from this underlying substantive theory. Section III briefly describes the total scope of protocol analysis, delineating the parts of the analysis we are currently attempting to handle automatically.

* The companion paper contains references to related artificial intelligence programs.

Section IV gives a detailed description of the PAS-I analysis program. Section V presents some preliminary results of the program just described and discusses the adequacy of a number of its aspects. The concluding section, VI, gives an indication of how the system can be extended.

II. THEORETICAL SUBSTRUCTURE

The theory, in brief outline here, is expounded at length in Newell and Simon (1972); earlier versions also exist (Newell, 1967, 1968; Newell and Simon, 1965). We are only concerned with the theory as it applies to the class of puzzles called cryptarithmetic problems (Brooke, 1963). A typical task with instructions is shown in Figure 1. It was given to Subject S3, whose protocol will form the basis for most of our illustrations and whose analysis has been accomplished manually (Newell, 1967; Newell and Simon, 1972). The task itself is symbolic in nature, drawing on the subject's past knowledge of arithmetic and elementary algebra; solution time varies from five minutes to several times that long,for people with college educations and some quantitative aptitude.

Structure of Problem Solving

Knowledge states. We assume human problem solving takes place by search in a problem space. The elements of this space are the possible states of knowledge the subject can have about the task. This space can be generated in a grammar-like way, so that a state of knowledge can be viewed as an expression in some language of what the subject knows when he is at a particular point in the space. This language is idiosyncratic to the subject and to his general state of learning and experience up to the point of the problem solving session. It reflects the basic relations and properties the subject can distinguish about the task.

$$\begin{array}{r} \text{DONALD} \\ +\ \text{GERALD} \\ \hline \text{ROBERT} \end{array} \qquad D = 5$$

The above expression is a simple arithmetic sum in disguise. Each letter represents a digit, that is, 0, 1, 2, ..., 9. Each letter is a distinct digit. You are given that D represents the digit 5; thus, no other letter may be 5.

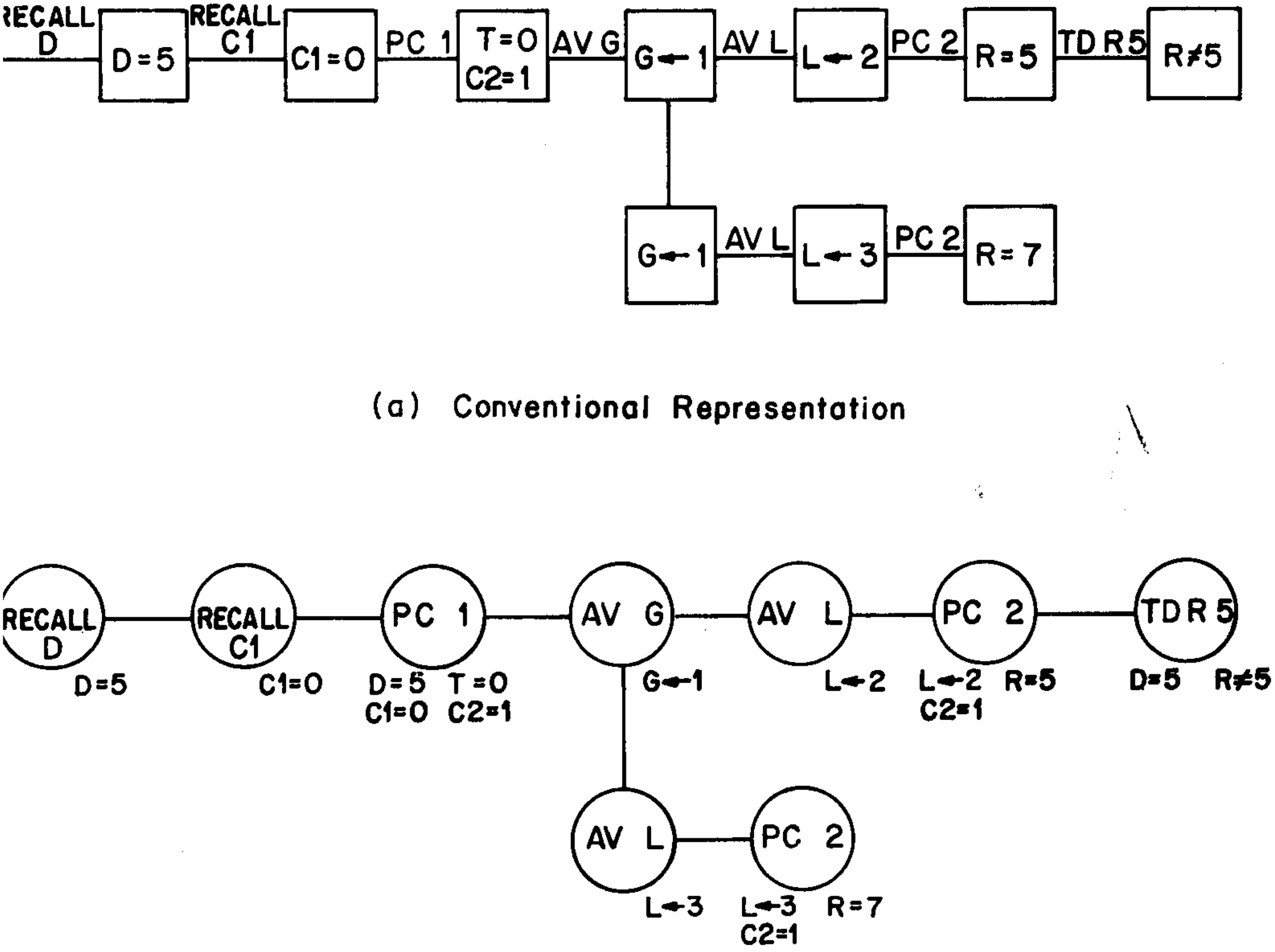
What digits should be assigned to the letters such that when the letters are replaced by their corresponding digits the above expression is a true arithmetic sum?

Please talk all the time while you work, saying whatever is on your mind at each moment, however fragmentary, trivial, apparently irrelevant, impolitic, or indiscreet. Whenever you fall silent for more than a moment the experimenter will ask you to "please talk."

Figure 1. Instructions for Cryptarithmetic Task

Operators. Besides states of knowledge, the problem space also includes a set of operators. These define operations the subject can perform on knowledge at a particular state to yield new knowledge -- hence to move to a new knowledge state. Operators need not be applicable in every state of knowledge, since an operator could require types of information not currently available. A subject will have a small finite set of these operators, which constitutes his sole means for making headway in solving the problem. Like the states of knowledge, the operators used during a particular problem solving session are idiosyncratic to the subject and his current state of learning and experience.

Problem Behavior Graph. The behavior of the subject over time can be described as a sequence of operator applications that create a string of incrementally changing states of knowledge. In short, his behavior is describable as search through the problem space. This search will terminate when the subject arrives at a state of knowledge that includes the solution (and the knowledge that it is the solution). The plot of this search is called the Problem Behavior Graph (PBG). Two representations of a fragment of a PBG for the cryptarithmetic problem DONALD+GERALD=ROBERT are shown in Figure 2, along with brief definitions of the operators and elements of the problem space used in the illustration. The conventional representation, where boxes represent knowledge states and horizontal lines the operators applied to produce new knowledge states, is given as example (a) of Figure 2. The group representation, used by the system described in this paper, is given as example (b) of Figure 2. Here a circle represents the application of an operator, and the knowledge elements immediately preceding and following the circle represent the inputs and outputs of the operator. Note that the knowledge state at any point in the graph is the conjunction of all output elements on the path from the given point back to the beginning of the graph.



(a) Conventional Representation

(b) Group Representation

KEY:	*Knowledge*		*Operators*	
	=	equal	RECALL	recall element
	←	assign equal	PC	process column
	≠	not equal	AV	assign value
			TD	test digit

FIGURE 2. Two Representations of a Problem Behavior Graph (PBG)

The problem behavior graph for a specific problem solving occasion asserts two empirical propositions, namely, (1) the subject's knowledge of the task can be restricted to the limited basis given by the elements of the problem space, and (2) all changes in knowledge can be described as the operation of a fixed set of information transformations.

Not all the subject's knowledge about the problem is expressed directly in the problem space. The subject has much knowledge that is constant throughout the experimental session, and does not show up explicitly in the definition of the space, e.g., basic knowledge of arithmetic. He may have information that is generated within a node of the space, but is entirely local to that node, e.g., various temporary states of attention. The subject also has information about the course of his search through the problem space -- so-called path information -- which allows him to avoid repeating the same path twice and to return to a prior knowledge state to begin search in a new direction.

Production system. The problem behavior graph makes relatively weak statements about the subject's behavior, since it simply posits the operators that occur in the search and provides no explanation of why a particular operator occurred at a particular time. This additional specification is given by a production system, which is a collection of rules, each of the form

condition → action .

This is read: if the condition is true of the current knowledge state then perform the action. The action is a sequence of one or more operators of the problem space. Without loss of generality, the action could always be exactly one operator, in which case just one production would be needed to specify the operator of any node of the PBG.

The productions are given in the order of priority in which they are to be evoked. Thus, the first condition in the list, starting from the top, that is true of the current knowledge state leads to the evocation of the corresponding action. The processes is then repeated with the (now) modified knowledge state. It will be seen that this forms a complete system for specifying behavior over time.

To be concrete, consider a production system with two rules:

(1) PC fail → FL(*COL), AV(*L)

and (2) (*L = *D) → FC(*L), PC(*COL) .

The symbols preceded by an asterisk (*) are variables: *L for letters, *D for digits and *COL for columns. Now consider the first knowledge element in the PBG of Figure 2, (D = 5). This element is an instance of (*L = *D), and thus invokes production (2) above. This in turn invokes the operator FC (which, like FL, is an attention directing operator and is not shown in the PBG of Figure 2). FC finds column 1, and then PC is evoked on *COL, which now has this value, and produces the result shown in Figure 2, namely (T = 0) and (C2 = 1). Now the production system repeats itself. This time (2) is again satisfied (still by D = 5) and FC finds column 6. However, when PC is applied it fails to produce new information. Thus, on the next cycle (1) is evoked, which leads to FL selecting a letter from the current column (say G) and AV assigning it a value (G ← 1). Thus, we see how a system of productions can lead to a continuing sequence of operator evocations that select at each node of the PBG the operator to be applied.

Sources of Problem Solving

Both the problem space and the production system are given, thus providing a description of the behavior of the subject from which other properties follow. In a qualitative way it is clear where problem spaces come from. The basic capabilities for discrimination and operation must be available in the subject's repertoire prior to the experimental session. The actual space can be already available (e.g., as in chess, where the subject has already played many chess games), or it can be constructed at the initiation of the experimental session from the perceived characteristics of the situation plus the instructions. Empirically, the problem spaces used by subjects for cryptarithmetic bear a close relation to the way the task is described in the instructions.

The operators of the problem space correspond to actions or behaviors the subject can evoke reliably to obtain relevant ends. They do not necessarily correspond to elementary behaviors (e.g., to a single reaction), but may be extended sequences of contingent actions. The important criterion is that an operator provides, from the subject's viewpoint, a reliable process that produces a result with known properties. In essence, it is a subroutine.

The source of the individual productions is less clear. The basic question is whether the production system is really the appropriate representation of the subject's information processing system. Production systems have exactly the same logical scope as any other general programming language, thus any set of contingent behaviors describable by production systems can also be described by any general programming langague. Still, each production appears to represent an act of learning and to generate a bit of locally adaptive behavior which is meaningful even in isolation.

The productions take the current knowledge state as input, without discriminating between the various types of memory systems available. It appears to be possible to distinguish an immediate working memory or short term memory (STM) from a long term memory (LTM), from an external memory (EM). It does not appear possible in the sort of subject controlled problem solving covered by the theory to distinguish the various image stores (e.g., visual or auditory) or the modality of various memories. The theory is sensitive to the important operational specifications of these various memories (such as their capacities, read and write times, and accessing schemes), and provides a partial explanation of various features of the problem space and production system in terms of these specifications (Newell and Simon, 1972, Chapter 14).

III. THE SCOPE OF PROTOCOL ANALYSIS

Figure 3 gives a representational scheme that serves as the basis for our current attempts to develop automatic techniques for protocol analysis. The figure proposes a set of representations that are constructed on the basis of the audio tape to produce the total psychological model. The representations are compatible with the manual analysis used for S3 on DONALD+GERALD=ROBERT (Newell and Simon, 1972).

Basic Representations

From an experimental session an _audio tape_ is produced, which contains the speech utterances, along with additional signals that encode experimental events (clock signals, etc.). This audio tape constitutes the primary data to be analyzed.

Linguistic representations. The first intermediate representation, the _linguistic_ one, can be subcategorized into two parts. The first, the _lexical_ representation, consists of a sequence of words, including any additional notations for experimental events, prosodic features, and paralinguistic features. This continuous stream of words is segmented* into _topic segments_, which are short phrases or fragments containing only a single task topic. This refined lexical representation is called the _topic_ representation and performs primarily an attention directing role by providing a reference scheme for designating various parts of the verbal stream.

Semantic representation. The next representation, the _semantic_ one, consists of a set of semantic elements which can be arranged in a time ordered sequence. There are two main classes of elements: _problem space_

* The segmentation process is actually a relatively simple form of parsing.

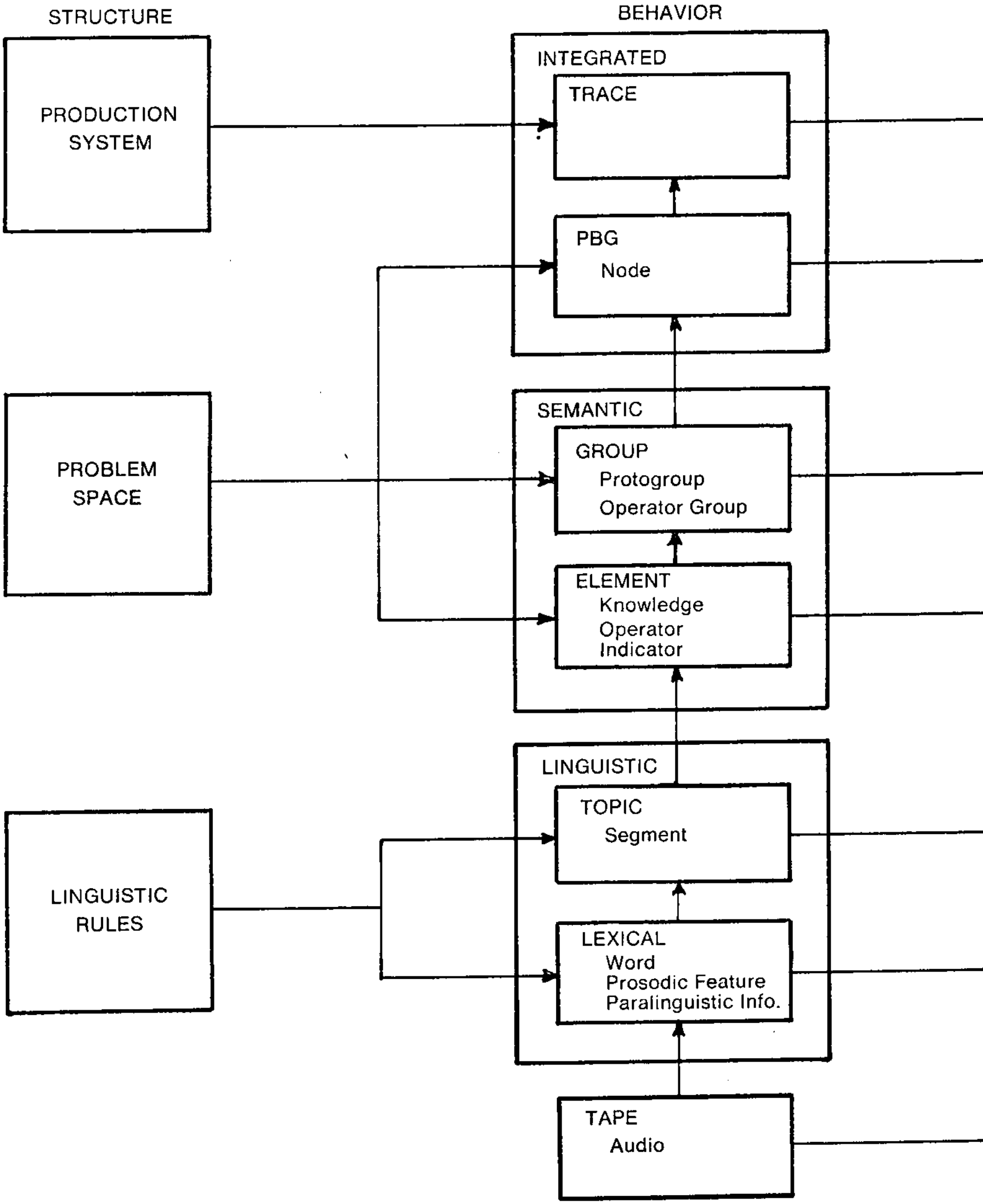


Figure 3. Representations for Protocol Analysis

elements and indicator elements. The problem space elements are subdivided into knowledge elements, which represent the knowledge the subject has about the problem, and operator elements, which represent the action he takes to produce new knowledge. The indicator elements describe the relations between one or more problem space elements.

The semantic elements can be arranged into functional units or groups. An operator group consists of an operator together with its input and output knowledge elements. The protogroup is simply an initial hypothesis of the components of an operator group.

Integrated Behavior Representations. The first form of integrated representation is the Problem Behavior Graph (PBG). It is a tree structure (recall Figure 2a) in which the nodes represent statesof knowledge at a given point in time and the branches represent the application of an operator at that time to change the knowledge state. If we orient the tree as shown in Figure 2, then time runs across the page and down. The tree structure arises because subjects abandon information (due to discovered error, irrelevance, or forgetting) and return to prior states of knowledge.

The knowledge state at a node is given by a list of all the knowledge elements true at that time. Associated with each node are the knowledge elements created by the operator element just applied. Then the total knowledge state is obtained by taking the conjunction of all knowledge elements from the given node, working back to the first node of the tree. This implies that the operator elements are incremental in nature, each operating on only a small subset of the total knowledge present. It also admits the possibility that contradictory, or otherwise variant, information can exist in the knowledge state.

Both the semantic elements and the PBG depend on the existence of the problem space, which defines the allowable types of knowledge and operator elements. The problem space is not a behavior representation, but rather a static representation of the structure of the subject's knowledge. We show it in Figure 3 to the left of the sequence of behavior representations.

The second integrated representation, the trace, relies on the production system, which is another static representation of the structure of the subject's information processing system. The production system generates the trace, which is a sequence of the information in various memories (especially STM and EM, since LTM changes slowly) at each instant together with the productions evoked by this information. The trace is a linear sequence in time, since the returns to prior knowledge states that give the PBG its tree-like shape show up simply as additional instances of transformations.

Assessment representations. The PBG plus the production trace provides a detailed description of the integrated behavior of the problem solving system representing the subject. The final step is to measure the adequacy of the model. This consists of assessing the amount of behavior explained by the model (the PBG at one level, the production system at the next), and the number and types of errors encountered.

Generally, one thinks of the derivation of the representation in Figure 3 as follows. The audio tape yields a lexical representation which is segmented, and then processed to yield semantic elements. These elements are grouped and incorporated into the PBG, which is then analysed to yield the production system and its trace. But the derivation need not proceed this way entirely, since information from all parts of the analysis may be needed to take a particular inferential step at any level. For instance, to resolve the referent to the word, IT, one may on one occasion need to know the prior sentence, but on another occasion need to know that the subject is working on column 4 (which may not have been mentioned directly in recent sentences), and on yet another occasion need to know that the only possible value of IT is the letter D because it can be concluded that IT is 5 and that the subject's knowledge state contains D = 5. Thus the strategy of analysis is a relatively fluid aspect of the total system. In fact, a generally valid principle is that all levels must be brought forward simultaneously, so that each has available a maximal amount of contextual information.

Data Analysis Tasks

The system of representations exhibited in Figure 3 indicates the total scope of protocol analysis as we now see it. Various scientific tasks exist when information is available about some of the representations in the figure and it is desired to obtain information about the other representations. To each pattern of given information there corresponds a characteristic task of data anlysis. If the structure representations are given and also the behavior data at the bottom, then the task is that of behavior description. If the behavior representations are given, but not the structural representations, then the tasks are those of induction. If the structure is given and also the behavior at the top, then the task is that of predicting the behavior at more detailed levels. When all representations are given, the task is that of verification. For all such tasks, there arises the further task of determining and representing the success of the analysis made. All these tasks, though distinct, are closely related and together are the essence of what we call protocol analysis.*

These data analysis tasks vary widely in difficulty and types of analysis required. Since all parts of the analysis are interdependent, we operate in a highly experimental mode, continually defining and redefining partial systems to account for an everwidening range of information and tasks. We do not view any particular component as more than a tentative hypothesis, which may or may not survive subsequent iterations. In this way we expect to evolve a system for automated analysis.

The present scheme tackles only some of the tasks implicit in Figure 3. It takes as given the topic representation and the problem space, and attempts to create the semantic elements and the PBG. The difficult problem of going from the audio tape to the lexical level is put to one side, as is the problem of the induction of the problem space or the production system. Two central problems are retained: (1) extracting task relevant content from the linguistic representation; and (2) inferring from the extracted content the relevant knowledge state of the subject at each point in time. We refer to this as the task of describing behavior according to a given theoretical scheme.

Starting with the segmented text (the topic representation), rather than the stream of ungrouped words (the lexical representation) is an appropriate restriction. Currently available initial representations (transcriptions prepared in the usual way by listening to the audio tape) contain only lexical data and not prosodic data. The latter is required to pose the task of topic segmentation in a reasonable form.

* A more careful statement of these various analysis tasks can be found in the companion paper (Waterman and Newell, 1971).

IV. DETAILED STRUCTURE OF PAS-I

The data analysis strategy used in implementing Protocol Analysis System I (PAS-I)* is shown by the flow diagram in Figure 4. As seen from the figure, the system contains four major components: the Linguistic Processor, the Semantic Processor, the Group Processor, and the PBG Generator.

Linguistic Processor

The Linguistic Processor operates on a single topic segment at a time, producing for each segment one or more semantic elements. These elements, taken together, represent the meaning of the segment. This linguistic process operates in isolation from other knowledge in the system, using only the information in the single input segment. As a result only simple inferences can be made; thus many of the elements produced contain ambiguous or incomplete information.

Semantic elements. The semantic elements used in the system are given in Table 1, together with their intended interpretation. They constitute a particular definition of a problem space, essentially that used in the manual analysis of S3. A subset of these elements, marked with asterisks, require more contextual information than do the others and consequently are generated at a later stage of the analysis by the Semantic Processor.

Linguistic analysis. We use a simple scheme of linguistic analysis, based on a key-word grammar. This analysis consists of searching for (generally)

* The system is running on the PDP-10 at CMU. The linguistic processor is coded in SNOBOL4 and the rest of the system in LISP 1.5, with communication existing between the two parts via the disc. It takes roughly 12 seconds per topic segment for the analysis: about 3 seconds for linguistic processing and 9 seconds for the remainder.

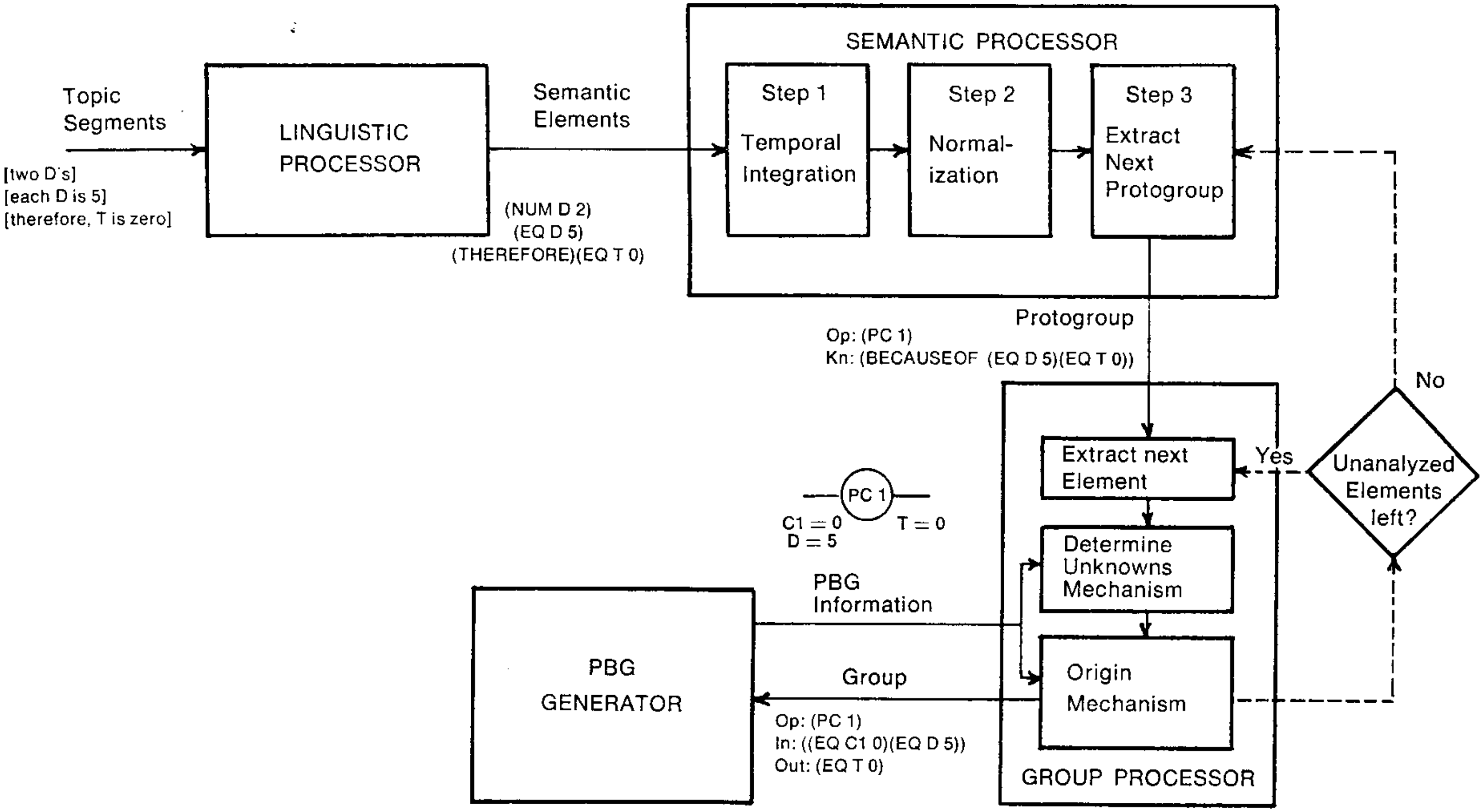


Figure 4. Flow Diagram of PAS-I

SEMANTIC ELEMENTS

KNOWLEDGE	MEANING
(LETTER l)	An occurrence of the letter l
(DIGIT d)	An occurrence of the digit d
(PLS u)	u is added to something
(IN v d)	v is in column d
(EVEN v)	v is even
(ODD v)	v is odd
(EQ v d)	v equals d
(PEQ v d)	One possible value for v is d
(GREATER v d)	v is greater than d
(SMALLER v d)	v is smaller than d
(CIN d col)	The carry into column col is d
(COUT d col)	The carry out of column col is d
(MEQ v d_1 d_2 ...)*	v must equal either d_1, d_2, or ...
(NEQ v d)*	v is not equal to d
(AEQ v d)*	v is assumed to have the value d
(COND e_1 e_2)*	If e_1 is true then e_2 is true
(BECAUSEOF p_1 p_2)**	p_2 is true because p_1 is true
(CPIO p_1 p_2 p_3)**	p_1 is an operator with inputs p_2 and output p_3

OPERATORS	MEANING
(FC v)	Find a column containing v
(NUM l d)	the number of l's is d
(PLUS u_1 u_2)	u_1 is added to u_2
(EQC (PLUS u_1 u_2)u_3)	u_1 plus u_2 equals u_3
(COUNT l)	Count the number of l's
(RECALL v)*	Recall the value of v
(PC d)*	Process column d
(GN l)*	Generate possible values for l
(IG c)*	Ignore the carry c
(AV v)*	Assign some value to v
(FA e)*	Find the antecedent of element e
(FN e)*	Find the negative of the antecedent of e
(TD v d)*	Test if v can be equal to d
(TE e)*	Test if e can be true

INDICATORS
(OR)
(IF)
(AND)
(YES)
(NEG)
(WNOT)
(QUES)
(THEN)
(BECAUSE)
(UNLESS)
(ASSUME)
(DIFFICULT)
(THEREFORE)
(CORRECTION)
(INSTEADOF)
(PLACE loc)

Key			
l :	letter	e :	knowledge element
d :	digit	p :	list of problem space elements
c :	carry	loc :	location
v :	letter or carry	col :	column indicator, such as (PLUS A A)
u :	letter, carry, or digit		

Table 1. Examples of Semantic Elements Used in PAS-I
(Not all knowledge elements are shown.***)

[*] These elements are generated by the Semantic Processor rather than the Linguistic Processor.

[**] These elements represent intermediate knowledge and thus do not appear in the PBG.

[***] For each knowledge element shown above not tagged with an asterisk there exist two corresponding elements, the negation and the assumption, prefixed with N and A respectively. For example, NGREATER and AGREATER are the corresponding elements for GREATER.

noncontiguous strings of keywords, classifying them semantically, and translating them into the appropriate semantic elements. We do not use contextual or semantic information to resolve ambiguities during the linguistic analysis, except on a very local level within a segment. Instead, we pass these ambiguities, usually in the form of incomplete or ambiguous semantic elements, down the line to be resolved by a later stage of the system that does make use of contextual or semantic information. The rationale is that there already exists in the system a very powerful mechanism* for inferring information totally omitted from the text. Consequently much is gained by applying this mechanism to the somewhat simpler problem of resolving ambiguities in existing semantic elements.

The reasons for using a key-word approach to the linguistic analysis are two-fold. First, the language of the subject is fragmented and not completely grammatical. The task of building an adequate grammar for such utterances is an open problem. Thus, an appropriate grammar is not readily available, although a grammar substantially more sophisticated than the grammar used here could be constructed. The second reason is that the system makes extensive use of semantic analysis. The cryptarithmetic task provides a somewhat restricted universe of discourse, and a strong semantic component exists in the form of the problem solving theory. Consequently, the strong semantic analysis tends to compensate for the realtively weak linguistic analysis used.

* The Origin Mechanism, discussed in detail later.

Grammar. The grammar used by the Linguistic Processor is given in Figure 5 in a modified BNF notation. Linguistic classes are represented by names in angle brackets, and those classes marked with an asterisk(*) represent semantic elements. A class is defined by giving its name at the left, followed by colon-equal (:=) followed by the sequence that defines the class in terms of words or other classes. A vertical bar (|) indicates a disjunction, a number-sign (#) indicates the beginning of a string, while a dash (-) or the absence of a blank between two items of the sequence indicates concatenation.[*] A blank between two items means that any arbitrary string of words may exist between them. Thus if we have the definitions:

```
    <eq> := <letter><prep> <digit> | #CHANGE<letter> <digit>
  <prep> := AS | FOR-THE
<letter> := D | R
 <digit> := 2
```

then D FOR THE 2, USE D AS THE NUMBER 2, CHANGE R 2, and CHANGE R TO 2 are all examples of the class eq, while D IS FOR THE 2, CHANGE THE R TO 2, and SO CHANGE R TO 2 are not examples of this class. The test for a class proceeds from left to right through a string, with the first match defining the values of the class variables. Thus in D AS R FOR THE 2 the test for eq yields D as the letter, AS as the prep and 2 as the digit.

The grammatical analysis proceeds in the order shown in Figure 6. For example, the test for <cin> precedes the test for <cout> and the test for <neg> precedes all other tests. The tests within each box are applied in order until either an instance of a class is found (in which case the remaining classes in the box are skipped) or all the tests have been applied. Note that the tests for <neg>, <digit> and <letter> are applied as many times as necessary

[*] The dash (-) indicates concatenation between two words, while the absence of a blank indicates concatenation between either a word and a word class or two word classes.

```
* <cin>         := <carryeq> INTO <sum> |  INTO <carryeq> <sum> | <carryeq> <sum> INTO
* <cout>        := <carryeq> FROM <sum> |  FROM <carryeq> <sum> | <carryeq> <sum> FROM
  <carryeq>     := <digit> <carry> <digit> | <carry> <digit> | <digit> <carry> | <carry>
* <greater>     := <sum> <large> <optdigit> | <ltr> <large> <optdigit> | <large> <digits>
* <smaller>     := <sum> <small> <optdigit> | <letter> <small> <optdigit> |
                   <pronoun> <small> <optdigit>
* <even>        := <even1> | <even2>
  <even1>       := <sum> <equals> EVEN <optnumber> | <ltr> <equals> EVEN <optnumber>
  <even2>       := <add> <digit> <equal> EVEN <optnumber>
* <odd>         := <odd1> | <odd2>
  <odd1>        := <sum> <equals> ODD <optnumber> | <ltr> <equals> ODD <optnumber>
  <odd2>        := <add> <digit> <equal> ODD <optnumber>
* <eqc>         := <eqc1> | <eqc2>
  <eqc1>        := <sum> <equal> <sum> | <sum> <equal> <optletdig> |
                   <digits> <equals> <sum> | <ltr> <equals> <sum>
  <eqc2>        := <add> <digit> <equal> <digit> <optnumber>
* <peq>         := <ltr> <poseq> <digits> | MIGHT <letter> <equal> <optdigit>
* <eq>          := <carryeq> | <letter><prep> <digit> | <letter> <equal> <optdigit> |
                   <letter> 'S <digits> | <pronoun> <equals> <digit> | <digit><prep><ltr>|
                   <make> <ltr> <optdigit> | <letter><digit> | <letter>UH<digit> |
                    CHANGE<letter> <digit>
* <plus>        := <two><letdig>'S | <letdigs><ad> <letdigs> | <letdig>AND <letdig> |
                   <digit><comma><digit>
* <place>       := <location> <loc1> | IN<location> | COLUMN PRECEEDING |
                   <loc2>HAND | THE<loc2>
* <pls>         := <ad> <letdigs> | AND<letdig>
* <num>         := <digit><letdig>'S
* <in>          :=  HAVE <letdig>
* <fc>          :=  HAVE <letter> ?
* <count>       :=  HOW-MANY <letter>

  <ltr>         := <pronoun> <letter> | <letter> <pronoun> | <letter> | <pronoun>
  <sum>         := <letdigs><ad> <letdigs> | <two><letdig>'S | <letdig>AND <letdig> |
                   <digit><comma><digit>
  <letdigs>     := <letdig> | <pronoun>
  <letdig>      := <letter> | <digit>
  <optletdig>   := <digit> | <letter> | < >
  <optdigit>    := <digit> | < >
  <optnumber>   :=  NUMBER | < >
  <add>         := <ad> | AND
  <equals>      := <equal> | 'S
  <digits>      := <digit> | <pronoun>

* <letter>      := A|B|D|E|G|L|N|O|R|T|LETTER
* <digit>       := <dig>|10|11|12|13|14|15|16|17|18|19|20|
                   ZERO|TWO|THREE|NINE|NUMBER
* <therefore>   := THEREFORE | #SO | IN-THAT-CASE | IN-WHICH-CASE | WHICH-WILL-MEAN |
                   WHICH-WILL-MAKE | WHICH-MEANS | WHICH-LEAVES | WHICH-WOULD-MEAN |
                   IMPLIES | MIGHT-INDICATE | AUTOMATICALLY
```

```
* <if>            := IF | SUPPOSE | S'POSE | AS-SOON-AS | IN-ORDER-TO | THAT'S-ASSUMING
* <assume>        := ASSUME | ASSUMES | ASSUMING | ASSUMPTION | ASSUMED | LET | USING |
                     <self> <make>
* <difficult>     := DIFFICULT | DIFFICULTY | TROUBLE | DILEMMA | MISSING
* <correction>    := INSTEAD | OR-RATHER | #RATHER | I'M-SORRY
* <then>          := THEN | THAT-MEANS | THIS-IMPLIES
* <neg>           := CANNOT | NOT | NO | N'T | BAD-GUESS
* <wnot>          := DEPENDING-ON-WHETHER-OR-NOT
* <yes>           := YES | YEAH | TRUE
* <or>            := #OR | OR<letdig>
* <because>       := BECAUSE | SINCE
* <insteadof>     := INSTEAD-OF
* <unless>        := UNLESS
* <and>           := #AND
* <ques>          := ?

  <pronoun>       := ITSELF | IT | THAT | SOMETHING | THIS | THESE | THOSE | THEY | THEM |
                     NUMBER | LETTER
  <equal>         := IS | EQUAL | EQUALS | BECOMES | BE | WAS | WERE | ARE | BEING |
                     EQUALING | =
  <carry>         := CARRY | CARRIES | CARRIED | CARRYING
  <location>      := OTHER | SIDE | SECOND | FIRST | LEFT | RIGHT | THERE | GERALD |
                     PRECEEDING | FOLLOWING
  <prep>          := FOR | OR | AS | FOR-THE
  <poseq>         := CAN-BE | COULD-BE | MIGHT-BE
  <make>          := MAKE| MAKING | MADE | MAKES
  <large>         := LARGER | GREATER | BIGGER | MORE
  <small>         := SMALLER | LESS
  <loc1>          := SIDE | COLUMN
  <loc2>          := LEFT | RIGHT
  <dig>           := 0|1|2|3|4|5|6|7|8|9
  <ad>            := PLUS | +
  <self>          := I | WE
  <two>           := TWO | 2
  <comma>         := ,
  < >             :=
```

Figure 5. Grammar Used by the Linguistic Processor
(The classes marked with an asterisk represent semantic elements.)

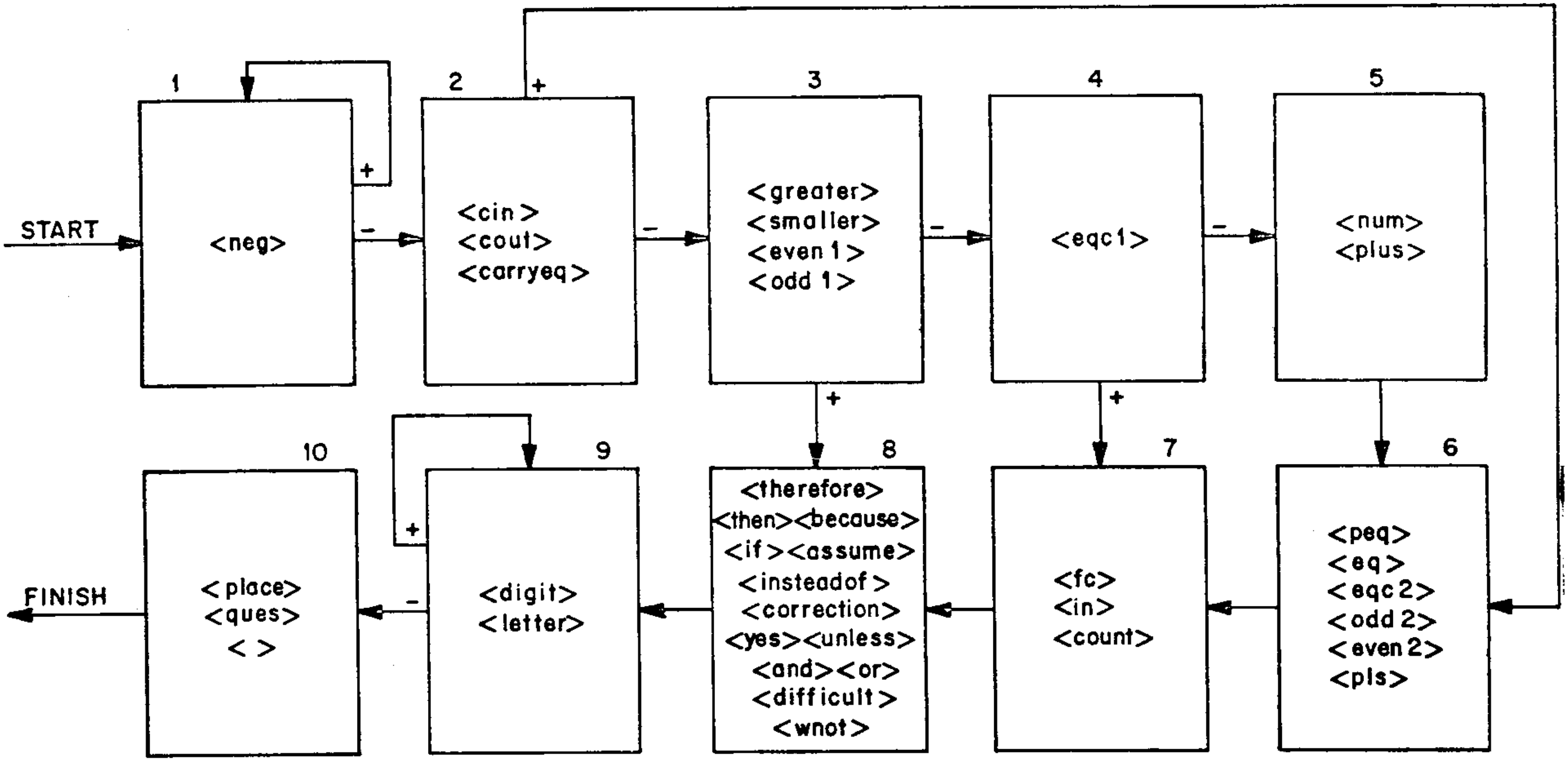


FIGURE 6. Flow Diagram for Linguistic Analysis

to recognize all instances of these classes in the segment.

To illustrate more clearly the control flow shown in Figure 6 consider the topic segment BECAUSE I DO N'T WANT TO BE CARRYING 1 INTO THAT E + O COLUMN. The test for <neg> is applied and matches the N'T, which is then removed from the segment. The test for <neg> is again applied but to the shortened segment. This fails, so control passes to block 2 where the test for <cin> is applied and matches the string CARRYING 1 INTO THAT E + O. This string is also removed from the segment, and control passes to block 6 where the indicated tests are applied to the segment, which is now BECAUSE I DO WANT TO BE COLUMN. These fail, as do all the other tests except <because> in block 8. The result is the set of semantic elements (BECAUSE)(NEG)(CIN 1 (PLUS E O)).

The control flow in Figure 6 reflects our current hypotheses about the number and kinds of different classes which can occur in a single segment. It is closely tied to the definitions of the linguistic classes; consequently, as the grammar is extended and refined, the control flow must be redefined in some corresponding way.

Segment processing. Figure 7 illustrates the operation of the linguistic processor on three segments taken from S3. Example (a) is a simple phrase which leads to a single knowledge element, namely that G is an even number. Example (b) shows a more complex analysis in which several elements are extracted, including indicators. The last example illustrates that more than one problem space element can be extracted from a single segment, although normally a segment contains only one topic. The *C in this example stands for an unknown carry. Note that not all words of a segment participate in the parse in a positive way.

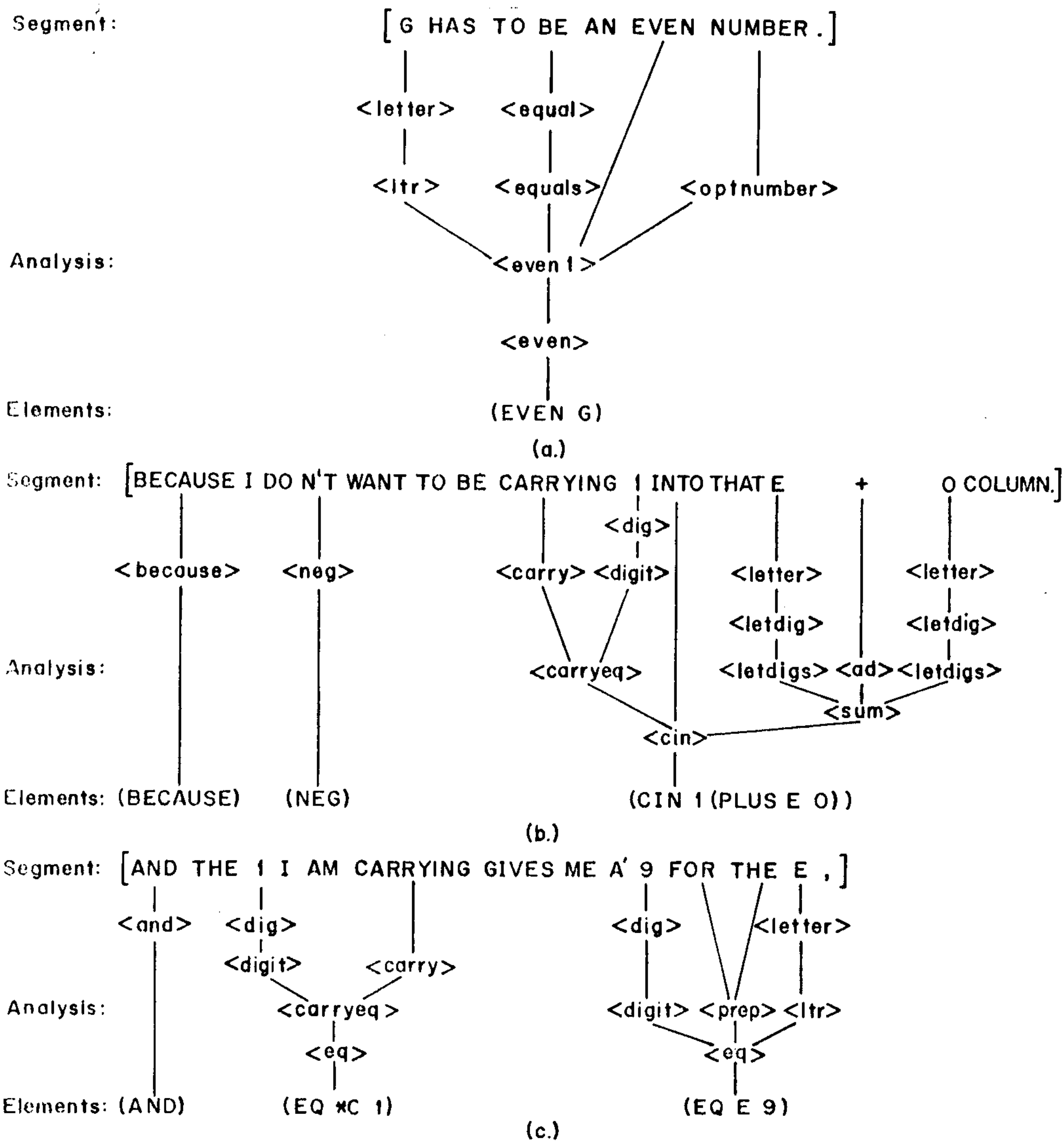


FIGURE 7. Operation of the Linguistic Processor

The grammar of Figure 5 is based on several kindsof knowledge beyond that of basic English. There is the task of cryptarithmetic, which determines many of the terms used. There is the definition of the problem space, which dictates the set of meaningful semantic elements, and thus which sorts of grammatical classes much be included. Finally, general knowledge of how people compose spontaneous speech determines another set of classes, especially those involved in tying parts of the verbal stream together (e.g., the conversational use of "if" or "therefore").

Semantic Processor

The next stage after the Linguistic Processor is the Semantic Processor (recall Figure 4). In this stage the semantic elements from the Linguistic Processor are arranged into tentative operator groups, called protogroups. Many of the semantic elements do not survive this stage in explicit form, either because they are assimilated into more specific elements or because they indicate input or output relations, and become represented in the structure of the grouping.

We assume that each of the knowledge elements used by the subject must be the ouput of some operator element or the recall of an element previously produced by an operator. Thus when an operator produces an output, all of the operator's inputs must be known to the subject. Similarly, the outputs of an operator are known to the subject the moment they are produced. These considerations lead to the following representational principle:

> The stream of semantic elements can be represented by a sequence of disjoint segments, called operator groups (or just groups), consisting of a single operator with its inputs and outputs, such that each input knowledge element was produced earlier as the output of some operator.

The operator group principle does not reflect any particular assumptions about memory structure. Assumptions about a division of current knowledge between a short-term memory, a long-term memory and an external memory (such as a scratch pad) must be added. They will affect accessibility of the various knowledge elements in the current knowledge state. PAS-I does not yet have an explicit model of a memory structure. In effect we assume the following continuity principle: that knowledge once produced is available in the knowledge state thereafter.

Temporal integration. Semantic processing occurs in several steps, each identified with the application of certain classes of rules. Figure 8 shows these steps in conjunction with a specific example. The semantic elements, as delivered by the Linguistic Processor, are to the left. In the first step sequences of elements are put together using the set of rules given in Table 2. In this table each line to the left of the arrows represents a set of knowledge elements (i.e., the knowledge extracted from a single segment), and each line to the right, the set resulting from the application of the rule. Note that some rules contain more than one line; these reflect relationships existing between segments. The single-line rules reflect relationships within a segment. Step 1, temporal integration, consists of applying these rules to each set of knowledge elements produced by the Linguistic Processor. The rules applied to a particular set are all those applicable using the ordering shown in the table. Thus rule 2 is applied to the result obtained from the application of rule 1. For a multi-line rule to be applicable not only must the first line of the rule apply to the set, but also the remaining lines must apply to the sets following the one in question. Ordering and contiguity are relevant between lines which form a rule but are

Initial
Semantic Elements

```
(PLUS L L)
(BECAUSE)(EQ C2 1)
(ODD R)
(EQ R 1)
(DIGIT 3)
(NEG)(DIGIT 5)
(DIGIT 7)
(OR)(DIGIT 9)
(EQC(PLUS D G)R)
(ASSUME)(DIGIT 9)
(AND)(EQ D 5)
(THEREFORE)(EQ G 3)(OR)(DIGIT 4)
(AND)
(?)
(COUT 1(PLUS O E))
(THEREFORE)(LETTER G)
(NEG)(EQ G 4)
(?)
(PLUS O E)
(EQ E 0)
(DIGIT 9)
(IF)(EQ E 9)
(THEN)(EQ *C 1)
```

STEP 1 ⟶

```
(PLUS L L)
(BECAUSE)(EQ C2 1)
(ODD R)
(MEQ R 1 3 7 9)
(NEQ R 5)
(EQC(PLUS D G)R)
(AEQ *L 9)
(AND)(EQ D 5)
(THEREFORE)(MEQ G 3 4)
( )
(EQ C6 1)
(THEREFORE)(EQ G *D)
(NEQ G 4)
( )
(PLUS O E)
(MEQ E 0 9)
(IF)(EQ E 9)
(THEN)(EQ *C 1)
```

STEP 2 ⟶

```
(PLUS L L)
(BECAUSEOF(EQ C2 1)(ODD R))
(MEQ R 1 3 7 9)
(NEQ R 5)
(EQC(PLUS D G)R)
(BECAUSEOF((AEQ *L 9)(EQ D 5))(MEQ G 3 4))
( )
(BECAUSEOF(EQ C6 1)(EQ G *D))
(NEQ G 4)
( )
(PLUS O E)
(MEQ E 0 9)
(COND(EQ E 9)(EQ *C 1))
```

STEP 3 ⟶

Protogroups

```
(PLUS L L)
(BECAUSEOF(EQ C2 1)(ODD R))
(MEQ R 1 3 7 9)
(NEQ R 5)
```

first proto-group

```
(EQC(PLUS D G)R)
(BECAUSEOF((AEQ *L 9)(EQ D 5)) (MEQ G 3 4))
```

second proto-group

```
(BECAUSEOF(EQ C6 1)(EQ G *D))
(NEQ G 4)
```

third proto-group

```
(PLUS O E)
(MEQ E 0 9)
(COND(EQ E 9)(EQ *C 1))
```

fourth proto-group

Figure 8. Example of Semantic Processor Operation

1.	(*type* (PLUS l_1 l_2))	→	(PLUS l_1 l_2) (*type* (PLUS l_1 l_2))
2.	(PLS *dig*)	→	(EQ *C *dig*)
3.	(PLS d)	→	(EQ *L d)
4.	(PLS l)	→	(LETTER l)
5.	(NEG) (*eqs* l d)	→	(NEQ l d)
6.	(NEG) (GREATER l d)	→	(NGREATER l d)
7.	(NEG) (SMALLER l d)	→	(NSMALLER l d)
8.	(NEG) (DIGIT d)	→	(NDIGIT d)
9.	(NEG) (LETTER l)	→	(NLETTER l)
10.	(ASSUME) (LETTER l_1)	→	(OPIO (FA(EQ l_2 *D))( ) (EQ l_1 *D)) (AEQ l_1 *D)
11	(THEREFORE) (LETTER l_2)		
12.	(ASSUME) (*equ* l d)	→	(AEQ l d)
13.	(ASSUME) (GREATER l d)	→	(AGREATER l d)
14.	(ASSUME) (SMALLER l d)	→	(ASMALLER l d)
15.	(ASSUME) (DIGIT d)	→	(AEQ *L d)
16.	(ASSUME) (LETTER l)	→	(AEQ l *D)
17.	(THEREFORE) (DIGIT d)	→	(THEREFORE) (EQ *L d)
18.	(THEREFORE) (LETTER l)	→	(THEREFORE) (EQ l *D)
19.	(*car* d (PLUS l_1 l_2))	→	(EQ C_n d)
20.	(EQ l d_1) (INSTEADOF) (DIGIT d_2)	→	(NEQ l d_2) (EQ l d_1)
21.	(EQ l d_1) (OR) (DIGIT d_2)	→	(MEQ l d_1 d_2)
22.	(EQ l d_1) (DIGIT d_1)	→	(EQ l d_1)
23.	(EQ l d_1) (DIGIT d_2) ⋮ (DIGIT d_{i-1}) (NEG) (DIGIT d_i) (DIGIT d_{i+1}) ⋮ (DIGIT d_m)	→	(EQ l d_1) (DIGIT d_2) ⋮ (DIGIT d_{i-1}) (DIGIT d_{i+1}) ⋮ (DIGIT d_m) (NEQ l d_i)
24.	(EQ l d_1) (DIGIT d_2) ⋮ (DIGIT d_m)	→	(MEQ l d_1 $d_2 \ldots$ d_m)

25. (MEQ l d_1 d_2 $\ldots$ d_m) → (PEQ l d_1)
(PEQ l d_2)
$\vdots$
(PEQ l d_m)
(MEQ l d_1 d_2 $\ldots$ d_m)

where: *type* = EVEN , ODD , GREATER , SMALLER
dig = *D , 1 , 0
equ = EQ , PEQ
car = CIN , COUT
eqs = EQ , PEQ , AEQ

Table 2. Rules for Temporal Integration: Step 1 of the Semantic Processor.

(l stands for any letter, d any digit, and C_n the carry defined by the rule context.)

ignored within a line. For example, rule 8 applies to any set containing both (NEG) and (DIGIT d) in any order, while rule 22 applies only to contiguous sets arranged in the order shown.*

The basis for the rules in Table 2 is partly linguistic, partly conversational, and partly based on the fact that a subject operating within the problem space** seems to respond rationally to certain elementary aspects of the situation. For example, Rule 23 in Table 2 says to interpret (EQ R 7) followed by (DIGIT 9) as (MEQ R 7 9), i.e., as the statement that R must equal 7 or 9. Two assumptions are involved here. The first is that the dangling digit (9) is associated with the R. This is based primarily on linguistic premises. The second is that the relation is one of disjunction. The exclusion of conjunction is based on rational grounds -- that is, it doesn't make sense to say that R could be 7 and 9. It is possible that a correction was indicated -- something equivalent to "R is 7 ... no, it's 9." But in general there will be additional linguistic indicators associated with a correction, such as a word indicating negation.

Normalization. At the beginning of the second step the indicator elements left are those associated with establishing input and output relations: (THEREFORE), (IF), (THEN), (BECAUSE), (OR), etc. A series of ordered rules, given in Table 3, assimilates this information into elements of the form (BECAUSEOF ...) (COND ...) and (OPIO ...). These transformations effect a normalization that simplifies the inference problem occurring later in the analysis.

* Thus (EQ l d_1) must appear in a set directly preceding a set containing (DIGIT d_2).

** We assume the subject is operating within the so-called augmented problem space (Newell and Simon, 1972).

RULES

1.	A_1 (THEREFORE) A_2 (THEREFORE) A_3	→	(BECAUSEOF A_1 A_2) (BECAUSEOF A_2 A_3)
2.	A_1 (THEREFORE) A_2	→	(BECAUSEOF A_1 A_2)
3.	A_1 (THEREFORE)(EQC ...)	→	(BECAUSEOF A_1 (EQC ...))
4.	A_1 (BECAUSE) A_2	→	(BECAUSEOF A_2 A_1)
5.	(BECAUSE) A_1 A_2	→	(BECAUSEOF A_1 A_2)
6.	(IF) A_1 (BECAUSE) A_2	→	(BECAUSEOF A_2 A_1)
7.	(THEREFORE) A_1 (BECAUSE) A_2	→	(BECAUSEOF A_2 A_1)
8.	(IF)(NEG)K_1 (THEN)(NEG)K_2	→	(OPIO (FN K_1) $\overline{K}_1$ $\overline{K}_2$)
9.	(IF)(AEQ...) A_1	→	(BECAUSEOF (AEQ ...) A_1)
10.	(IF) A_1 A_2	→	(COND A_1 A_2)
11.	(AEQ...) (IF) A_2	→	(BECAUSEOF (AEQ ...) A_2)
12.	A_1 (IF) A_2	→	(COND A_2 A_1)
13.	A_1 (THEN) A_2	→	(BECAUSEOF A_1 A_2)

14.	K_1 (OR) K_2 (WNOT) K_3 (OR) K_4	(COND K_3 K_1) (COND $\overline{K}_3$ K_2) (COND K_4 K_1) (COND $\overline{K}_4$ K_2)
15.	(MEQ l d_1 d_2) (WNOT) K_1 (OR) K_2	(COND K_1 (EQ l d_1)) (COND $\overline{K}_1$ (EQ l d_2)) (COND K_2 (EQ l d_1)) (COND $\overline{K}_2$ (EQ l d_2))

DEFINITIONS

A_i ≡ A sequence of knowledge elements from adjacent sets of semantic elements, where each set is connected to the adjacent ones through the (AND) indicator.

K_i ≡ A single knowledge element (its negation is represented by $\overline{K}$).

Table 3. Rules for Normalization: Step 2 of the Semantic Processor

Examples of applications of some of the rules illustrated in Table 3 are given in Figure 8. Note that A_i in Table 3 represents a sequence of knowledge elements from sets connected through the (AND) indicator.* For example, rule 2 can be applied to the sets

```
(EQ C6 0)(YES)
(AND)(EQ G2)
(AND)(EQ D 5)
(THEREFORE)(EQ R 7)
(AND)(EQ C7 0)(QUES)
```

to produce (BECAUSEOF ((EQ C6 0)(EQ G 2)(EQ D 5)) ((EQ R 7)(EQ C7 0))). This is interpreted as "Because C6=0, G=2 and D=5, we know that R=7 and C7=0." BECAUSEOF and OPIO are temporary knowledge elements used to convey information from the Semantic Processor to the Group Processor. COND, on the other hand, is a permanent knowledge element; it is incorporated intact into the PBG.

Grouping. In the third step grouping takes place and a tentative operator group, called the protogroup, is produced. This protogroup is defined to be the largest consecutive sequence of elements that contains no empty elements,** no more than one operator element, and at least one non-empty element. Thus a sequence of empty elements cannot be grouped and will instead be interpreted as a single empty element. In the example of Figure 8, four protogroups are produced as output.

[*] This will eventually be extended to include the (OR) indicator.

[**] In Figure 8 the empty elements are those indicated by ( ).

Group Processor

The output of the last step of semantic processing is a single protogroup (recall Figure 4). This protogroup is revised and refined during the next stages of analysis (Determine Unknowns Mechanism and Origin Mechanism) to produce one or more operator groups, which are then incorporated into the current PBG by the PBG Generator. Then the cycle starts again: a new protogroup is created, refined, and incorporated into the PBG. If any of the knowledge elements in a protogroup are not used (i.e., are not incorporated into the PBG) they are saved and defined as part of the next protogroup. Thus knowledge elements occurring between two operators have a chance to be included in the protogroup for each operator.

Determine Unknowns Mechanism. Many of the elements produced by the Linguistic Processor are incomplete, that is, they contain variables, denoted *L, *D, *C, etc. Examples of these occurred in Figures 7 and 8. The Linguistic Processor provides partial information for these variables by placing them in restricted domains, e.g., the set of digits (*D), the set of letters (*L), the set of carries (*C), or the set of sums (*X). The job of the Determine Unknowns Mechanism is to discover the values of these variables.

An informal description of the heuristics employed by the Determine Unknowns Mechanism is given in Table 4. This mechanism requires information about the current knowledge state of the subject, and thus must access the PBG. Furthermore, it must be able to assess the possibility of a given knowledge element being derived from a column. To do so it uses a basic routine, also used in later parts of the program, for deducing values from a column, given any of several variations of input information (i.e., the scientist's version of PC, the column processor).

I. If the incomplete element is a knowledge element, match it against the elements comprising the current knowledge state. If an identical element is found, use it.

II. If the incomplete element is a process column operator, match the known letters in the element against the letters of each column. Pick the column with the most letter matches. In case of ties, pick the column most recently processed.

III. If the incomplete element is a knowledge element (other than BECAUSEOF or COND) not in the current knowledge state, compile a list of columns which, through processing, could possibly produce the element. Base the list on information about letter matches and the columns most recently processed. Process each column on the list, in a one-step attempt to produce an element which matches the incomplete one. The first match obtained is the one used.

IV. If the above steps fail, then attempt to generate an element which matches the incomplete element, based on the current knowledge state but independent of column processing, i.e., (ODD R) and (GREATER R 7) leads to (EQ R 9).

V. If the incomplete element is BECAUSEOF or COND then:

   a. If the right part of the element has unknowns, attempt to fill them in as in I, II, III, and IV above. For this calculation, all elements (without unknowns) from the left part are considered part of the current knowledge state.

   b. If the left part has unknowns, and the right part still has unknowns or contains either PEQ or AEQ, then attempt to fill in the left part as in I and II above.

Table 4. Heuristics Employed by Determine Unknowns Mechanism (See Figure 9 for specific examples.)

Figure 9 shows examples of input and output sequences for the Determine-Unknowns Mechanism, which includes the use of all five heuristics from Table 4. The various sequences are annotated to show the information source and the heuristic applied.

Origin Mechanism. The primary goal of the Origin Mechanism is to infer the problem space elements that are missing from the topic segments and to infer the values of the unknown variables that the Determine Unknowns Mechanism failed to determine. A secondary goal is to produce a set of operator and knowledge elements that mutually satisfy the conditions of the operator-group principle. We assume semantic elements are arranged as operator groups with a full complement of input and output elements. Thus, from the existence of a knowledge element, we can infer the existence of an operator that produced it; and from an operator we can infer the existence of its inputs and outputs. The proliferation of conjectured operators and knowledge elements is terminated either because the knowledge elements already exist in the knowledge state of the subject or because the operators do not require further inputs. An example of this latter case in cryptarithmetic is the assignment operator (AV), which needs no inputs.

Table 5 gives knowledge elements from the augmented problem space for cryptarithmetic (Newell and Simon, 1972), and with each element are listed the operators capable of producing the element. This table forms the basis for inferring new elements. There is a relatively trivial solution to this construction problem if all unexplicit operators are assumed to be assignments -- i.e., all knowledge elements are simply assumed to be stipulated. The difficulty with this solution is its failure to explain how subjects happen to stipulate things that are the results of various computational and inferential processes. That is, if the current

Information Source

Knowledge State: (EQ D 5)(EQ C1 0)(EQ T 0)(SMALLER N 3)(ODD R)(GREATER R 7)(EQ C3 1)

```
            c6 c5 c4 c3 c2 c1
             D  O  N  A  L  D
Display:   + G  E  R  A  L  D
            -----------------
             R  O  B  E  R  T
```

Column Last Processed: 1

Input/Output Sequences

Heuristic			Examples	
I	(GREATER *L 7)	=>	(GREATER R 7)	
	(EQ T *D)	=>	(EQ T 0)	
II	(PLUS D *L)	=>	(PLUS D D)	
	(EQC (PLUS 5 5)*L)	=>	(EQC (PLUS 5 5)T)	
III	(EQ C2 *D)	=>	(EQ C2 1)	column 1 processed
	(MEQ E *D 9)	=>	(MEQ E 0 9)	column 5 processed
IV	(EQ R *D)	=>	(EQ R 9)	
	(MEQ N *D *D)	=>	(MEQ N 1 2)	
V	(BECAUSEOF (EQ *C 1)(ODD *L))	=>	(BECAUSEOF (EQ *C 1)(ODD R))	
	(COND (EQ A 4) (EQ E *D))	=>	(COND (EQ A 4)(EQ E 9))	

Figure 9. Examples of Input and Output for the
Determine Unknowns Mechanism
(See Table 4 for definitions of the heuristics.)

KNOWLEDGE ELEMENTS	OPERATORS
EQ	PC, GN, IG, FA, TD, TE, AV
PEQ	PC, GN, FA
MEQ	PC, GN, FA
NEQ	FN, TD, TE, PC
AEQ	FA, AV
EVEN	PC, FA, TD, TE
ODD	PC, FA, TD, TE
GREATER	PC, FA, TE
SMALLER	PC, FA, TE

Table 5. Knowledge Elements and the Operators for Generating Them

knowledge state of a subject contains (EQ E 8) and (EQ L 3) and the subject says (EQ A 4), it could be assumed either that an assignment operator (AV) produced (AEQ A 4) or that a column processor (PC) derived it from column 3 (i.e., from A + A = E). While coincidence is on the side of AV, rationality is on the side of PC.

When a set of elements is inferred, each such construction must remain tentative to some degree. The ultimate test of validity does not lie in the immediately surrounding data situation and the heuristics used there, such as giving precedence to PC over AV, but in the way the entire protocol analysis appears when finished.

Figure 10 shows the flow diagram for the current version of the Origin Mechanism. We always infer from a knowledge element the existence of an operator with associated inputs,and never infer from an operator the existence of its inputs and outputs. The rationale is that we can save operators with no apparent inputs or outputs, and use them to guide later searches for operators.* Thus, the process illustrated in Figure 10 starts with a knowledge element that has no operator. The Origin Mechanism engages in a breadth-first heuristic search for the missing operator, considering possibilities that could have led to the given knowledge element, that is, possible combinations of operators with specific inputs. In actuality, not all sets are considered as this would lead to an excessively large search tree. The following rather powerful heuristic is used to prune the search tree: for the PC operator both the carries but only one unknown letter in a column may be hypothesized as an input. Thus to obtain (EQ B 2) from (PC 4), either N or R could be hypothesized to be some value, but not both. Consequently, if both N and R are unknown then (PC 4) cannot be considered capable of producing (EQ B 2).

* At present this feature is not implemented.

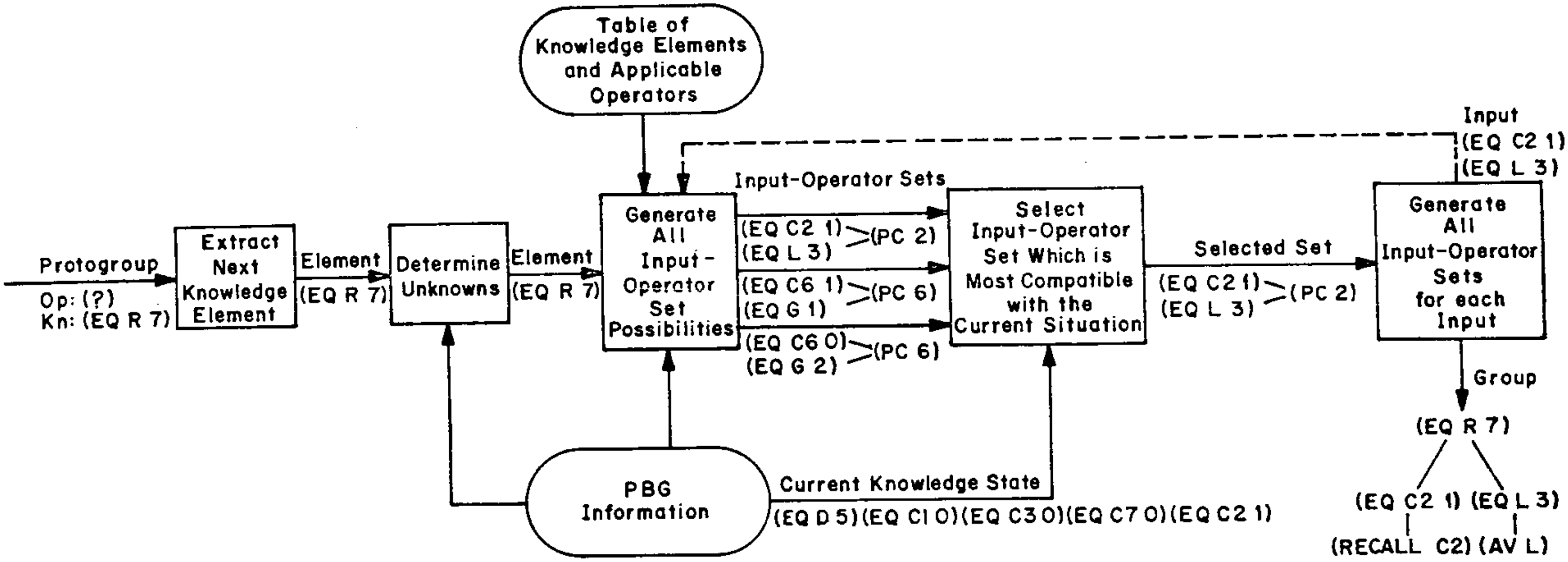


FIGURE 10. Flow Diagram of the Origin Mechanism

These operator-input combinations constitute a set of hypotheses about how the element could have been produced. The Origin Mechanism then selects one of these hypotheses for further exploration. To explore the hypothesis it must take each of its inputs, which is a knowledge element itself, and determine its origin. These inputs may be identical to elements already in the knowledge state, or may require the additional step of hypothesizing an operator and obtaining yet another set of input elements whose origins need to be accounted for.

At each stage of the search one hypothesis (an operator with specific inputs) is selected for further exploration. Three factors are responded to in this selection. First, positive credit accrues for using inputs already in the current knowledge state, these being called used-inputs. Second, negative credit accrues for using inputs that require further hypothesizing, these being called unused-inputs. Third, no credit accrues to an operator that is easily satisfied, in particular AV which has no inputs to be satisfied. The decision function currently in use is:

Choose to maximize: ((3 x used-inputs) - unused-inputs)

Break ties according to order: PC, GN, TD, IG, AV.

Figure 11 illustrates the operation of the Origin Mechanism. The current knowledge state is given at the top of the figure. Under it and heading the tree is the knowledge element (EQ A 4) which is given as input to the Origin Mechanism. At the first level three separate hypotheses that could have produced (EQ A 4) are generated, two involving the processing of column 3, each with a different set of inputs, and one involving assignment. These hypotheses were rated as shown by the subscripts on the operators. Note that at each level the hypothesis selected is the one with

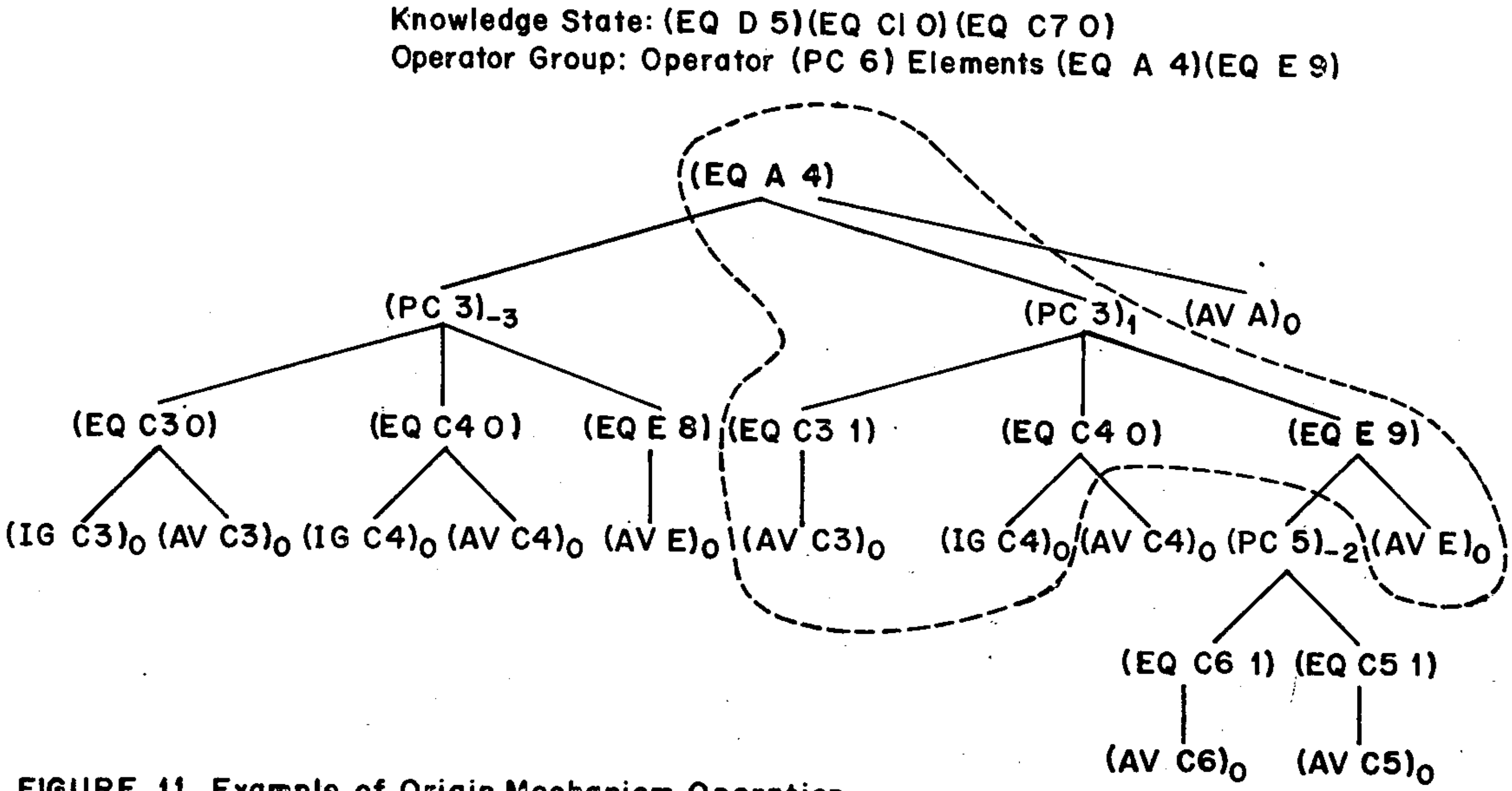


FIGURE 11. Example of Origin Mechanism Operation

the highest rating. The encircled branches show the path chosen to represent the origin of (EQ A 4).

PBG Generator

The final stage of PAS-I produces the problem behavior graph. This graph is generated incrementally, by taking the current PBG and adding to it the next operator group. Time continuity is important, since a key issue is whether or not the new information is consistent with the old; inconsistent information is not allowed and leads to a restructuring of the PBG.

The input to the PBG Generator is an operator group, i.e., an operator with a set of inputs and outputs. Actually, a set of operator groups are given, corresponding to a complete investigation by the Origin Mechanism. For example, in the case of Figure 11, four groups would be presented, one for (PC 3), one for (AV C3), one for (IG C4) and one for (AV E).

Backing up. We assume the subject traces some trajectory in the problem space. For this to be tree structured, as in Figure 2, the subject must return to prior states of knowledge at various times during the course of the trajectory. In cryptarithmetic this can happen for a number of reasons. First, the subject may discover a contradiction in his current solution. He will then abandon the information which initiated the contradiction, returning to some prior point in the problem space. He does not necessarily forget this information, rather he abandons it. The difference is that in abandoning the information he removes it from the current knowledge state, as in forgetting, but retains the knowledge that it was abandoned.

Table 6 defines the pairs of elements which may result in a conflict, assuming both elements of the pair are describing the same letter. The 1's indicate that a conflict is possible, the 0's that a conflict is not possible

		Second Element of the Conflict Pair								
		EQ	AEQ	PEQ	NEQ	MEQ	EVEN	ODD	GREATER	SMALLER
First Element of the Conflict Pair	EQ	1	1	1	1	1	1	1	1	1
	AEQ	1	1	0	1	1	1	1	1	1
	PEQ	0	0	0	1	1	1	1	1	1
	NEQ	1	1	1	0	1	0	0	1	1
	MEQ	1	1	1	1	1	1	1	1	1
	EVEN	1	1	1	0	1	0	1	1	1
	ODD	1	1	1	0	1	1	0	1	1
	GREATER	1	1	1	1	1	1	1	1	1
	SMALLER	1	1	1	1	1	1	1	1	1

Table 6. Conflict Matrix

(1: conflict possible,
0: conflict not possible.)

(by definition). For example, if (EQ R 6) is currently active in the PBG and (PEQ R 3) is then added to the PBG the element pair is (EQ R 6), (PEQ R 3), which has an entry of 1 in Table 6 indicating that a conflict is possible. In this case a conflict exists; in fact the only case where a conflict won't exist with the pair EQ,PEQ is when both elements use the same digit, i.e., as in (EQ R 6), (PEQ R 6). However, if the element pair is (PEQ R 3),(EQ R 6), meaning that now the PEQ element is already in the PBG and the EQ is being added, the entry in Table 6 is 0. Thus no conflict occurs (by definition) regardless of the values used for the digits.

A second reason for backing up in the problem space is the abandonment of (perceived) irrelevant information. Unlike the issue of contradictory information, the pressures toward this stem from the limited memory capacities of the human. Human long-term memory, though it has adequate capacity, is much too slow in its acquisition rate compared to the rate at which new nodes in the problem space can be generated. Consequently when information not yet acquired in long-term memory is discovered to be irrelevant it will simply be abandoned.

A third reason is that relevant information can occasionally be forgotten. However, this occurs rarely in situations where the subject is self paced, as in the problems under discussion here. The subject will generally employ a strategy that is not self-destructive, i.e., that permits sufficient time to remember the relevant information called for by the strategy.

Restructuring rules. The above digression into the underlying theory provides the rationale for the rules used in the construction of the PBG. These rules, together with a set of explanatory definitions, are given in Table 7. The rules are in the form of an ordered set of production rules (Waterman, 1970), and are applied by comparing a vector composed of the current state vector values with the left side of each rule, restructuring according to the action specified by the right side of the first rule matched. They are

DEFINITIONS

STATE VECTOR: (variables affecting the PBG-restructuring rules)
state vector = (TYPE HAVETD HAVEFN INCLUDE)

where

TYPE ≡ the type of restructuring problem, i.e., SIMILARITY or CONFLICT.

HAVETD ≡ the question "is the second knowledge element of the similarity or conflict pair a TD element?", i.e., T or F.

HAVEFN ≡ the question "is the second knowledge element of the similarity or conflict pair a FN element?", i.e., T or F.

INCLUDE ≡ the question "are all output elements of cn1 included in those of cn2?", i.e., T or F.

and cn1 designates the operator which produced the first knowledge element of the similarity or conflict pair, cn2 designates the operator which produced the second knowledge element of the similarity or conflict pair.

ACTIONS: (restructuring operations on the PBG)

BLOCK REJECTION: a restructuring involving returning to a previous knowledge stage by abandoning all nodes including and beyond cn1.

INCREMENTAL REJECTION: a restructuring by abandoning only the node cn1

COPYING: restructuring in which cn2 is not abandoned

CHAINING: restructuring in which cn1 is redefined to be the earliest node in the graph which produces an AEQ element used as input to the original cn1 node.

CURRENT RULES

(a). (SIMILARITY * F T) → BLOCK REJECTION, COPYING

(b). (CONFLICT * F *) → BLOCK REJECTION, COPYING, CHAINING

(c). (CONFLICT * * *) → BLOCK REJECTION, CHAINING

Note: These rules are applied in the order shown, where the first applicable rule defines the restructuring action taken. The left side of each rule refers to the current set of values comprising the state vector. An asterisk (*) indicates that the value of the state vector variable is irrelevant.

Table 7. Rules for PBG Construction

organized as production rules to facilitate the process of expanding and refining them. The rules can be extended two ways: by rule manipulation, i.e., adding, deleting, or redefining rules, and by state vector expansion, i.e., adding new variables to the state vector. We envision a system that will ultimately require a large complex set of heuristics for restructuring, thus the rules given in Table 7 represent only a first approximation to the required set.*

Restructuring, as defined by the rules in Table 7, leads to a graph which is tree-structured. As remarked earlier, the current knowledge state can be obtained by taking the conjunction of all elements which are the outputs of operators, starting from the current node back to the root. In the standard PBG orientation (in which time runs horizontally from left to right and then down) this consists of all the output elements lying along the lower (growing) edge of the tree. These elements are called the currently active output elements, and their nodes are called the currently active nodes. In the tree at the bottom of Figure 12 (used as an example below), the set of currently active nodes is indicated by the heavy line.

Example of PBG generation. The operation of the rules of Table 7 can be understood most easily by means of a simple example. Figure 12 shows a list of operator groups and the PBG generated by applying these rules. This is an artificial example, constructed to show the operation of all of the rules in a short sequence. Each restructuring in the figure is labeled with the rule used and the nodes responsible.

The first restructuring in Figure 12 is based on similarity and takes place after the eighth operator group is incorporated into the PBG. Note that nodes 5, 6 and 7 are abandoned but node 8 is copied and saved as node 8'.

* For example, the current rules do not contain incremental rejection (see Table 7), although this restructuring technique has been implemented and will be included in later versions of the system.

Initial or Given
Knowledge State: (EQ D 5)(EQ C1 O)(EQ C7 O)(EQ C2 1)

Operator Groups:

	Operator	Inputs	Outputs
1	(RECALL D)	( )	(EQ D 5)
2.	(RECALL CI)	( )	(EQ CI O)
3.	(RECALL C2)	( )	(EQ C2 1)
4.	(PC 1)	( )	( )
5.	(FC T)	( )	( )
6.	(FC D)	( )	(IN D 6)
7.	(COUNT R)	( )	(NUM R 3)
8.	(PC 1)	( )	( )
9.	(AV A)	( )	(AEQ A 4)
10.	(IG C3)	( )	(EQ C3 O)
11.	(PC 3)	(EQ C3 O)(AEQ A 4)	(EQ E 8)
12.	(PC 5)	( )	(MEQ E O 9)
13.	(FC L)	( )	(IN L 2)
14.	(AV L)	( )	(AEQ L 1)
15.	(PC 2)	(EQ C2 1)(AEQ L 1)	(EQ R 3)
16.	(RECALL C7)	( )	(EQ C7 O)
17.	(TD C7 1)	(EQ C7 O)	(NEQ C7 1)
18.	(FN (EQ C7 1))	(NEQ C7 1)	(NEQ R 3)
19.	(AV L)	( )	(AEQ L 3)
20.	(PC 2)	(EQ C2 1)(AEQ L 3)	(EQ R 7)

Problem Behavior Graph:

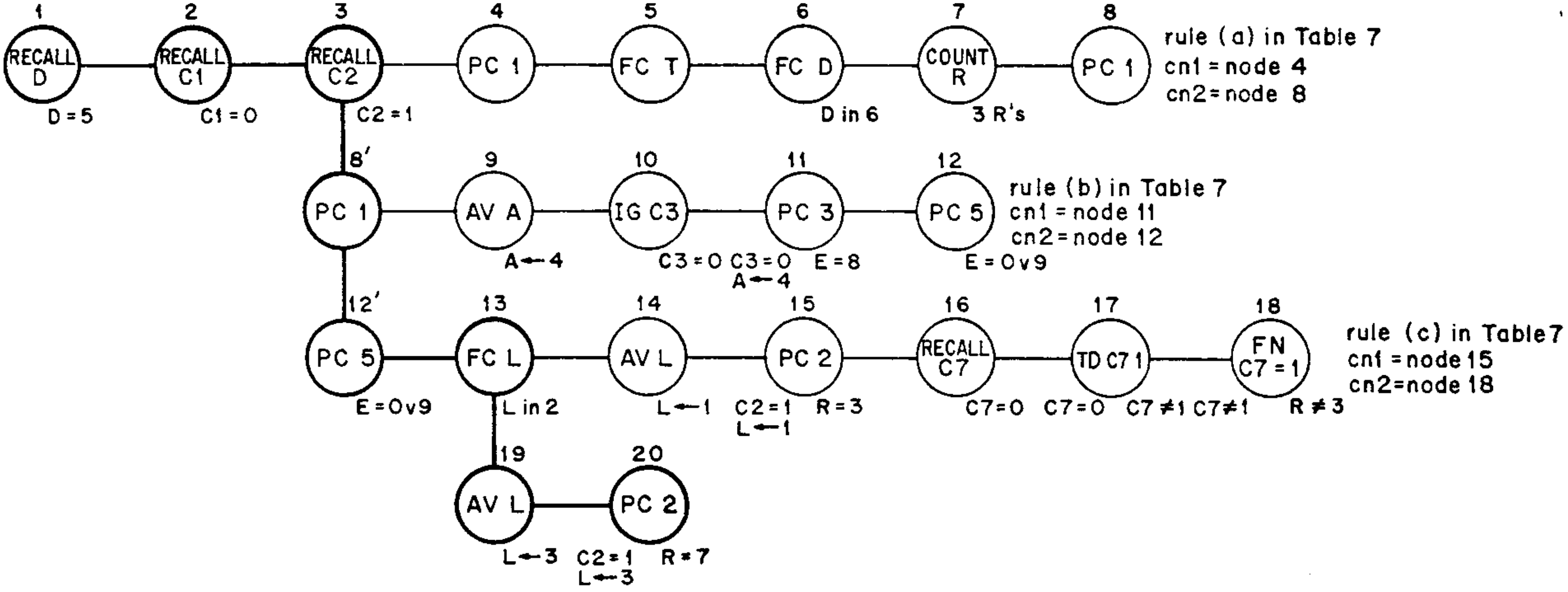


FIGURE 12. Example of PBG Generation

After the 12th operator group is grown a restructuring based on conflict occurs since (EQ E 8) is inconsistent with (MEQ E 0 9). Here, nodes 9 and 10, as well as 11, are abandoned since node 9 created on assignment element used as input by node 11. After the 18th operator group is grown another restructuring based on conflict takes place, since (NEQ R 3) is inconsistent with (EQ R 3). This leads to the abandonment of nodes 14 through 18.

Let us now reconsider the input to the PBG Generator. This input is a collection of operator groups, rather than a single one, since the Origin Mechanism produces a chain of groups in the process of filling in missing elements. Each chain is headed by a group located in time. This is the group holding as output the initial knowledge element that triggered the Origin Mechanism. The other groups are only bounded in time: they occur before the heading group and after their inputs are produced. Assimilation of these groups into the PBG requires some additional assumptions. The basic assumption* is that all the groups in the chain occur after time _t_, the time when the previous node was added to the PBG. Thus the groups in the chain are incorporated into the PBG, one at a time, starting with the last one created by the Origin Mechanism and ending with the heading group. One exception to this assumption currently exists. If a group in the chain is similar to a currently active node in the PBG (i.e., the group inputs are a subset of the node inputs and the operators are identical) then the group is integrated into the PBG by adding its output to the outputs of the similar node. The assumption

* As the system gains in sophistication, this assumption will include a large number of exceptions.

only applies to groups which are similar to recently grown nodes.* An example of integrating a group into the PBG in this manner is shown in Figure 13, where group 5 (with an hypothesized operator) is similar to node 3. Groups similar to currently active nodes which were not recently grown are handled by similarity restructuring as shown in Figure 12.

Control Structure of the Total System

The overall structure of the computation involves updating the PBG to time _t_ before processing the protogroups formed from the utterance at time _t_. The PBG represents the best estimate, at the given point in the computation, of what the subject knows. As we have seen in the Group Processor, this estimate is used extensively to infer the meaning of the current utterance and to infer from it additional knowledge the subject has and operators he has used.

The dependence on a current estimate of the subject's knowledge implies that the computation cannot be organized in a sequential fashion, such that each stage of processing (Linguistic, Semantic, Group, PBG) is completed as a separate pass. However, in PAS-I the Linguistic Processor and the first two stages of the Semantic Processor are independent of the PBG and make use only of the current utterance. This is a feature of the current heuristics used in PAS-I. It will not hold, for instance, if we want the system to reanalyse the raw utterance (the initially given topic segment) after obtaining from the Origin Mechanism an hypothesis about what the subject might actually have said.

* If the node has inputs or outputs it must be one of the last six nodes grown to be considered "recent," otherwise it must be one of the last three nodes grown.

Initial or Given
Knowledge State: (EQ D 5)(EQ C1 0)(EQ C7 0)

		Operator	Inputs	Outputs
Operator Groups:	1.	(RECALL D)	( )	(EQ D 5)
(H stands for	2.	(RECALL C1)	( )	(EQ C1 0)
hypothesized)	3.	(PC 1)	(EQ D 5)(EQ C1 0)	(EQ T 0)
	4.	(AV L)	( )	(AEQ L 1)
	5.	(PC 1 H)	(EQ D 5)(EQ C1 0)	(EQ C2 1)
	6.	(AV A)	( )	(AEQ A 4)

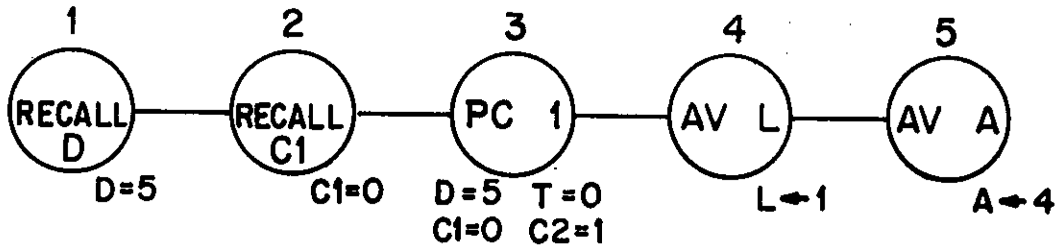


FIGURE 13. PBG Generation with Similar Groups

Thus the structure of the processing is to bring forward in time as complete a picture as possible of the subject's current state of knowledge. This means that changes in this picture provided by later evidence could invalidate earlier processing. PAS-I does not yet involve a recycling of the computation to do over again the processing of earlier parts of the protocol. The types of changes that can occur in PAS-I, such as restructuring of the PBG, do not yet require this.

V. PRELIMINARY RESULTS

We are now ready to present some results of runs with PAS-I. At this early stage, interest focuses on the operation of the system; consequently summary statistics of how well the system performs are of secondary importance. Our approach is to run the system on protocols already analyzed by hand, and to note and discuss various similarities and differences.

We present three examples. The first is an initial segment of the protocol with S3. This is the one used primarily in developing the system. The second is a subsequent segment of S3, following directly after the initial one. The third example is a protocol for another subject, S4, on the same task, DONALD+GERALD=ROBERT. These three examples provide a reasonable picture of the state of the current system.

Performance on B1-100 of S3

We present in Appendix I the output of PAS-I on the first 100 topic segments of the protocol for S3. Section I.1 of Appendix I gives the initial linguistic and semantic processing. Each topic segment* taken as input is listed (B1, B2, etc.). Immediately below each input topic segment is given the results of applying the Linguistic Processor to the topic segment in isolation. Just below this is the output of the first two stages of the Semantic Processor, the stages which perform integration and normalization. This output does not occur with each topic segment, since the Semantic Processor can put together the information in several segments. We associate the output of the first two stages of the Semantic Processor with the last

* This text is exactly as it occurs in the original manual analysis (Newell, 1967), except for the following typographic conventions which provide various separations and disambiguations: (1) the plural of NUMBER is written with 'S, (2) 'S, 'LL, 'M, 'RE, and N'T are written with an extra space in front, e.g., I'M → I 'M, and DON'T → DO N'T, (3) certain negative contractions are expanded, e.g., CAN'T → CAN N'T, and (4) A used as an article is rewritten A'.

topic segment it uses. For example, the output for topic segments B6 and B7 has the following format:

B6. Sixth topic segment.

Initial semantic elements for sixth topic segment.

B7. Seventh topic segment.

Initial semantic elements for seventh topic segment.

B6-7. (Processed semantic elements for segments B6 and B7).

Section I.2 of Appendix I gives the trace of PAS-I as it conducts the remainder of the analysis: selecting a protogroup from the list of processed semantic elements, determining unknowns and origins, and growing the PBG. At the beginning of each major cycle of this analysis the name of the extracted protogroup is printed, with a tentative identification of its knowledge elements and operators. The source of the conjecture of the operator is printed with each operator. For example in the first protogroup, B5-7, the operator (PC 1) is conjectured from the existence of the semantic element (NUM D 2).

The system takes each element of the protogroup in turn and applies the appropriate mechanisms: the Determine Unknowns Mechanism, if variables are present, and (in all cases) the Origin Mechanism. Intermediate results provided by the Determine Unknowns Mechanism are printed out when they occur. The result for each element is an origin list consisting of the set of operator and knowledge elements to be grown onto the PBG. Each evocation of the PBG Mechanism is noted by printing the disposition of the element under consideration -- e.g., a new node was grown (NDi for operator nodes and Ki for knowledge elements), the node was recognized as an old one, a conflict was detected, etc.

Thus the entire course of the processing of PAS-I can be followed via sections I.1 and I.2 of Appendix I. A graphical representation of the final PBG is shown in Figure 14.* The PBG was folded to get it on a single page; one should imagine it stretched out, as shown in the dot-graph at the bottom of the figure.

Comparison of PBG's. To obtain some feeling for the quality of PAS-I on these first 100 nodes, we compare its output with the PBG produced manually from the same information. Ultimately, each step (i.e., linguistic processing, integration, normalization, grouping, determining unknowns, etc.) of the processing must be evaluated, but we must be careful in doing so. Many early stages are weak precisely because they are compensated for by later stages. Conversely, improving the response of early stages is futile unless later stages have mechanisms that use the additional information. Thus, the appropriate strategy for evaluation is to start with the final results and work backward into the system in an attempt to diagnose the causes of good or bad performance.

Figure 15 gives the PBG for the first 100 topic segments as produced by the original manual analysis (Newell and Simon, 1972) in the notation of PAS-I. Comparing two PBG's requires putting their nodes into correspondence and making judgments about the degree of equivalence between corresponding nodes and the status of the extra nodes occurring in each PBG (i.e., nodes in PAS-I with no correspondent in the manual analysis and vice versa). Besides correspondence between nodes (operator groups) we need to compare the PBG's on the positions and occurrences of backups. Backup comparison is accomplished after

* This graph is a manually drawn version of the graph produced by PAS-I on the printer.

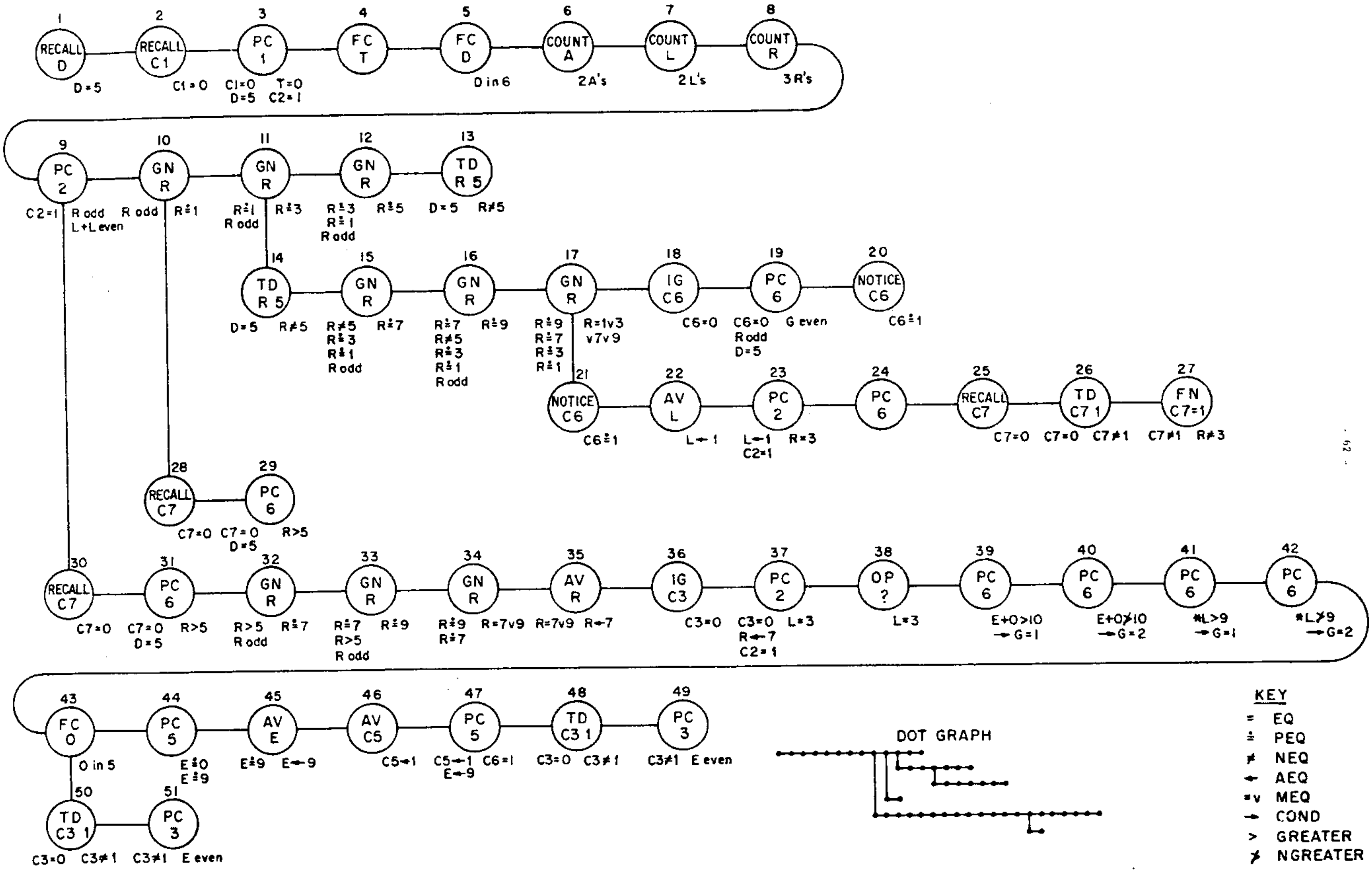


FIGURE 14. PBG by PAS-I for S3 on BI-100

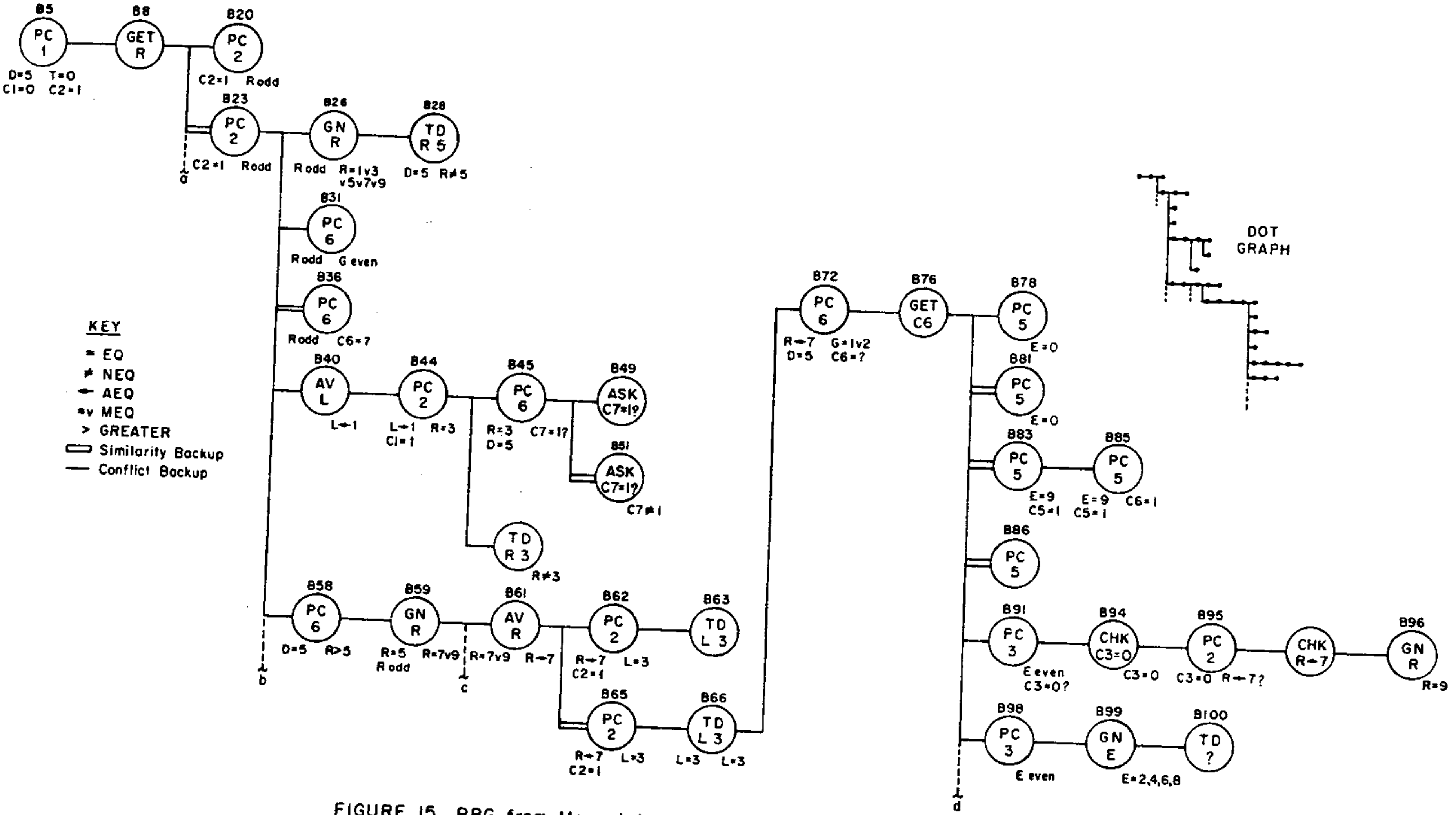


FIGURE 15. PBG from Manual Analysis for S3 on B1–100 (in Pas-I notation)

first placing the nodes in correspondence according to their content.

Section I.3 of Appendix I shows the correspondence between the two PBG's and records for each node and backup point an indication of its status (correspond and agree, extra node in PAS-I, etc.) and an indication of the source of the difficulty. This diagnosis of source is necessarily rough, being based on a judgment of what would have to be different in order to avoid the error.

Table 8 gives a summary of the comparison revealed by section I.1 of Appendix I. It lists separately each of the distinct types of correspondence, and segregates backups from nodes. The categories are mostly self-explanatory, except for the second where a PAS-I node is <u>subsumed</u> in a manual node. This category arises because PAS-I nodes often correspond to something implicitly assumed to occur within a manual node. For example, PAS-I posits a RECALL operation for variables whose values are given (D = 5), whereas the manual analysis simply takes the recall of the value as occurring within the appropriate PC. The most elaborate example is the detailed working out of GN by PAS-I, where each separate act of generation is represented by a separate node. In contrast, the manual analysis lumps all the steps together. This permits PAS-I to localize correctly the occurrence of a TD operator that detects that R cannot be 5 (B28) while generating all odd digits for R. Also, PAS-I posits certain attentional operators (FC and FL) that occur in the manual analysis at the next stage in the production system. Thus, subsumed nodes reflect design decisions in PAS-I that make its grain somewhat finer in places than that of the manual analysis.

Of the 80 total items in Table 8, 43 are in essential agreement (14 + 25 + 4), 34 exhibit some sort of disparity, and 3 items can be excluded as irrelevant. In section I.4 of Appendix I, we have annotated each of these disparities (including the excluded cases). From these annotations can

Types of Correspondence	Code	Number
Nodes correspond and agree	=n	14
PAS-I nodes subsumed in Manual node	s	25
Nodes correspond, agree/disagree	=≠n	1
Nodes correspond and disagree	≠n	2
Extra PAS-I nodes	+pn	3
Extra Manual nodes	+mn	17
		62
Backups correspond and agree	=b	4
Extra PAS-I backups	+pb	1
Extra Manual backups	+mb	10
		15
Cases excluded from consideration	x	3
		80

Table 8. Comparison between PBG of PAS-I and Manual Analysis for S3 B1-100 on D+G=R (see Appendix I.1).

be read off the occurrences of program deficiencies that caused the disparities (one deficiency can cause several disparities, as counted in Table 8). Table 9 lists these deficiencies, tying them to the number of items to which they contribute. The total number of items (38) exceeds the actual number (34), since several deficiencies can participate in a single item. We discuss each of the deficiencies briefly.

The Linguistic Processor failed clearly on three occasions. Some of these constitute (from our present vantage point) genuine difficulties, where the inferences used by the human analyst are complex. The one case where PAS-I simply failed was B62-B65:

B61. SO WE 'LL START BACK HERE AND MAKE IT A 7.
B62. NOW IF THE --
B63. OH, I 'M SORRY, I SAID SOMETHING INCORRECT HERE.
B64. I 'M MAKING --
B65. NO, NO, I DID N'T EITHER.
B66. R IS GOING TO BE A 7,
B67. THEN THIS WILL BE 7
B68. AND THAT WILL BE 7.
B69. AND IT 'S THE L 'S THAT WILL HAVE TO BE 3 'S,
B70. BECAUSE 3 + 3 IS 6
B71. + 1 is 7.

Both PAS-I and the manual analysis obtain (AV R) → (AEQ R 7) for B61. PAS-I makes nothing out of B62-B65 and then gets (PC 2) → (EQ L 3) for B66-71. However, the manual analysis argued as follows:

1. Something is happening at B62-65, probably a (PC 2), since that is what would follow (AEQ R 7).

    1.1. The phrasing "BACK HERE" in B61 helps localize the PC at column 2 rather than elsewhere.

2. B63 indicates that an error was made in (PC 2).

3. The result of the error is to make the path (AV R), (PC 2) fail.

4. B65 indicates that the judgment of error was seen to be in error, and thus the path is correct.

Deficiencies	Number of disparities produced
Linguistic Processor (LP)	
No output	4
Wrong output	5
Semantic Processor (SP)	
Merge two separate operators	2
Origin Mechanism (OM)	
Not seek origin of operator arguments	1
Not take immediate context into account	2
Wrong treatment of IG operator	1
Grow PBG Mechanism (GPBG)	
Merge two separate operators	10
Not use clue for merging: Redundant output	1
Not use clue for backup: Information unused	1
Missing concepts (-C)	
Goals	8
Experimenter	2
Minor difficulties (Minor)	
Backup to middle of compound operator GN	1
	38

Table 9. PBG Disparities caused by PAS-I Deficiencies for PAS-I vs Manual Analysis for S3 B1-100 on D+G=R (see Appendix I.1).

4.1 B66-71, which carries out the path successfully, supports this.

4.2 The language of B66 weakly supports this, though "IS GOING TO BE" is often used in other contexts.

4.3 The language of B69 more strongly supports this.

5. The language of B69 indicates that the error satisfies "*L HAS TO BE 3" where *L is not the letter L.

6. The subject has just returned from trying (EQ R 3) and finding that it cannot be (B57). Therefore the error (upon which the detection of error was based) was that R was 3.

7. If R had been taken to be 3, then TD would have produced an indication of an error (at B63).

8. Since R was assigned 7 (not 3), the original error must have been to derive from (EQ R 7) the value 3 by (PC 2) and then to confuse the letter for which this was to be the value (namely, to see it as the value for R, rather than the value of L).

9. The correction of the error was to see that the value just derived (3) was to be assigned to L and not R, hence that the TD on R and 3 was inappropriate.

10. If the above is posited, namely (PC 2) → (EQ *L 3) → (EQ R 3) followed by (TD R 3) → (NEG), then the language in the portion under consideration becomes interpretable and we can complete (in paraphrase at least) the fragmentary phrases:

B62. NOW IF THE (R IS 7, THEN I GET 3

B64. I 'M MAKING (THE R A 3 AGAIN, WHICH CAN N'T BE)

The above line of reasoning may not be conclusive, for the phrasing is indeed obscure. In any event, we want PAS to be able to construct such a line of reasoning.

Giving PAS-I such a capability requires substantial additions. First, PAS-I must be given the concept that a subject can make errors, be aware of them, and reveal this awareness in the varbalizations. Second, the notion of errors must be enlarged to include the misassignment of letters to digits on an occasion to occasion basis. Without these conceptual additions PAS-I could not hope to tackle this inference problem.

The second substantial failure of the Linguistic Processor is at B72-73:

B72. NOW, IT DOES N'T MATTER ANYWHERE WHAT THE L 'S ARE EQUAL TO --
B73. SO I 'M INDEPENDENT OF L WHEN I GET PAST THE SECOND COLUMN HERE.

The Linguistic Process detects the letter L and the negation in B72, and the letter L and the column reference in B73, producing

B72. (NEG) (EQ L *D)
B73. (THEREFORE) (LETTER L) (PLACE SECOND)

From this point on some sort of disparity is almost inevitable, and its details are not really interesting. To rectify this, PAS-I must be given the relevant concepts behind the attentional operators FC, F1 and FNC -- since the subject may use information about where letters occur, which columns are independent of which letters, etc. Although PAS-I makes use of a few such notions (as in distinguishing COUNT from PC), it is still relatively weak in this area.

The third failure of the Linguistic Processor occurs right at the end of the run (B98-100). At B98 it fails to use the grammatical clue of an article (AN) to infer from IF E HAS GOT TO BE AN -- to (EVEN E) rather than just (EQ E *D). This is the one place where a purely grammatical clue is critical (if indeed the inference is sound). The two following phrases (B99-100) make it clear that S3 applied some operators, while giving no indication of which: NOW, WAIT A SECOND, I GOT SOMETHING OUT OF THIS. The Linguistic Processor extracts nothing from this. Thus PAS-I misses the final branch of the manual PBG completely, though the failure is mitigated since the crucial information used in the manual analysis occurs after the end of the run (at B101-103).

The Semantic Processor appears to be responsible for only one disparity. At B35 the manual analysis sees the derivation of (EVEN G) and then a repeat of this at B36 leading to detecting the possibility of the carry. The Linguistic Processor picks up both of these adequately and keeps them separate as processed semantic elements. However, when the protogroup is selected it uses one of these as evidence for an operator and the other as evidence for knowledge elements, thus producing a single protogroup. After this there is no hope of separating the two. Examination of the language shows that there is no direct linguistic evidence for two operations, and that the inferences made in the manual analysis come from the knowledge elements, the order in which they occur, and the phrasing in B36, "I 'M LOOKING AT ...," which supports the notion of re-consideration. The basic conflict in interpretation is that sometimes clues about the order of elements are to be ignored and sometimes are critical. PAS-I still treats order information too uniformly.

The Origin Mechanism contains three deficiencies, each quite distinct. None require conceptual advances in PAS-I, though they do require additional mechanism. One revealed as a disparity at B92, is that PAS-I does not handle the subject's ignoring of values entirely satisfactorily. PAS-I posits an operator (IG x) when the subject ignores a value that he should have known (e.g., a carry). Given the role of operators, this makes ignoring into an active process that produces values that are remembered, just as if done by PC or AV.

The second deficiency, revealed at B83, is that our processing strategy of working with an updated PBG as the context for inference (the loop in Figure 4) is incomplete. PAS-I holds the context fixed while it generates the origin list for a particular knowledge element. Normally this works quite well. However, a problem can arise if an element being developed on one branch of the search tree is needed to provide context for an element being developed on a separate branch. In fact this is what happened at B83, where PAS-I developed the origin of (EQ C5 1) to be (AV C5), not taking into account the development the development of (EQ E 9) in another part of the tree, which implied that (EQ C5 1) was produced by (PC 5) with (EQ E 9) given.

The third deficiency, revealed at B76, is that PAS-I does not seek the origin of arguments to operator elements. The example at B76 is the origin of the letter O in (FC O), which is the first appearance of O in the protocol. If we had chosen to represent this argument by a knowledge element (LETTER O), then the Origin Mechanism would automatically have asked what produced (LETTER O). To be able to answer such a question requires that PAS-I have some operators that produce letters as output, e.g., that a PC on a column can also produce an awareness of the letters in that column. In the present case, there is no reasonable source of awareness of the letter O, and an alternative hypothesis would have been required to explain why column 5 was processed. The problem is not limited to letters, but applies to all direct arguments, such as the column in (PC 1) and the digits and letter in (TD R 3). Its solution appears to involve the creation of goal elements, such as (GET R), which is another type of element that has not yet been added to the system.

Three deficiencies were uncovered in the PBG Mechanism, accounting for 12 disparities in Table 9. By all odds the most important deficiency (accounting for 10 of the 12 disparities) was the merging of two identical PC operations, where the manual analysis distinguished a reconsideration.

Performance of the Linguistic and Semantic Processors. The comparison just made between the manual and automatic PBG's does not give the full picture. An independent judgment can be made about the adequacy of the Linguistic Processor to extract all the task relevant information from the topic segment and of the Integration and Normalization stages of the Semantic Processor to produce semantic elements that have all the information possible as input to the rest of the system. As we noted earlier, improvement of these early stages would be productive only if the remainder of the system could take advantage of it.

Figure 16 shows a graph of the results produced by the Linguistic Processor based on a judgment, for each topic segment in section I.1 of the appendix, of whether the correct semantic elements have been extracted and whether additional elements are missing. The judgments are necessarily tentative, since we are not relating the elements to how they were used by later stages of the system. Thus, of the 100 topic segments, four were excluded (by the decision not to encode Experimenter utterances), 10 were seriously missed and 86 were basically satisfactory (OK). Often, however, some elements were missed, even though the essential elements seemed to be extracted. Thus, the 86 satisfactory cases split into 58 that appeared complete and 28 where more information could have been obtained. In the figure we have also mapped the completely satisfactory elements into those that actually produced semantic elements as output (51) and those where the Linguistic Processor (properly, in our judgment) decided that nothing could be obtained (7). An example of the latter occurs at B91: LET 'S SEE --, which was processed as (?).

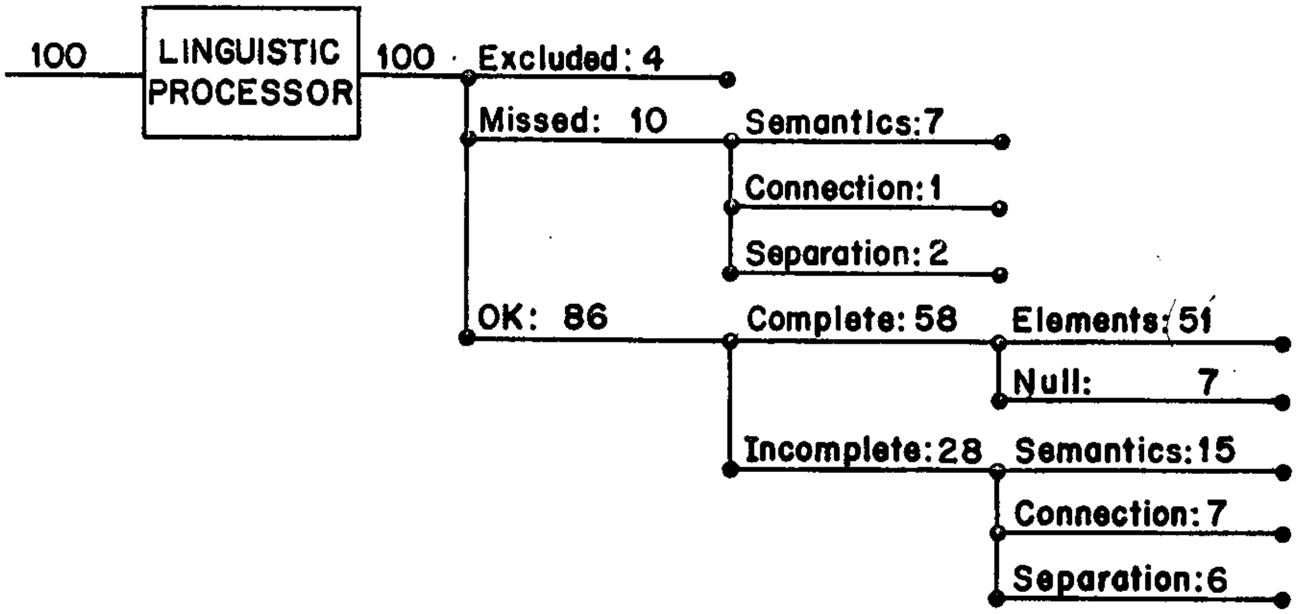


FIGURE 16. Performance of Linguistic Processor on BI-100 for S3.

The errors are categorized according to whether they involved a semantic element, a connection between semantic elements, or an indication of a separation between elements. An example of a complete miss on a semantic element occurs at B50:

```
B50.  IT 'S NOT POSSIBLE THAT THERE COULD BE ANOTHER LETTER IN
      FRONT OF THIS R IS IT ?

      (NEG) (PEQ *L *D) (LETTER R) (QUES)
```

Although a substantial amount of information was extracted, the Linguistic Processor completely missed the notion of IN FRONT OF, and, with it, any possibility of a correct interpretation later. The reason for the miss is that the grammar is incomplete, and the total system is weak with respect to locations of tokens in the display.

An example of missing a connection, this time where the basic translation was judged satisfactory, occurs at B25:

```
B24.  ANY TWO NUMBER 'S ADDED TOGETHER HAS TO BE AN EVEN NUMBER

B25.  AND 1 WILL BE AN ODD NUMBER

      (ODD (PLUS *X 1))
```

The AND is used to infer that an addition of 1 is being mentioned. The Linguistic Processor does not also infer a connection to B24, as it does elsewhere when it extracts (AND) as an indicator element.

An example of missing a separation occurs at B37, also in a basically satisfactory case:

```
B37.  OH , PLUS POSSIBLY ANOTHER NUMBER .

      (PLS *D)
```

The exclamation, OH, indicates (possibly) that something was seen that had not been seen previously, hence that the processing may have backed up or repeated. The system does not extract information about separation, thus the number of cases where separation information is missed gives an indication of whether adding such a capability to the system would be worthwhile.

The Linguistic Processor delivers an output for each input topic segment. Thus, the Semantic Processor is executed 100 times in dealing with the first 100 segments. It produces only 66 processed semantic elements, which are shown in the graph of Figure 17. Information was completely missed or misinterpreted in 16 cases whereas the remaining 50 were basically satisfactory, although, as with the Linguistic Processor output, a subset of these (here 9) had some additional elements missing.

The Semantic Processor must accept the initial semantic elements provided to it, and in many cases cannot improve on the input, either because the input is already satisfactory or because it is in error and there is no information from which to make a recovery. Figure 18 classifies the output of the Semantic Processor according to whether the output was essentially unchanged from the input (=), whether it was improved (+) or whether it was degraded (-). As can be seen, those elements which did not undergo significant change consist not only of satisfactory elements (29), but also of incomplete elements (6) and missed elements (8). The improvements, on the other hand, were made to inputs that were basically satisfactory, where the missing elements either did not add anything substantial or were compensated for by the inferences made (this happened mostly on inferences to connections). All of the improvements became elements judged to be satisfactory (so that 29 + 12 = 41, the number of OK-complete elements in Figure 17).

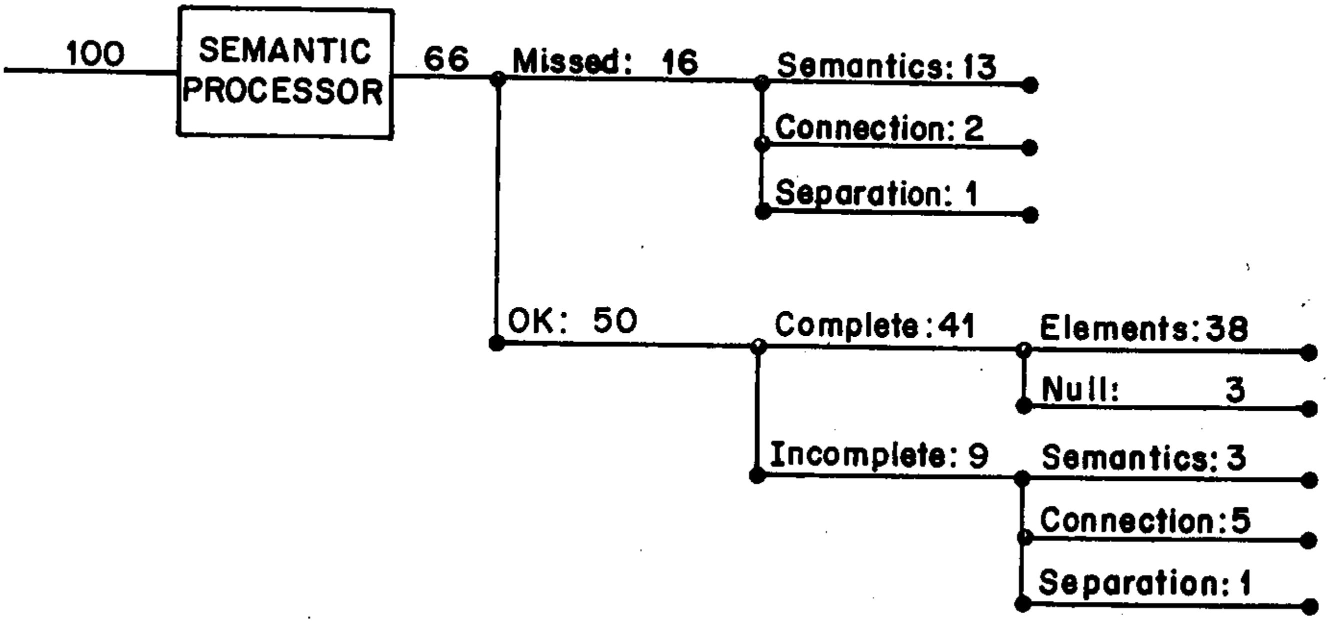


FIGURE 17. Performance of Semantic Processor on BI-100 for S3.

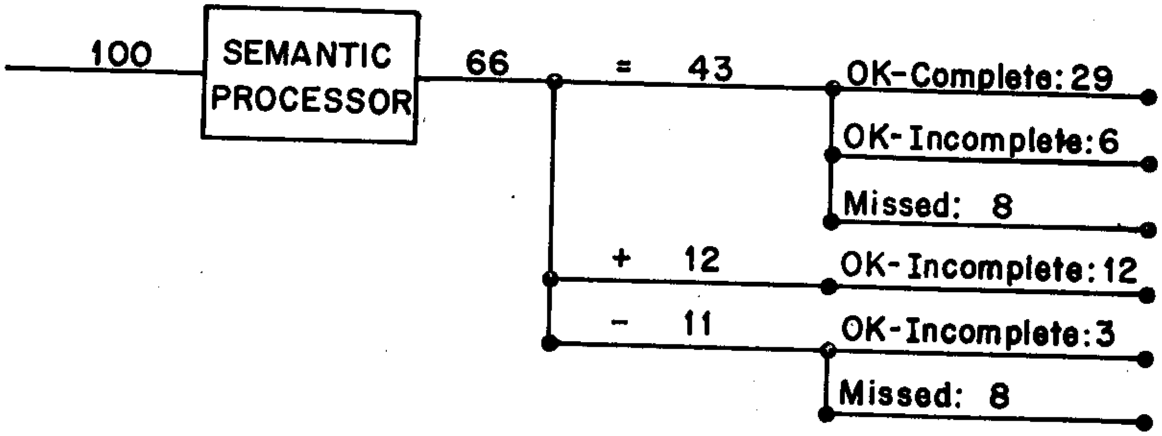


FIGURE 18. Changes Produced by Semantic Processor on BI-100 of S3.

Figures 16-18 are not meant to be definitive representations of the adequacy of the system. They are presented simply to give an indication of how the Linguistic and Semantic Processors are functioning as measured against direct judgments of what the linguistic input data permits.

Performance of the Determine Unknowns Mechanism. Unknowns are created primarily by the Linguistic Processor, and are passed on to later stages of the system which attempt to determine their proper values. For this problem of anaphoric reference, the sources of knowledge are not just the previous linguistic context, but the entire task environment. Table 10 summarizes the performance of PAS-I on the first 100 topic segments. There were 39 instances in which an unknown was created. Actually, unknowns sometimes get replaced with new unknowns of different scope (see B79-80), and these intermediate unknowns are not counted in the table. On the other hand, there are several places where essentially the same task of determining unknowns shows up repeatedly and each of these is counted separately (see especially B23-30).

Of the 39 total instances, there were 31 successful determinations and 8 failures. Of these 8, 3 were associated with one of the Linguistic Processor errors analysed earlier (at B71-73), 3 more were associated with failures to produce anything that showed up in the PBG, and 2 were associated with the analysis of B74-78. This latter involved the analysis of the effects of possible carries out of column 5 on the value of G(i.e., G = 1 or 2) and was all subsumed into a single node in the manual analysis, so that the failure to determine the unknows did not effect our overall evaluation of how well the PAS-I PBG matched the manual one.

	+(DU)	+(OM)	+(SPG)	+(tot)	-(none)	-(LP)	-(tot)	Total
*COL	3	2	0	5	0	0	0	5
*C	2	3	0	5	0	0	0	5
*D	3	1	0	4	3	1	4	8
*L	8	2	2	12	2	1	3	15
*X	5	0	0	5	1	0	1	6
Total	21	8	2	31	6	2	8	39

+(DU)	Correctly determined in Determine-Unknowns Mechanism
+(OM)	Correctly determined in Origin Mechanism
+(SPG)	Correctly determined in Selection of Protogroup Mechanism
-(none)	No value determined
-(LP)	Incorrect value determined due to error in Linguistic Processor

Table 10. PAS-I performance on determining unknowns for S3 B1-100 on D+G=R.

It can be seen from Table 10 that most of the determinations (21) are made in the Determine Unknows Mechanism. In a few cases (8), the Determine Unknows Mechanism is unable to determine the value, but it can be done by the Origin Mechanism hypothesizing an element. For instance, at B19-22 the Determine Unknowns Mechanism cannot find the value of the *C in:

```
(BECAUSEOF ((EQ *C 1)) ((ODD R)))
```

However, the Origin Mechanism hypothesizes (PC 2) with an input of (EQ C2 1), which essentially determines that the value of *C is C2.

In a very few cases (2) an unknown is determined by the mechanism that puts together the protogroup (called SPG in Table 10). An example occurs at B47:

```
(PC 6) from (EQC (PLUS 5 *L) 3)
```

When a protogroup is formed an attempt is made to infer the exact form of the operator. In the case above it was possible to determine that the operator was (PC 6), even though the source information had an unknown, *L. In effect, the value of the unknown was determined (to be G), though in fact the value is not needed for further processing and so is not actually symbolized internally. In general, the value of an unknown is determined exactly only when it is needed for further processing; often it is simply assimilated into the overriding inference being made.

The operator inference made by the protogroup selection mechanism gives rise to the possibility of intermediate unknowns as mentioned earlier. An example occurs at B79-80:

```
(PC *COL) from (EQC (PLUS *L *L) *D)
```

From (EQC ...) an inference is made that a PC occurred; since there is no information about the letters or digits (i.e., just *L's and *D's occur) the system generates simply (PC *COL), thereby creating a new unknown. Leter, the Origin Mechanism determines this to be (PC 5).

The ability of PAS-I to deal with anaphoric references, while undoubtedly capable of much improvement, does not appear to be a limiting factor in the current system. However, improvements will certainly be required as the scope of the system increases.

Summary. The behavior of PAS-I on the first 100 topic segments of S3 does indeed produce a PBG and one with quite reasonable features. It differs from the manually produced PBG for the same protocol in a number of ways. One striking feature is the diversity of these disparities, so that we have already reached the place where we must tune PAS-I up, bit by bit. Each new mechanism (or modification of an old one) will account for only a small improvement in the quality of the total performance. Where this is not the case (as in the missing notion of goals), it is because of an incompleteness that is deliberate at this stage of development.

Performance on B101-143 of S3

Appendix II presents the output of PAS-I on the next 43 topic segments of the protocol of S3 in the same manner as in Appendix I: Section II.1 gives the initial linguistic and semantic processing; section II.2 gives the trace of PAS-I as it selects protogroups, makes the appropriate inferences, and grows the PBG. We can conduct the same comparison of the behavior of PAS-I with the manual analysis as we did for the first 100 topic segments. Figure 19 gives the PBG of PAS-I, drawn from the output in the Appendix. Figure 20 gives the PBG from the manual analysis (Newell and Simon, 1972). It is a rather messy part of the protocol. S3 is concerned with whether E is 0 or 9, having forgotten that T is already 0. The trace is quite discontinuous as S3 drops back on several occasions to check and revise prior inferences (the three segments on the left side of Figure 20). The main part of the problem solving is concerned almost exclusively with columns 3 and 5, the two columns containing E, and thus there is ample opportunity for confusion even in the manual analysis.

Tables 11 and 12 summarize the comparison. They were derived in the same fashion as Tables 8 and 9, though we have omitted the detailed material from Appendix II. Table 11 gives the number of agreements and disagreements between the two PBG's. It confirms what is clear from a visual inspection of Figures 19 and 20: that performance is much more degraded than in the first 100 topic segments, especially with respect to the shapes of the PBG (the pattern of back-ups). Only five out of 30 items concerning nodes were in agreement, though another 9 had at least partial agreement, usually the operator. Also, only 1 out of 8 items concerning backups were in agreement. However, from our vantage point we are

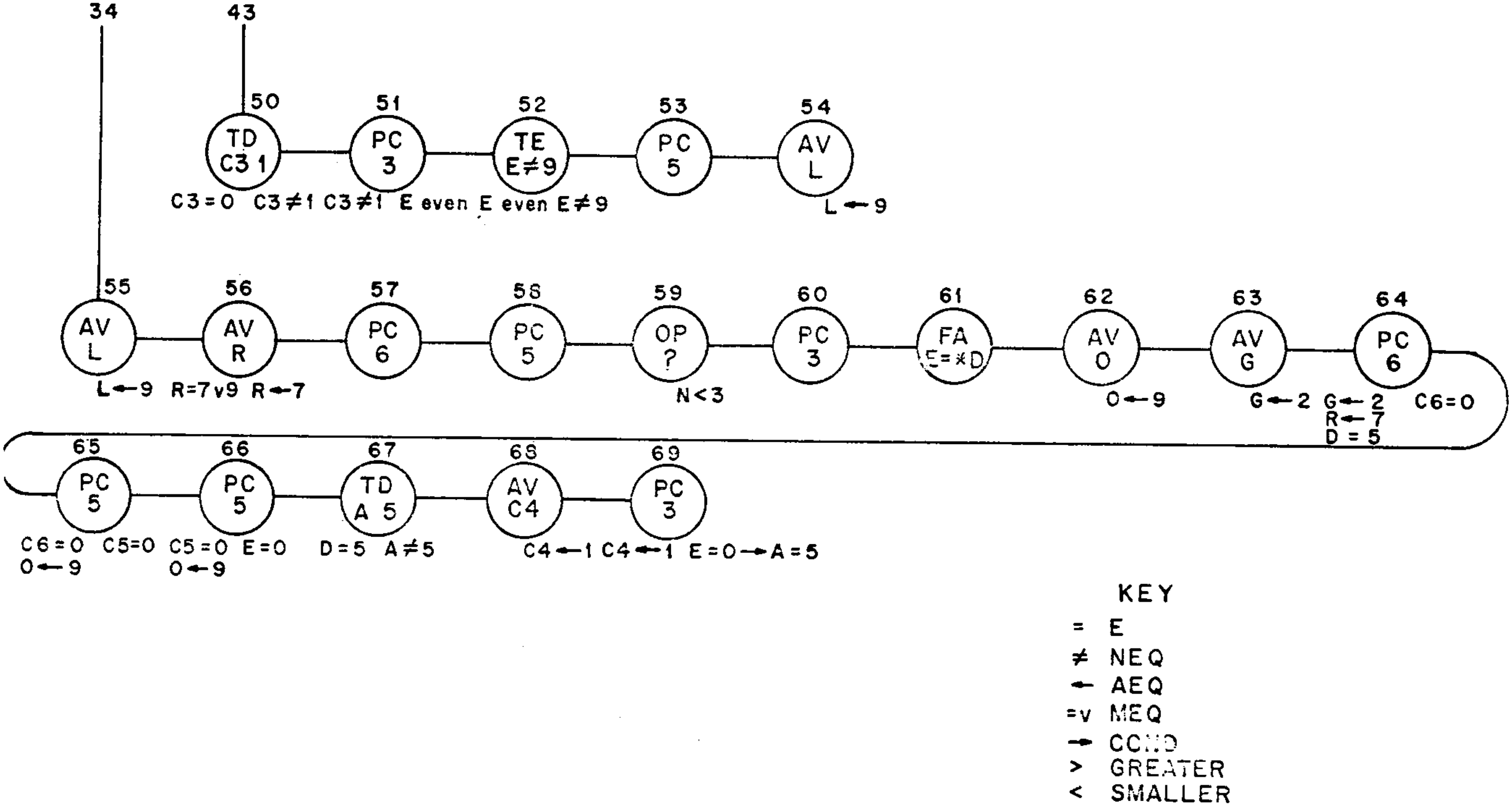


FIGURE 19. PBG by PAS-I for S3 on B100-143

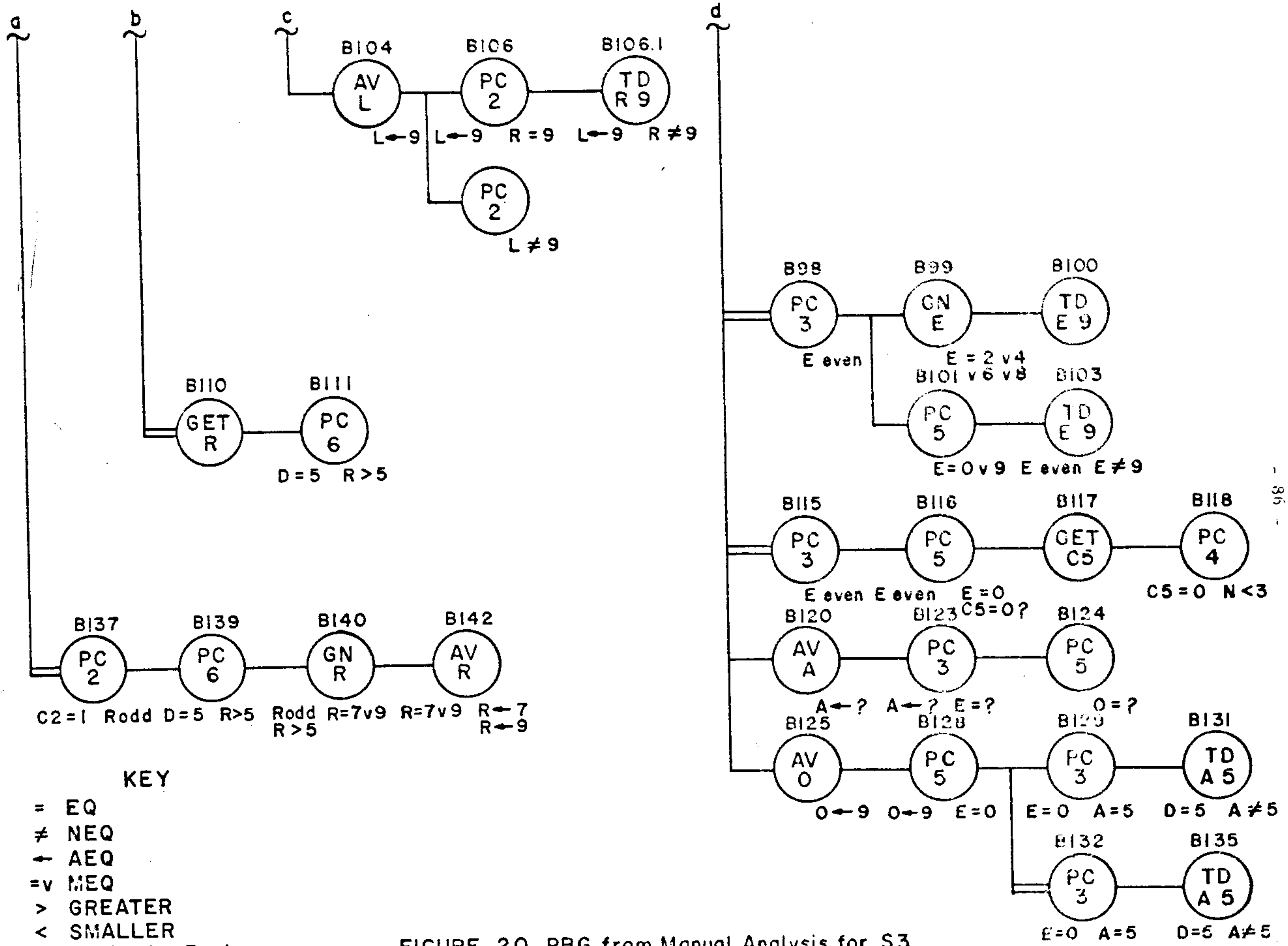


FIGURE 20. PBG from Manual Analysis for S3 on B100-143 (in PAS-I notation)

Types of Correspondence	Code	Number
Nodes correspond and agree	=n	5
PAS-I nodes subsumed in manual node	s	0
Nodes correspond, agree/disagree	=≠n	9
Nodes correspond and disagree	≠n	1
Extra PAS-I nodes	+pn	4
Extra Manual nodes	+mn	11
		30
Backups correspond and agree	=b	1
Backups correspond and agree/disagree	=≠b	0
Extra PAS-I backups	+pb	0
Extra Manual backups	+mb	7
		8
Cases excluded from consideration	x	0
		38

Table 11. Comparison between PBG of PAS-I and Manual Analysis for S3 B101-143 on D+G=R.

Deficiencies	Number of disparities produced
Linguistic Processor (LP)	
No output	6
Semantic Processor (SP)	0
Origin Mechanism (OM)	
Not seek origin of operator arguments	2
Not seek origin for partial elements	1
Weak inference program	7
Not work forward	4
Grow PBG Mechanism (GPBG)	
Merge two separate operators	7
Not use clue for merging: Redundant output	1
Not use clue for backup: Information unused	2
Not attend to order	1
Missing Concepts (-C)	
Goals	2
Repeating	5
Planning	4
Minor difficulties (Minor)	
Not expand conditional	1
	43

Table 12. PBG Disparities caused by PAS-I Deficiencies for PAS-I vs Manual Analysis for S3 B101-143 on D+G=R.

much more impressed that the program could operate at all with new material given that all the tuning occurred on the first 100 topic segments.*

The critical information is given in Table 12, which provides a list of the deficiencies in the various components of PAS-I that contribute to the disparities between the manual and automatically generated PBG's. As in Tables 8 and 9, the total number (43) shown in Table 12 is somewhat larger than the number of disparities (38) shown in Table 11, since there are several cases of multiple participation.

There is a single failure of the Linguistic Processor, again showing that PAS-I has strict limits on the subtlety of language it can handle. The difficult passage occurs at B106-109:

B106: RATHER --

B107: LET 'S SEE , HOW DID I ARRIVE AT THE POINT OF THAT ?

B108: THIS IS GOING TO BE A LITTLE CONFUSING TO START TRYING TO TRACE BACK HERE .

B109: WHAT 'S THE REASONING HERE ?

The manual analysis relies heavily on a reconstruction of what must have happened, given that the subject assigned (AEQ L 9) at B105 (which PAS-I also detects). In column 2, since there is a carry from column 1, 1 +L + L = R implies

[*] It should be clear that we are not conducting a test of the program, in the sense that psychologists use the term. PAS-I is still at an early stage of development and each application to a new protocol can be expected to turn up simple bugs, minor conceptual problems, and incomplete aspects. What counts at this stage is whether the program can be brought to run on the new material with only minor debugging and adaptation, and without invalidating the runs on prior material. This is what happened on the extension to B101-143 of S3 and to the protocol of S4 on DONALD+GERALD, discussed in the next section. Since the program had been conceived in complete interaction with the first 100 topic segments, it was conceivable that it had become over specialized. Thus, extension to new protocols was certainly of critical importance.

that R must also be 9 if L is 9. But this is not possible, hence L cannot be 9 -- but then R is not 9 either, thus removing the source of the contradiction. It is understandable that the subject could be confused, and hence the confusion exhibited in B106-109 above acts as a confirmation that some such processing must have occurred. The Linguistic Processor is simply unable to do anything with B106-109; hence there are no clues to support a processing inference such as the above. As a result, there is nothing for the Origin Mechanism to work with later on and PAS-I misses the entire episode.

Several different deficiencies of the Origin Mechanism contributed to the disparities between the two PBGs. Only one of these, the failure to seek the origin of operator arguments, also showed up in the first 100 segments. A second deficiency is that the Origin Mechanism does not attempt to discover the origin for any elements that are incomplete or partial. In the case in point (B122), the earlier stages produced (AEQ A *D) in response to a clear linguistic phrase: THAT IS, JUST ASSUME SOME VALUE FOR A. But, since the Unknowns Mechanism could not find a specific value for *D (indeed there was none to be found), the knowledge element was simply abandoned. PAS-I might have been able to do something with B120-123 if it had insisted on keeping this element and incorporating it into the PBG. Actually, it is unclear what the overall effects of such a change in PAS-I would be, since changing it to seek the origin of partial elements opens a pandora's box of other cases where the proper behavior is to ignore partial elements.

The largest category of deficiencies in the Origin Mechanism concerns weaknesses in its inference capability. These actually constitute four different situations, each with their own etiology (B115, B116, B126-129 and B132). The situation around B128 will suffice for an illustration. The program correctly gets that (PC 5) produces (EQ E 0) (ND66). This carries the implication both that (EQ C5 0) and (EQ C6 0). The Origin Mechanism considers only the possibility that (EQ C6 0) was determined prior to the (PC 5) under consideration and this leads it off on a fantasy about what happened at (PC 6) and (AV G). The Origin Mechanism is not yet able to consider that (EQ C6 0) was derived from (PC 5) along with (EQ E 0). It is able to consider that (EQ C6 0) is derived from (IG C6), as we have seen in other places, but has no way to prefer this hypothesis when there is another path of actual computation that yields the result.

The final type of deficiency in the Origin Mechanism is a general lack of working forward -- of saying "if such and such occurred what would probably have resulted." Indeed, describing the inference mechanism as an origin seeking mechanism indicates the one-sided nature of the present scheme. An example arises in connection with the B120-123 sequence just discussed. Even if the program kept (AEQ A *D) and sought its origin successfully as (AV A), it would probably need to work forward to get (PC 3) and then (PC 5). There is a clue in B123 that mentions E, but without some assumptions of using an assignment, one would not get to the processing of both columns 3 and 5. This reveals the difficulty in working forward: it involves predicting the future behavior from current behavior. This level of theory is not available in the assumption that the subject is working in a given problem space. The problem space dictates only the possibilities for forward movement, not which ones will occur. The more stringent constraints are available in the production system. Thus, to work forward with any generality implies developing and carrying forward a current estimate of the production system as well as the PBG. However, some minimal working-forward inference schemes that rest on obvious tenets of behavior could possibly be added to this stage of data analysis without evoking a complete production system.

A variety of deficiencies of the mechanism for growing the PBG were involved in the performance of Figure 19. Most of these are familiar from the first 100 topic segments: merging two separate operators into a single operator (either by merging or by recognition); not merging when it should; and not using deliberate clues for the existence of a backup. The overwhelming difficulty is over-merging, which is a sequence of the way S3 seesaws back and forth on columns 3 and 5, and drops back to repeat earlier sequences. The one new type of deficiency is the weakness in ordering the elements in the Origin List when building the PBG. The inference tree, as developed by the Origin Mechanism, puts some constraints on growing the tree: if the output of A is the input of B, then A must come before B. But there are no additional heuristics to determine an order beyond this. In the example that occurred (B101-102), PAS-I failed to connect the (PC 5) at ND53 with the (TE (NEQ E 9)) at ND52 due to not seeking the origin of operator arguments. However, it still wished to posit the (PC 5), which it simply put after the TE. It might have been able to decide that (PC 5) must have occurred before (TE...) on the basis of the general semantics of TE and PC.

Extension to topic segments B101-143 revealed two additional basic concepts not yet incorporated into PAS-I. (The lack of the concept of a goal also contributed here, as it did in the first 100 segments.) The really important missing concept was that of repeating a sequence of prior inferences in order to check them out. This is clearly what the subject is doing at three places in Figure 20. Not having this concept, PAS-I can only stumble around, recognizing old elements in the PBG and suppressing them. The second missing concept is that of a plan. This is the underlying difficulty at B120-123 (though we have used B120-123 to illustrate some other deficiencies). The manual analysis sees the subject as working forward at a symbolic level, not attempting to compute actual values, but only considering the general scheme of the inference steps. PAS-I has no notion of this and hence could not deal adequately with such behavior, even if the other deficiencies were corrected.

The only minor deficiency was the discovery (at B132) that the conditional element (COND...) does not get detached into two parts: a positing of the antecedent and the production of the consequent. This has a major impact on the total performance, since PAS-I cannot make further use of the knowledge elements, which are locked up inside the conditional expression. But this problem is not difficult to solve.

Performance on B1-128 of S4

To provide a third example of the performance of PAS-I, we ran it on a second subject, S4, using the same problem (DONALD+GERALD=ROBERT).* This protocol was analysed in Newell and Simon (1972), though no PBG was developed there. The subject is a relatively good problem solver, taking only 12 minutes to solve the problem and generate a protocol of 128 topic segments.** Three changes were made in the transcribed protocol where it seemed that either a transcription error was made or that the subject said one thing but meant something else.*** Appendix III gives the output of PAS-I on the entire protocol in the usual two parts:

* The program does, of course, work on other cryptarithmetic tasks. We have run it on the task CROSS+ROADS=DANGER (Newell and Simon, 1972, Chapter 7).

** Both the automatic and manual analyses were made on this protocol of 128 segments. Later we discovered that the actual protocol contained 130 segments, the missing segments being SO WE KNOW THAT R MUST BE AN ODD NUMBER and BECAUSE THE TWO D'S ADDED TO 10. These segments belonged between B65 and B66 in S4's protocol, and their omission led to (ODD R) being inferred in the wrong place in the manual analysis (see B71 in Figure 22).

*** G was changed to D in B16, A was changed to L in B56, and 5 plus 5 was changed to 4 plus 4 in B87.

Section III.1 shows the output of the Linguistic Processor and the first two stages of the Semantic Processor; section III.2 shows the basic cycle from protogroup formation through determining unknowns and finding origins to growing the PBG. Figures 21 and 22 give the PBG's of PAS-I and a manual analysis, the latter being done by one of us (AN) to form a basis for comparison. Table 13 gives a summary of the comparison of the two PBGs. We omit not only the detail of the comparison from the Appendix, but also the table showing the deficiencies of the particular parts of the program. The prior two tables (9 and 12) have shown the pattern of deficiencies. Many of the deficiencies already identified repeat themselves and some new ones appear. Nothing appears to be gained by further detailed enumeration.

Two things are important about this third run. The first is that PAS-I was able to carry out an analysis on a new protocol by a different subject with only a small amount of further debugging and adaptation. The second is to note what changes were required. The most extensive were additions to the grammar. Given the nature of the Linguistic Processor this was expected. The grammar is adapted rather closely to the individual subject and makes no pretense of being a general grammar; it does not even differentiate between the rules expected to be constant over subjects and those expected to be idiosyncratic. However, the additions to the grammar and problem space were not extensive, as Figure 23 shows. In addition, a slight rerouting in the top level of the grammar was required (recall Figure 6). Almost no other changes were made to PAS-I further down stream, except to add rules in the Semantic Processor to take care of the new elements provided by the grammar additions (EQR, MEQR, BETW, REMAIN). No changes were made to the basic inference parts of the program. Thus PAS-I on S4 is essentially the same program as was used on S3.

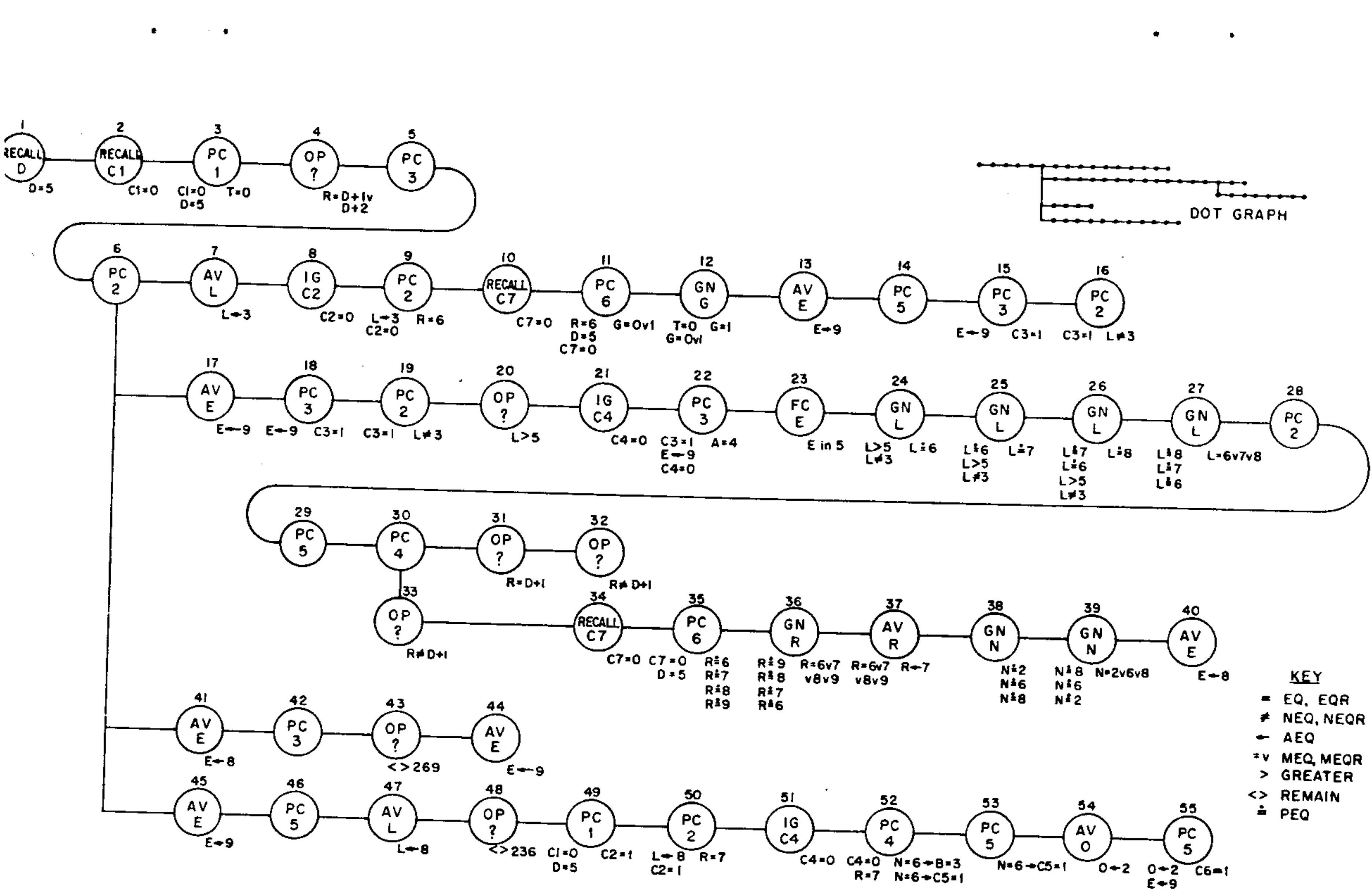


FIGURE 21. PBG by PAS I for S4 B1-128 on D+G=R

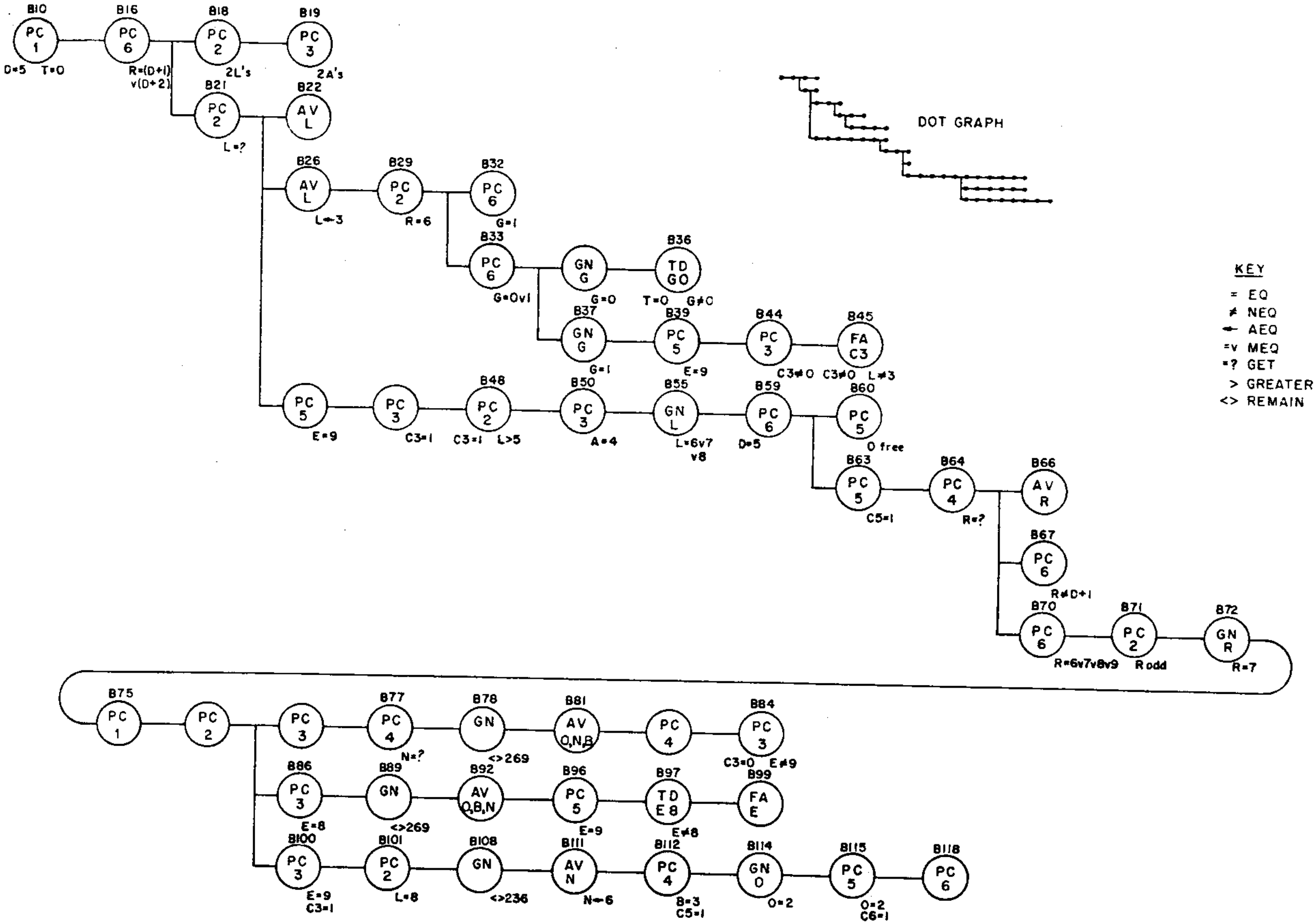


FIGURE 22. PBG by Manual Analysis for S4 B1-128 on D+G=R

Types of Correspondence	Code	Number
Nodes correspond and agree	=n	11
PAS-I nodes subsumed in manual node	s	12
Nodes correspond, agree/disagree	=≠n	9
Nodes correspond and disagree	≠n	10
Extra PAS-I nodes	+pn	7
Extra Manual nodes	+mn	21
		70
Backups correspond and agree	=b	4
Backups correspond and agree/disagree	=≠b	0
Extra PAS-I backups	+pb	0
Extra Manual backups	+mb	6
		10
Cases excluded from consideration	x	7
		87

Table 13. Comparison between PBG of PAS-I and Manual Analysis for S4 B1-128 on D+G=R.

```
* <MEQR>       := <LETTER> <EQUAL> <DIGIT> <DIGIT> <SIZE> <OPTLETTER>

* <EQR>        := <LETTER> <EQUAL> <DIGIT> <SIZE> <OPTLETTER>

* <BETW>       := <LETTER> <EQUAL> BETWEEN <DIG> <DIG>

* <REMAIN>     := REMAINING <DIG>

  <SIZE>       := <LARGE> | <SMALL>

  <OPTLETTER>  := <LETTER> | < >

  <LOCATION>   := ...... THIRD | FOURTH | FIFTH | SIXTH

* <ASSUME>     := ...... GUESS | AT-MOST

* <CARRY>      := ...... ADDING
```

(a). Grammar

KNOWLEDGE	MEANING
(MEQR u_1 (PLUS u_2 u_3) (PLUS u_4 u_5))	u_1 is u_2 plus u_3 or u_4 plus u_5
(EQR u_1 (PLUS u_2 u_3))	u_1 is u_2 plus u_3
(BETW l d_1 d_2)	l can have any value from d_1 through d_2
(REMAIN d_1 d_2 ...)	the unused digits are d_1, d_2, ...

(b). Semantic Elements

Figure 23. Additions to Grammar (see Figure 5) and Problem Space (see Table 1) needed to apply PAS-I to S4 on B1-128. In (a) the dotted lines indicate extensions to existing classes, and classes marked with an asterisk represent problem space elements. In (b) u stands for a letter, carry, or digit, l a letter, and d a digit.

VI. CONCLUSION

We have presented the initial results of a system for automatic protocol analysis, using a system called PAS-I. We view it as the first step toward a practical system the cognitive psychologist can use to analyse the detailed time course of a subject's developing knowledge as revealed in his concurrent verbalizations.

In our estimation, the results shown here provide a demonstration of the basic feasibility of the project for the task environment of cryptarithmetic. The success is due to some special circumstances, no doubt. The language of the subjects, though free, is quite simple. Where it does become complex the system has difficulties with it. The task itself, though posing genuine problems for our subjects, is simple enough so that incorporating its semantics into a computer program is possible with reasonable effort. Furthermore, the results are not yet of a quality to compete with manual analysis. They do approach it however, turning out an analysis that appears to be essentially correct over stretches of the protocol that are not too difficult to interpret manually. A correction of some of the simpler deficiencies and a method of resetting the state of knowledge, so that deficiencies do not propagate, would produce a system much closer in quality to manual analysis.

The program is partial and incomplete in several distinct ways, each worth noting. First, the program is task specific to cryptarithmetic. No program can be task independent, since knowledge of the task environment is one of the basic sources of knowledge that permits inferences to the subject's knowledge. Thus, to seek task independent analysis programs is chimerical. What can be sought are representations of task dependent knowledge that permit using the programs on different tasks. Although some of the task aspects in PAS-I were factored into rules, no attempt was made to do this systematically.

The second way the program is specialized is in the analysis task it performs: it does only the task of behavior description given, a complete representation of the problem space. The scientist working with a protocol is hardly limited to this one concern. As noted in Section III, he is also concerned with behavior description at the level of production systems as well as the PBG; with the induction of various structures that represent the human subject (linguistic mechanisms, the problem space and the production rules); with making detailed predictions given the structures and higher level behavioral descriptions; and with comparing and evaluating the structures and descriptions obtained over sets of subjects and sets of tasks. The selection of the task of behavior description for PAS-I was not made because it was the easiest of all of these, but because it contained a central kernel of the whole problem -- interpreting natural language and inferring the subject's knowledge state. All the other data analysis tasks have their own separate difficulties; none seems unapproachable given the success with PAS-I to date (see Waterman and Newell, 1971, for more discussion).

In a somewhat separate category is the extension of the program back toward the speech signal. The transcriptionist who listens to the audio tape and provides the lexical representation (and the punctuation for the topic segmentation) makes use of an analysis that duplicates that of PAS-I in indefinite ways. Ultimately one wants the automatic analysis to begin prior to transcription to remove that source of uncontrolled interpretation. The difficulty of doing speech recognition is well recognized and makes this aspect of extension heavily dependent on progress in a more basic artificial intelligence task.

The third incomplete aspect is the quality of the current analysis. As already noted, it is not the case that only one or two key features are deficient in PAS-I. Rather, an almost indefinite set of deficiencies contribute to the limited quality of its output. As the prior discussion indicated, these range from major to minor in terms of the amount of program modification and augmentation required to remove them. With every added protocol analysis a few additional deficiencies will come to light. This is hardly a situation to be deplored however. The development of a complex tool such as PAS-I is a long term accretion of continually more adequate inference and analysis mechanisms. The main tactical question is how long to continue working with a given program framework and when to recast the entire system in a more flexible and efficient form. A more strategic issue is whether there are some fundamental deficiencies of the approach that imply complete rethinking of how protocol analysis should be mechanized. With the limited experience to date we have not turned up any indications that our current approach needs extensive revision or rethinking.

The linguistic phase of PAS-I constitutes a special issue. As noted, it is deliberately simple and hardly matches the sophistication of current systems like Winograd (1971) or Woods (Woods and Kaplan, 1971). We are rather pleased with the performance of the linguistic phase of the system. Yet, it is clear that a much better grammar is required to work with the places where the Linguistic Processor failed. We do retain some skepticism, however, since the real issue in those missed linguistic phrases is extracting the semantics, and it is unclear that a more adequate parse takes us very far along that road. That is, the difficulties in handling the parts of the protocol that the Linguistic Processor missed starts with the Linguistic Processor, but does not end there. They constitute obscure parts of the protocol that provide difficulty for manual analysis as well.

The fourth way the program is incomplete is in its emphasis on providing a fully automatic analysis, rather than being developed as an interactive tool to be used by scientists. Our basic reason for this choice is to focus development on the central inference and interpretation problems, without being distracted by issues of communication and display. It is clear, however, that recasting into a form where PAS-I does what it can while the human user monitors the analysis, correcting and augmenting it, would provide an excellent tool. It would minimize the impact of an existing set of deficiencies, since the user would simply substitute his own recognitions, inferences,and decisions whenever required. The essence of objectivity would remain, since the parts of the analysis provided by rule and by human analysis could be labeled. Later conclusions could be traced back to determine their dependence on the evidence and to assess its status. Constructing such an interactive PAS does not present major technical problems, though it is not without interest as a computer science development.

The final specialization of the system is its dependence on a particular theory of human problem solving. PAS is not a general purpose, theoretically neutral, data analysis device. It is to be contrasted, for instance, with the General Inquirer (Stone, Dunphy, Smith, Olgivie, I966), where the conceptual commitments built into the program are essentially methodological. Thus, PAS does not provide a tool for protocol analyses based on different or absent theoretical conceptions. For its commitment PAS obtains a much more powerful analysis than would be possible otherwise. It must be admitted that the authors, having a substantial interest in the cognitive theory incorporated in PAS, designed the current version, PAS-I, for their own use in conneotion with the theory. Thus, we have little interest in moving the development in theoretically neutral ways.

PAS-I contains only limited aspects of the existing theory of human problem solving and that theory will, no doubt, continue to be developed. Thus, extensions to PAS will be required to incorporate additional knowledge from the theory. Some of these will occur naturally with the extension of the scope of PAS, e.g., the incorporation of production systems. Others, e.g., a theory of short and long term memory, could be added even to the present level of analysis.

Development of PAS in all of the ways indicated above appears promising, both in terms of payoff and in terms of feasibility. Effort cannot be applied in all directions at once, and it is not clear at this time which aspects will receive intensive attention.

REFERENCES

1. Bartlett, F. C., Thinking. New York: Basic Books, 1958.

2. Baylor, G. W., Program and protocol analysis on a mental imagery task, Proceedings of the Second International Joint Conference on Artificial Intelligence, The British Computer Society, 1971.

3. Bree, D., The Understanding Process as seen in Geometry Theorems, Unpublished PhD Dissertation, Carnegie-Mellon University, 1968.

4. Brookes, M., 150 Puzzles in Cryptarithmetic. New York: Dover, 1963.

5. DeGroot, A. D., Thought and Choice in Chess. The Hague: Mouton, 1965.

6. Dunker, K., On problem solving, Psychological Monographs, Whole No. 270, 1945.

7. Eastman, C., Cognitive Processes and ill-defined problems: A case study from design, in D. E. Walker and L. M. Norton (Eds.) Proceedings of the First International Joint Conference on Artificial Intelligence, 1969. Available from the Association for Computing Machinery.

8. Feldman, J., Simulation of behavior in the binary choice situation, in E. Feigenbaum and J. Feldman (Eds.) Computers and Thought. New York: McGraw-Hill, 1963.

9. Frijda, N. H. and Meertens, L. A., A simulation model of human information retrieval, in The Simulation of Human Behavior, Proceedings of a NATO Symposium, Paris, 1967. Paris: Dunod, 1969.

10. Humphrey, L., Thinking. New York: Wiley, 1951.

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

12. Laughery, K. R. and Gregg, L. W., Simulation of human problem-solving behavior, Psychometrika, 1962, 27, 265-282.

13. Newell, A., On the analysis of human problem solving protocols, in J. C. Gardin and B. Jaulin (Eds.) Calcul et Formalisation dans les Sciences de L'Homme. Paris: Centre National de la Recherche Scientifique, pp 146-185, 1968.

14. Newell, A., Studies in Problem Solving: Subject 3 on the cryptarithmetic task: DONALD+GERALD=ROBERT. Pittsburgh: Carnegie-Mellon University, 1967.

15. Newell, A. and Simon, H. A., An example of human chess play in the light of chess playing programs, in N. Weiner and J. P. Schade (Eds.) Progress in Biocybernetics, (Vol. 2). Amsterdam: Elsevier, 1965.

16. Newell, A. and Simon, H. A., GPS, a program that simulates human thought, in E. Feigenbaum and J. Feldman (Eds.) Computers and Thought, New York: McGraw-Hill, 1963.

17. Newell, A. and Simon, H. A., Computer simulation of human thinking, Science, 1961, 134, 2011-2017.

18. Newell, A. and Simon, H. A., Human Problem Solving. Englewood Cliffs: Prentice-Hall, 1972.

19. Paige, J. and Simon, H. A., Cognitive processes in solving algebra word problems, in B. Kleinmuntz (Ed.), Problem Solving. New York: Wiley, 1966.

20. Reitman, W., Cognition and Thought. New York, Wiley, 1965.

21. Stone, P. J., Dunphy, D. C., Smith, M. S., and Ogilvie, D. M., The General Inquirer. Cambridge, Mass: MIT Press, 1966.

22. Wagner, D. A. and Scurrah, M. J., Some characteristics of human problem solving in chess, Cognitive Psychology, 1971, 2, 454-478.

23. Waterman, D. A., Generalization learning techniques for automating the learning of heuristics, Artificial Intelligence, 1970, 1, 121-170.

24. Waterman, D. A., and Newell, A., Protocol analysis as a task for artificial intelligence, Artificial Intelligence, 1971, 2, 285-318.

25. Winikoff, A., Eye Movements as an Aid to Protocol Analysis of Problem Solving Behavior, Unpublished PhD Dissertation, Carnegie Institute of Technology, 1967.

26. Winograd, T., Understanding Natural Language, Cognitive Psychology, 1972, 3, 1-191.

27. Woods, W. A. and Kaplan, R. M., The Lunar Sciences Natural Language Information System, Report 2265, Cambridge, Mass: Bolt, Beranek and Newman, 1971.

APPENDIX I

I.1. PAS-I Linguistic and Semantic Processor Output for S3 B1-100 on D+G=R.

```
B1. EACH LETTER HAS ONE AND ONLY ONE NUMERICAL VALUE --
        (?)
B2. EXP : ONE NUMERICAL VALUE .
        (EXP)
B3. THERE ARE TEN DIFFERENT LETTERS
        (?)
B4. AND EACH OF THEM HAS ONE NUMERICAL VALUE .
        (AND)

      <B1-4>. ((?))


B5. THEREFORE , I CAN , LOOKING AT THE TWO D 'S --
        (THEREFORE) (NUM D 2)

        <B5>. ((THEREFORE) (NUM D 2))


B6. EACH D IS 5 ;
        (EQ D 5)
B7. THEREFORE , T IS ZERO .
        (THEREFORE) (EQ T 0)

      <B6-7>. ((BECAUSEOF ((EQ D 5)) ((EQ T 0))))


B8. SO I THINK I 'LL START BY WRITING THAT PROBLEM HERE .
        (THEREFORE)

        <B8>. ((?))


B9. I 'LL WRITE 5 , 5 IS ZERO .
        (EQC (PLUS 5 5) 0)

        <B9>. ((EQC (PLUS 5 5) 0))


B10. NOW , DO I HAVE ANY OTHER T 'S ?
        (FC T)

       <B10>. ((FC T))


B11. NO .
        (NEG)

       <B11>. ((?))


B12. BUT I HAVE ANOTHER D .
        (IN D *COL)

       <B12>. ((IN D *COL))


B13. THAT MEANS I HAVE A' 5 OVER THE OTHER SIDE .
        (THEN) (IN 5 *COL) (PLACE OTHER)

       <B13>. ((THEN) (IN 5 *COL))
```

```
B14. NOW I HAVE 2 A 'S
       (NUM A 2)

        <B14>. ((NUM A 2))


B15. AND 2 L 'S
       (AND) (NUM L 2)

        <B15>. ((AND) (NUM L 2))


B16. THAT ARE EACH --
       (?)
B17. SOMEWHERE --
       (?)
B18. AND THIS R --
       (AND) (LETTER R)

     <B16-18>. ((?))


B19. 3 R 'S --
       (NUM R 3)

        <B19>. ((NUM R 3))


B20. 2 L 'S EQUAL AN R --
       (EQC (PLUS L L) R)

        <B20>. ((EQC (PLUS L L) R))


B21. OF COURSE I 'M CARRYING A' 1 .
       (EQ *C 1)
B22. WHICH WILL MEAN THAT R HAS TO BE AN ODD NUMBER .
       (THEREFORE) (ODD R)

     <B21-22>. ((BECAUSEOF ((EQ *C 1)) ((ODD R))))


B23. BECAUSE THE 2 L 'S --
       (BECAUSE) (NUM L 2)

        <B23>. ((BECAUSE) (NUM L 2))


B24. ANY TWO NUMBER 'S ADDED TOGETHER HAS TO BE AN EVEN NUMBER
       (EVEN (PLUS *D *D))

        <B24>. ((EVEN (PLUS *D *D)))


B25. AND 1 WILL BE AN ODD NUMBER .
       (ODD (PLUS *X 1))
B26. SO R CAN BE 1 ,
       (THEREFORE) (PEQ R 1)
B27. 3 ,
       (DIGIT 3)
B28. NOT 5 ,
       (NEG) (DIGIT 5)
B29. 7 ,
          (DIGIT 7)
 B30. OR 9 .
          (OR) (DIGIT 9)

      <B25-30>. ((BECAUSEOF ((ODD (PLUS *X 1))) ((PEQ R 1))))

      <B25-30>. ((BECAUSEOF ((ODD (PLUS *X 1))) ((PEQ R 3))))

      <B25-30>. ((BECAUSEOF ((ODD (PLUS *X 1))) ((PEQ R 5))))

      <B25-30>. ((NEQ R 5))

      <B25-30>. ((BECAUSEOF ((ODD (PLUS *X 1))) ((PEQ R 7))))

      <B25-30>. ((BECAUSEOF ((ODD (PLUS *X 1))) ((PEQ R 9))))

      <B25-30>. ((MEQ R 1 3 7 9))


 B31. EXP : WHAT ARE YOU THINKING NOW ?
          (EXP)
 B32. NOW G --
          (LETTER G)

      <B31-32>. ((?))


 B33. SINCE R IS GOING TO BE AN ODD NUMBER
          (BECAUSE) (ODD R)
 B34. AND D IS 5 ,
          (AND) (EQ D 5)
 B35. G HAS TO BE AN EVEN NUMBER .
          (EVEN G)

      <B33-35>. ((BECAUSEOF (AND ((ODD R)) ((EQ D 5))) ((EVEN G))))


 B36. I 'M LOOKING AT THE LEFT SIDE OF THIS PROBLEM HERE WHERE IT SAYS D + G .
          (PLUS D G) (PLACE LEFT)

         <B36>. ((PLUS D G))


 B37. OH , PLUS POSSIBLY ANOTHER NUMBER ,
          (PLS *D)
 B38. IF I HAVE TO CARRY 1 FROM THE E + O .
          (IF) (COUT 1 (PLUS E O))

      <B37-38>. ((COND ((EQ C6 1)) ((EQ *C *D))))


 B39. I THINK I 'LL FORGET ABOUT THAT FOR A' MINUTE .
          (?)
 B40. POSSIBLY THE BEST WAY TO GET TO THIS PROBLEM IS TO TRY DIFFERENT POSSIBLE SOLUTIONS .
          (?)
 B41. I 'M NOT SURE WHETHER THAT WOULD BE THE EASIEST WAY OR NOT .
          (NEG) (NEG)
 B42. WELL , IF WE ASSUME --
          (IF) (ASSUME)

      <B39-42>. ((?))


 B43. IF WE ASSUME THAT L IS , SAY , 1 ,
          (IF) (ASSUME) (EQ L 1)
```

```
B44. WE 'LL HAVE 1 + 1 THAT 'S 3 OR R --
          (PLUS 1 1) (EQ R 3)
B45. WE 'LL PUT IN A' 3 HERE ,
          (DIGIT 3)

     <B43-45>. ((PLUS 1 1) (BECAUSEOF ((AEQ L 1)) ((EQ R 3))))


B46. AND ONE HERE .
          (AND)

        <B46>. ((?))


B47. WELL , 5 PLUS SOMETHING HAS TO EQUAL 3 IN THAT CASE --
          (THEREFORE) (EQC (PLUS 5 *L) 3)

        <B47>. ((THEREFORE) (EQC (PLUS 5 *L) 3))


B48. I SUPPOSE IT 'S --
          (IF)
B49. WELL , NOT , --
          (NEG)

     <B48-49>. ((?))


B50. IT 'S NOT POSSIBLE THAT THERE COULD BE ANOTHER LETTER IN FRONT OF THIS R IS IT ?
          (NEG) (PEQ *L *D) (LETTER R) (QUES)

        <B50>. ((NEQ *L *D) (LETTER R) (QUES))


B51. IS IT OR NOT ?
          (NEG) (QUES)
B52. EXP : NO .
          (EXP)
B53. IT 'S NOT --
          (NEG)
B54. ALL RIGHT --
          (?)
B55. SO IF --
          (THEREFORE) (IF)

     <B51-55>. ((?))


B56. IF THAT COULD N'T BE A' 13 ON THE LEFT SIDE .
          (IF) (NEG) (PEQ *L 13) (PLACE LEFT)
B57. THEN R CANNOT BE 3 .
          (THEN) (NEG) (EQ R 3)

     <B56-57>. ((OPIO (FN ((EQ *L 13))) ((NEQ *L 13)) ((NEQ R 3))))


B58. R HAS TO BE A' NUMBER GREATER THAN 5 ,
          (GREATER R 5)
B59. WHICH MEANS THAT IT CAN BE EITHER 7
          (THEREFORE) (PEQ *L 7)
B60. OR 9 .
          (OR) (DIGIT 9)
B61. SO WE 'LL START BACK HERE AND MAKE IT A' 7 .
          (THEREFORE) (ASSUME) (EQ *L 7)
```

```
      <B58-61>. ((BECAUSEOF ((GREATER R 5)) ((PEQ *L 7))))

      <B58-61>. ((BECAUSEOF ((GREATER R 5)) ((PEQ *L 9))))

      <B58-61>. ((MEQ *L 7 9))

      <B58-61>. ((BECAUSEOF ((MEQ *L 7 9)) ((AEQ *L 7))))


B62. NOW IF THE --
        (IF)
B63. OH , I 'M SORRY , I SAID SOMETHING INCORRECT HERE .
        (CORRECTION)
B64. I 'M MAKING --
        (ASSUME)
B65. NO , NO , I DID N'T EITHER .
        (NEG) (NEG) (NEG)

      <B62-65>. ((?))


B66. R IS GOING TO BE A' 7 ,
        (EQ R 7)
B67. THEN THIS WILL BE 7 ,
        (THEN) (EQ *L 7)
B68. AND THAT WILL BE 7 ,
        (AND) (EQ *L 7)
B69. AND IT 'S THE L 'S THAT WILL HAVE TO BE 3 'S ,
        (AND) (EQ L 3)

      <B66-69>. ((BECAUSEOF ((EQ R 7)) (AND ((EQ *L 7)) (AND ((EQ *L 7)) ((EQ L 3))))))


B70. BECAUSE 3 + 3 IS 6
        (BECAUSE) (EQC (PLUS 3 3) 6)

         <B70>. ((BECAUSE) (EQC (PLUS 3 3) 6))


B71. + 1 IS 7 .
        (EQC (PLUS *X 1) 7)

         <B71>. ((EQC (PLUS *X 1) 7))


B72. NOW , IT DOES N'T MATTER ANYWHERE WHAT THE L 'S ARE EQUAL TO --
        (NEG) (EQ L *D)
B73. SO I 'M INDEPENDENT OF L WHEN I GET PAST THE SECOND COLUMN HERE .
        (THEREFORE) (LETTER L) (PLACE SECOND)

      <B72-73>. ((BECAUSEOF ((NEQ L *D)) ((EQ L *D))))


B74. BUT NOW I KNOW THAT G HAS TO BE EITHER 1
        (EQ G 1)
B75. OR 2 ,
        (OR) (DIGIT 2)
B76. DEPENDING ON WHETHER OR NOT E + O IS GREATER THAN 10
        (WNOT) (GREATER (PLUS E O) 10)
B77. OR GREATER THAN 9 .
        (OR) (GREATER *L 9)

      <B74-77>. ((PLUS E O) (GREATER (PLUS E O) 10) (COND ((GREATER (PLUS E O) 10)) ((EQ G 1))))

      <B74-77>. ((COND ((NGREATER (PLUS E O) 10)) ((EQ G 2))))
      <B74-77>. ((COND ((GREATER *L 9)) ((EQ G 1))))

      <B74-77>. ((COND ((NGREATER *L 9)) ((EQ G 2))))


B78. NOW I HAVE THIS O REPEATING HERE IN THE SECOND COLUMN FROM THE LEFT ;
        (IN O *COL) (PLACE SECOND) (PLACE LEFT)

        <B78>. ((IN O *COL))


B79. THAT IS , ITSELF PLUS ANOTHER NUMBER EQUAL TO ITSELF .
        (EQC (PLUS *L *L) *D)

        <B79>. ((EQC (PLUS *L *L) *D))


B80. THIS MIGHT INDICATE THAT E WAS ZERO --
        (THEREFORE) (PEQ E 0)

        <B80>. ((THEREFORE) (PEQ E 0))


B81. IN FACT , IT MIGHT HAVE TO NECESSARILY INDICATE THAT .
        (?)
B82. I 'M NOT SURE .
        (NEG)

      <B81-82>. ((?))


B83. OR , E COULD BE 9
        (OR) (PEQ E 9)

        <B83>. ((PEQ E 9))


B84. AND I WOULD BE CARRYING 1 ,
        (AND) (EQ *C 1)
B85. WHICH WOULD MEAN THAT I WAS THEN CARRYING 1 INTO THE LEFT HAND COLUMN .
        (THEREFORE) (THEN) (EQ *C 1) (PLACE LEFT)

      <B83-85>. ((BECAUSEOF (AND ((EQ E 9)) ((EQ *C 1))) ((EQ *C 1))))


B86. EXP : WHAT ARE YOU THINKING NOW ?
        (EXP)
B87. I WAS JUST TRYING TO THINK OVER WHAT I WAS JUST --
        (?)
B88. ABOUT THE POSSIBILITY --
        (?)

      <B86-88>. ((?))


B89. THE IMPLICATIONS OF AN O + ANOTHER NUMBER EQUALING AN O ,
        (EQC (PLUS O *L) O)

        <B89>. ((EQC (PLUS O *L) O))


B90. AND WHAT THAT NECESSARILY IMPLIES .
        (AND) (THEREFORE)
B91. LET 'S SEE --
```

```
          (?)

      <B90-91>. ((?))


  B92. I HAVE TWO A 'S EQUALING AN E .
          (EQC (PLUS A A) E)

         <B92>. ((EQC (PLUS A A) E))


  B93. THEREFORE , E HAS TO BE AN EVEN NUMBER ,
          (THEREFORE) (EVEN E)
  B94. BECAUSE I KNOW I 'M NOT CARRYING 1 .
          (BECAUSE) (NEG) (EQ *C 1)

      <B93-94>. ((BECAUSEOF ((NEQ *C 1)) ((EVEN E))))


  B95. OF COURSE , THIS ALL GOING ON THE ASSUMPTION THAT R IS 7 --
          (ASSUME) (EQ R 7)

         <B95>. ((AEQ R 7))


  B96. R COULD BE 9 ALSO .
          (PEQ R 9)

         <B96>. ((PEQ R 9))


  B97. WELL , MAYBE I 'LL JUST CONTINUE TO TRY TO WORK THIS THROUGH AGAIN .
          (?)

         <B97>. ((?))


  B98. IF E HAS GOT TO BE AN --
          (IF) (EQ E *D)

         <B98>. ((IF) (EQ E *D))


  B99. NOW , WAIT A' SECOND .
          (?)
 B100. I GOT SOMETHING OUT OF THIS .
          (?)

     <B99-100>. ((?))
```

I.2. PAS-I Group Processor and PBG Mechanism Output for S3 B1-100 on D+G=R.

```
B5-7 PROTOGROUP

ELEMENTS : ( ((THEREFORE))
             ((BECAUSEOF ((EQ D 5)) ((EQ T 0)))) )
OPERATOR : (PC 1)   FROM (NUM D 2)

    ORIGIN LIST : ()

    ORIGIN LIST : ( ((RECALL D) NIL (EQ D 5))
                    ((RECALL C1) NIL (EQ C1 0))
                    ((PC 1) ((EQ D 5) (EQ C1 0)) (EQ T 0)) )

        (RECALL D) GROWN AS ND1
        (EQ D 5) GROWN AS K1

        (RECALL C1) GROWN AS ND2
        (EQ C1 0) GROWN AS K2

        (PC 1) GROWN AS ND3
        (EQ T 0) GROWN AS K3


B9 PROTOGROUP

ELEMENTS : ()
OPERATOR : (PC 1)   FROM (EQC (PLUS 5 5) 0)

        ((PC 1) NIL NIL) MERGED WITH ND3


B10 PROTOGROUP

ELEMENTS : ()
OPERATOR : (FC T)   FROM (FC T)

        (FC T) GROWN AS ND4


B12-14 PROTOGROUP

ELEMENTS : ( ((IN D *COL))
             ((THEN) (IN 5 *COL)) )
OPERATOR : (PC 3)   FROM (NUM A 2)

    ((IN D *COL)) IS ((IN D 6))

    ORIGIN LIST : ( ((FC D H) NIL (IN D 6)) )

        (FC D) GROWN AS ND5
        (IN D 6) GROWN AS K4

    ((THEN) (IN 5 *COL)) IS ((THEN) (IN D 6))

    ORIGIN LIST : ( (ND5 NIL (IN D 6)) )

        ((FC D) NIL (IN D 6)) RECOGNIZED AS ND5

        (COUNT A) GROWN AS ND6
        (NUM A 2) GROWN AS K5
```

```
B15 PROTOGROUP

ELEMENTS : ( ((AND)) )
OPERATOR : (PC 2)   FROM (NUM L 2)

    ORIGIN LIST : ()

        (COUNT L) GROWN AS ND7
        (NUM L 2) GROWN AS K6


B19-22 PROTOGROUP

ELEMENTS : ( ((NUM R 3))
             ((BECAUSEOF ((EQ *C 1)) ((ODD R)))) )
OPERATOR : (PC 2)   FROM (EQC (PLUS L L) R)

    ORIGIN LIST : ( ((COUNT R H) NIL (NUM R 3)) )

        (COUNT R) GROWN AS ND8
        (NUM R 3) GROWN AS K7

    ((BECAUSEOF ((EQ *C 1)) ((ODD R)))) IS
    ((BECAUSEOF ((EQ *C 1)) ((ODD R))))

    ORIGIN LIST : ( (ND1 NIL (EQ D 5))
                    (ND2 NIL (EQ C1 0))
                    ((PC 1 H) ((EQ D 5) (EQ C1 0)) (EQ C2 1 H))
                    ((PC 2) ((EQ C2 1 H)) (ODD R)) )

        ((RECALL D) NIL (EQ D 5)) RECOGNIZED AS ND1

        ((RECALL C1) NIL (EQ C1 0)) RECOGNIZED AS ND2

        ((PC 1 H) ((EQ D 5) (EQ C1 0)) (EQ C2 1 H)) MERGED WITH ND3
        (EQ C2 1) GROWN AS K8

        (PC 2) GROWN AS ND9
        (ODD R) GROWN AS K9


B23-30 PROTOGROUP

ELEMENTS : ( ((BECAUSE))
             ((EVEN (PLUS *D *D)))
             ((BECAUSEOF ((ODD (PLUS *X 1))) ((PEQ R 1))))
             ((BECAUSEOF ((ODD (PLUS *X 1))) ((PEQ R 3))))
             ((BECAUSEOF ((ODD (PLUS *X 1))) ((PEQ R 5))))
             ((NEQ R 5))
             ((BECAUSEOF ((ODD (PLUS *X 1))) ((PEQ R 7))))
             ((BECAUSEOF ((ODD (PLUS *X 1))) ((PEQ R 9))))
             ((MEQ R 1 3 7 9)) )
OPERATOR : (PC 2)   FROM (NUM L 2)

    ORIGIN LIST : ()

    ((EVEN (PLUS *D *D))) IS ((EVEN (PLUS L L)))

    ORIGIN LIST : ( ((PC 2) ((EQ C2 1)) (EVEN (PLUS L L))) )

        ((PC 2) ((EQ C2 1)) (EVEN (PLUS L L))) MERGED WITH ND9
        (EVEN (PLUS L L)) GROWN AS K10

    ((BECAUSEOF ((ODD (PLUS *X 1))) ((PEQ R 1)))) IS
    ((BECAUSEOF ((ODD R)) ((PEQ R 1))))

    ORIGIN LIST : ( ((GN R H) ((ODD R)) (PEQ R 1)) )

        (GN R) GROWN AS ND1Ø
        (PEQ R 1) GROWN AS K11

    ((BECAUSEOF ((ODD (PLUS *X 1))) ((PEQ R 3)))) IS
    ((BECAUSEOF ((ODD R)) ((PEQ R 3))))

    ORIGIN LIST : ( ((GN R H) ((ODD R) (PEQ R 1)) (PEQ R 3)) )

        (GN R) GROWN AS ND11
        (PEQ R 3) GROWN AS K12

    ((BECAUSEOF ((ODD (PLUS *X 1))) ((PEQ R 5)))) IS
    ((BECAUSEOF ((ODD R)) ((PEQ R 5))))

    ORIGIN LIST : ( ((GN R H) ((ODD R) (PEQ R 1) (PEQ R 3)) (PEQ R 5)) )

        (GN R) GROWN AS ND12
        (PEQ R 5) GROWN AS K13

    ORIGIN LIST : ( (ND1 NIL (EQ D 5))
                    ((TD R 5 H) ((EQ D 5)) (NEQ R 5)) )

        ((RECALL D) NIL (EQ D 5)) RECOGNIZED AS ND1

        (TD R 5) GROWN AS ND13
        (NEQ R 5) GROWN AS K14

        PBG CONFLICT : (NEQ R 5) VS (PEQ R 5)

    ((BECAUSEOF ((ODD (PLUS *X 1))) ((PEQ R 7)))) IS
    ((BECAUSEOF ((ODD R)) ((PEQ R 7))))

    ORIGIN LIST : ( ((GN R H) ((ODD R) (NEQ R 5) (PEQ R 1) (PEQ R 3)) (PEQ R 7)) )

        (GN R) GROWN AS ND15
        (PEQ R 7) GROWN AS K15

    ((BECAUSEOF ((ODD (PLUS *X 1))) ((PEQ R 9)))) IS
    ((BECAUSEOF ((ODD R)) ((PEQ R 9))))

    ORIGIN LIST : ( ((GN R H) ((ODD R) (NEQ R 5) (PEQ R 1) (PEQ R 3) (PEQ R 7)) (PEQ R 9)) )

        (GN R) GROWN AS ND16
        (PEQ R 9) GROWN AS K16

    ORIGIN LIST : ( ((GN R H) ((PEQ R 1) (PEQ R 3) (PEQ R 7) (PEQ R 9)) (MEQ R 1 3 7 9)) )

        (GN R) GROWN AS ND17
        (MEQ R 1 3 7 9) GROWN AS K17


B33-38 PROTOGROUP

ELEMENTS : ( ((BECAUSEOF (AND ((ODD R)) ((EQ D 5))) ((EVEN G))))
             ((COND ((EQ C6 1)) ((EQ *C *D)))) )
OPERATOR : (PC 6)   FROM (PLUS D G)

    ORIGIN LIST : ( ((IG C6 H) NIL (EQ C6 Ø H))
                    (ND1 NIL (EQ D 5))
                    ((PC 6) ((EQ C6 Ø H) (EQ D 5) (ODD R)) (EVEN G)) )
        (IG C6) GROWN AS ND18
        (EQ C6 0) GROWN AS K18

        ((RECALL D) NIL (EQ D 5)) RECOGNIZED AS ND1

        (PC 6) GROWN AS ND19
        (EVEN G) GROWN AS K19

    ((COND ((EQ C6 1)) ((EQ *C *D)))) IS
    ((COND ((EQ C6 1)) ((EQ C6 1))))

    ORIGIN LIST : ( ((NOTICE C6 H) NIL (PEQ C6 1)) )

        (NOTICE C6) GROWN AS ND20
        (PEQ C6 1) GROWN AS K20

        PBG CONFLICT : (PEQ C6 1) VS (EQ C6 0)


B43-45 PROTOGROUP

ELEMENTS : ( ((BECAUSEOF ((AEQ L 1)) ((EQ R 3)))) )
OPERATOR : (PC *COL)   FROM (PLUS 1 1)

    ORIGIN LIST : ( ((AV L H) NIL (AEQ L 1))
                    ((PC 2 H) ((AEQ L 1) (EQ C2 1)) (EQ R 3)) )

        (AV L) GROWN AS ND22
        (AEQ L 1) GROWN AS K21

        (PC 2) GROWN AS ND23
        (EQ R 3) GROWN AS K22


B47 PROTOGROUP

ELEMENTS : ( ((THEREFORE)) )
OPERATOR : (PC 6)   FROM (EQC (PLUS 5 *L) 3)

    ORIGIN LIST : ()

        (PC 6) GROWN AS ND24


B50 PROTOGROUP

ELEMENTS : ( ((NEQ *L *D) (LETTER R) (QUES)) )
OPERATOR : ()

    ((NEQ *L *D) (LETTER R) (QUES)) IS ((NEQ D *D) (LETTER R) (QUES))

    ORIGIN LIST : ()


B56-61 PROTOGROUP

ELEMENTS : ( ((OPIO (FN ((EQ *L 13))) ((NEQ *L 13)) ((NEQ R 3))))
             ((BECAUSEOF ((GREATER R 5)) ((PEQ *L 7))))
             ((BECAUSEOF ((GREATER R 5)) ((PEQ *L 9))))
             ((MEQ *L 7 9))
             ((BECAUSEOF ((MEQ *L 7 9)) ((AEQ *L 7)))) )
OPERATOR : ()

    ((OPIO (FN ((EQ *L 13))) ((NEQ *L 13)) ((NEQ R 3)))) IS
    ((OPIO (FN (EQ C7 1)) ((NEQ C7 1)) ((NEQ R 3))))
```

```
ORIGIN LIST : ( ((RECALL C7) NIL (EQ C7 0))
                ((TD C7 1 H) ((EQ C7 0)) (NEQ C7 1))
                ((FN (EQ C7 1)) ((NEQ C7 1)) (NEQ R 3)) )

    (RECALL C7) GROWN AS ND25
    (EQ C7 0) GROWN AS K23

    (TD C7 1) GROWN AS ND26
    (NEQ C7 1) GROWN AS K24

    (FN (EQ C7 1)) GROWN AS ND27
    (NEQ R 3) GROWN AS K25

    PBG CONFLICT : (NEQ R 3) VS (EQ R 3)

    PBG CONFLICT : (NEQ R 3) VS (PEQ R 3)

((BECAUSEOF ((GREATER R 5)) ((PEQ *L 7)))) IS
((BECAUSEOF ((GREATER R 5)) ((PEQ R 7))))

ORIGIN LIST : ( ((RECALL C7) NIL (EQ C7 0))
                (ND1 NIL (EQ D 5))
                ((PC 6 H) ((EQ C7 0) (EQ D 5)) (GREATER R 5))
                ((GN R H) ((GREATER R 5) (ODD R)) (PEQ R 7)) )

    (RECALL C7) GROWN AS ND28
    (EQ C7 0) GROWN AS K26

    ((RECALL D) NIL (EQ D 5)) RECOGNIZED AS ND1

    (PC 6) GROWN AS ND29
    (GREATER R 5) GROWN AS K27

    PBG CONFLICT : (GREATER R 5) VS (PEQ R 1)

    (GN R) GROWN AS ND32
    (PEQ R 7) GROWN AS K28

((BECAUSEOF ((GREATER R 5)) ((PEQ *L 9)))) IS
((BECAUSEOF ((GREATER R 5)) ((PEQ R 9))))

ORIGIN LIST : ( ((GN R H) ((ODD R) (GREATER R 5) (PEQ R 7)) (PEQ R 9)) )

    (GN R) GROWN AS ND33
    (PEQ R 9) GROWN AS K29

((MEQ *L 7 9)) IS ((MEQ R 7 9))

ORIGIN LIST : ( ((GN R H) ((PEQ R 7) (PEQ R 9)) (MEQ R 7 9)) )

    (GN R) GROWN AS ND34
    (MEQ R 7 9) GROWN AS K30

((BECAUSEOF ((MEQ *L 7 9)) ((AEQ *L 7)))) IS
((BECAUSEOF ((MEQ R 7 9)) ((AEQ R 7))))

ORIGIN LIST : ( ((AV R H) ((MEQ R 7 9)) (AEQ R 7)) )

    (AV R) GROWN AS ND35
    (AEQ R 7) GROWN AS K31
```

```
ELEMENTS : ( ((BECAUSEOF ((EQ R 7)) ((EQ L 3))))
             ((BECAUSEOF ((EQ R 7)) ((EQ *L 7))))
             ((BECAUSEOF ((EQ R 7)) ((EQ *L 7))))
             ((BECAUSE)) )
OPERATOR : (PC *COL)   FROM (EQC (PLUS 3 3) 6)

    ORIGIN LIST : ( (ND35 ((MEQ R 7 9)) (AEQ R 7))
                    ((IG C3 H) NIL (EQ C3 0 H))
                    ((PC 2 H) ((EQ C3 0 H) (AEQ R 7) (EQ C2 1)) (EQ L 3)) )

        ((AV R) ((MEQ R 7 9)) (AEQ R 7)) RECOGNIZED AS ND35

        (IG C3) GROWN AS ND36
        (EQ C3 0) GROWN AS K32

        (PC 2) GROWN AS ND37
        (EQ L 3) GROWN AS K33

    ((BECAUSEOF ((EQ R 7)) ((EQ *L 7)))) IS
    ((BECAUSEOF ((EQ R 7)) ((EQ R 7))))

    ORIGIN LIST : ( (ND35 ((MEQ R 7 9)) (AEQ R 7)) )

        ((AV R) ((MEQ R 7 9)) (AEQ R 7)) RECOGNIZED AS ND35

    ORIGIN LIST : ()


B71-73 PROTOGROUP

ELEMENTS : ( ((BECAUSEOF ((NEQ L *D)) ((EQ L *D)))) )
OPERATOR : (PC *COL)   FROM (EQC (PLUS *X 1) 7)

    ((BECAUSEOF ((NEQ L *D)) ((EQ L *D)))) IS
    ((BECAUSEOF ((NEQ L *D)) ((EQ L 3))))

    ORIGIN LIST : ( ((OP ?) ((NEQ L *D)) (EQ L 3)) )

        (OP ?) GROWN AS ND38
        (NEQ L *D) CANNOT BE GROWN
        (EQ L 3) GROWN AS K34


B74-78 PROTOGROUP

ELEMENTS : ( ((GREATER (PLUS E O) 10) (COND ((GREATER (PLUS E O) 10)) ((EQ G 1))))
             ((COND ((NGREATER (PLUS E O) 10)) ((EQ G 2))))
             ((COND ((GREATER *L 9)) ((EQ G 1))))
             ((COND ((NGREATER *L 9)) ((EQ G 2))))
             ((IN O *COL)) )
OPERATOR : (PC 5)   FROM (PLUS E O)

    ORIGIN LIST : ( ((PC 6 H I) NIL (COND ((GREATER (PLUS E O) 10)) ((EQ G 1)))) )

        (PC 6 H I) GROWN AS ND39
        (COND ((GREATER (PLUS E O) 10)) ((EQ G 1))) GROWN AS K35

    ORIGIN LIST : ( ((PC 6 H I) NIL (COND ((NGREATER (PLUS E O) 10)) ((EQ G 2)))) )

        (PC 6 H I) GROWN AS ND40
        (COND ((NGREATER (PLUS E O) 10)) ((EQ G 2))) GROWN AS K36

    ((COND ((GREATER *L 9)) ((EQ G 1)))) IS
    ((COND ((GREATER *L 9)) ((EQ G 1))))
    ORIGIN LIST : ( ((PC 6 H I) NIL (COND ((GREATER *L 9)) ((EQ G 1)))) )

        (PC 6 H I) GROWN AS ND41
        (COND ((GREATER *L 9)) ((EQ G 1))) GROWN AS K37

    ((COND ((NGREATER *L 9)) ((EQ G 2)))) IS
    ((COND ((NGREATER *L 9)) ((EQ G 2))))

    ORIGIN LIST : ( ((PC 6 H I) NIL (COND ((NGREATER *L 9)) ((EQ G 2)))) )

        (PC 6 H I) GROWN AS ND42
        (COND ((NGREATER *L 9)) ((EQ G 2))) GROWN AS K38

    ((IN O *COL)) IS ((IN O 5))

    ORIGIN LIST : ( ((FC O H) NIL (IN O 5)) )

        (FC O) GROWN AS ND43
        (IN O 5) GROWN AS K39

        (PC 5) GROWN AS ND44


B79-80 PROTOGROUP

ELEMENTS : ( ((THEREFORE) (PEQ E 0)) )
OPERATOR : (PC *COL)   FROM (EQC (PLUS *L *L) *D)

    ORIGIN LIST : ( ((PC 5 H) NIL (PEQ E 0)) )

        (PEQ E 0) GROWN AS K40


B83-85 PROTOGROUP

ELEMENTS : ( ((PEQ E 9))
             ((BECAUSEOF (AND ((EQ E 9)) ((EQ *C 1))) ((EQ *C 1)))) )
OPERATOR : ()

    ORIGIN LIST : ( ((PC 5 H) NIL (PEQ E 9)) )

        (PEQ E 9) GROWN AS K41

    ((BECAUSEOF (AND ((EQ E 9)) ((EQ *C 1))) ((EQ *C 1)))) IS
    ((BECAUSEOF ((EQ E 9) (EQ *C 1)) ((EQ C6 1))))

    ORIGIN LIST : ( ((AV E H) ((PEQ E 9)) (AEQ E 9))
                    ((AV C5 H) NIL (AEQ C5 1 H))
                    ((PC 5 H) ((AEQ C5 1 H) (AEQ E 9)) (EQ C6 1)) )

        (AV E) GROWN AS ND45
        (AEQ E 9) GROWN AS K42

        (AV C5) GROWN AS ND46
        (AEQ C5 1) GROWN AS K43

        (PC 5) GROWN AS ND47
        (EQ C6 1) GROWN AS K44


B89 PROTOGROUP

ELEMENTS : ()
OPERATOR : (PC 5)   FROM (EQC (PLUS O *L) O)
```

```
         ((PC 5) NIL NIL) MERGED WITH ND47


B92-96 PROTOGROUP

ELEMENTS : ( ((BECAUSEOF ((NEQ *C 1)) ((EVEN E))))
             ((AEQ R 7))
             ((PEQ R 9)) )
OPERATOR : (PC 3)   FROM (EQC (PLUS A A) E)

    ((BECAUSEOF ((NEQ *C 1)) ((EVEN E)))) IS
    ((BECAUSEOF ((NEQ *C 1)) ((EVEN E))))

    ORIGIN LIST : ( ((TD C3 1 H) ((EQ C3 0)) (NEQ C3 1 H))
                    ((PC 3) ((NEQ C3 1 H)) (EVEN E)) )

        (TD C3 1) GROWN AS ND48
        (NEQ C3 1) GROWN AS K45

        (PC 3) GROWN AS ND49
        (EVEN E) GROWN AS K46

        PBG CONFLICT : (EVEN E) VS (AEQ E 9)

        PBG CONFLICT : (EVEN E) VS (PEQ E 9)

    ORIGIN LIST : ( (ND35 ((MEQ R 7 9)) (AEQ R 7)) )

        ((AV R) ((MEQ R 7 9)) (AEQ R 7)) RECOGNIZED AS ND35

    ORIGIN LIST : ( (ND33 ((PEQ R 7) (GREATER R 5) (ODD R)) (PEQ R 9)) )

        ((GN R) ((PEQ R 7) (GREATER R 5) (ODD R)) (PEQ R 9)) RECOGNIZED AS ND33


B98 PROTOGROUP

ELEMENTS : ( ((IF) (EQ E *D)) )
OPERATOR : ()

    ((IF) (EQ E *D)) IS ((IF) (EQ E *D))

    ORIGIN LIST : ()


*** PAS-I FINISHED ***
```

I.3. Comparison between PBG of PAS-I and Manual Analysis for S3 B1-100 on D+G=R (see Tables 8 and 9).

	PAS-I Analysis			Manual Analysis		Agreement*	
1.				B1-4	(Ask Exp about rules)	x	
2.	B5-7	ND1	(RECALL D)	B5-7	Sub (PC 1)	s	
3.		ND2	(RECALL C1)		Sub (PC 1)	s	
4.		ND3	(PC 1)		(PC 1)	s	
5.	B10	ND4	(FC T)	B8-19	In PS	s	
6.	B12-14	ND5	(FC D)		In PS	s	
7.		ND6	(COUNT A)		Sub FL in PS	s	
8.	B15	ND7	(COUNT L)		Sub FL in PS	s	
9.	B19-22	ND8	(COUNT R)		Sub FL in PS	s	
10.					(GET R)	+mn	-C
11.		ND9	(PC 2)	B20-22	(PC 2)	=n	
12.			(no backup)		(backup)	+mb	GPBG
13.	B23-30	(=ND9)	(PC 2)	B23-25	(PC 2)	+mn	GPBG
14.		ND10	(GN R) → (PEQ R 1)	B26-30	Sub (GN R)	s	
15.		ND11	(GN R) → (PEQ R 3)		Sub (GN R)	s	
16.		ND12	(GN R) → (PEQ R 5)		Sub (GN R)	s	
17.		ND13	(TD R 5)	B28	(TD R 5) (during (GN R)	=n	
18.		ND14	= ND13 (backup)		(backup implicit)	=b	
19.		ND15	(GN R) → (PEQ R 7)		Sub (GN R)	s	
20.		ND16	(GN R) → (PEQ R 9)		Sub (GN R)	s	
21.		ND17	(GN R) → (MEQ 1 3 7 9)		(GN R)	=n	
22.			(no backup)		(backup)	+mb	GPBG
23.	B33-38	ND18	(IG C6)	B31-35	Sub (PC 6)	s	
24.		ND19	(PC 6)		(PC 6)	=n	

* See Tables 8 and 9 for key.

	PAS-I Analysis			Manual Analysis		Agreement	
25.			(no backup)		(backup)	+mb	SP
26.				B36-39	(PC 6)	+mn	SP
27.		ND20	(NOTICE C6)		Sub (PC 6)↑	s	
28.		ND21	= ND20 (backup)		(backup)	=b	
29.	B43-45	ND22	(AV L)	B40-43	(AV L)	=n	
30.		ND23	(PC 2)	B44	(PC 2)	=n	
31.	B47	ND24	(PC 6)	B45-47	(PC 6)	=n	
32.				B48	(GET C7 = 1)	+mn	-C
33.				B49-50	(ASK EXP)	+mn	-C
34.			(no backup)		(backup)	+mb	-C
35.	B56-61	ND25	(RECALL C7)	B51-53	(ASK EXP)	x	
36.		ND26	(TD C7 1)		Sub (ASK EXP)	s	
37.		ND27	(FN (EQ C7 1))	B54-57	(FN (EQ C7 1)) in PS	s	
38.			(backup)		(backup)	=b	
39.					(backup)	x	
40.		ND28	(RECALL C7)	B58	Sub (PC 6)	s	
41.		ND29	(PC 6)		(PC 6)	=n	
42.		ND30	= ND28 (backup)		(no backup)	+pb	Minor
43.		ND31	= ND29 (for backup)			s	
44.		ND32	(GN R) → (PEQ R 7)	B59-60	Sub (GN R)	s	
45.		ND32	(GN R) → (PEQ R 9)		Sub (GN R)	s	
46.		ND34	(GN R) → (MEQ R 7 9)		(GN R)	=n	
47.		ND35	(AV R)	B61	(AV R)	=n	
48.				B62	(PC 2)	+mn	LP
49.				B63-64	(TD L 3)	+mn	LP
50.			(no backup)		(backup)	+mb	LP

	PAS-I Analysis			Manual Analysis		Agreement	
51.	B66-70	ND36	(IG C3)		Sub (PC 2)	s	
52.		ND37	(PC 2)	B65	(PC 2)	=n	
53.				B66-71	(TD L 3)	+mn	LP
54.	B71-73	ND38	(OP ?)			+pn	LP,GPBG
55.	B74-78	ND39	(PC 6 H I)	B72-75	(PC 6)	=n	
56.		ND40	(PC 6 H I)		Sub (PC 6)	s	
57.		ND41	(PC 6 H I)		Sub (PC 6)	s	
58.		ND42	(PC 6 H I)		Sub (PC 6)	s	
59.		ND43	(FC 0)	B76-77	(GET C6)	≠n	OM,-C
60.		ND44	(PC 5) → (PEQ E 0) (PEQ E 9)	B78-80	(PC 5) → (EQ E 0)	=≠n	GPBG
	B79-80	(=ND44)	(PC 5) → (PEQ E 0)				
61.			(no backup)		(backup)	+mb	GPBG
62.			((?))	B81-82	(PC 5) → (EQ 0)	+mn	GPBG
63.			(no backup)		(backup)	+mb	GPBG
	B83-85	(=ND44)	(PC 5) → (PEQ E 9)				
64.		ND45	(AV E) → (EQ E 9)	B83-84	(PC 5) → (EQ E 9) (EQ C5 1)	≠n	OM,GPBG
65.		ND46	(AV C5) → (EQ C5 1)			+pn	OM
66.		ND47	(PC 5) → (EQ C6 1)	B85	(PC 5) → (EQ C6 1)	=n	
67.			(no backup)		(backup)	+mb	GPBG
68.		(=ND47)	(PC 5)	B86-90	(PC 5)	+mn	GPBG
69.	B92-94	ND48	(TD C3 1)			+pn	OM
70.		ND49	= ND50 (backup)		(backup)	=b	
71.		ND50	= ND48 (backup)		(no backup)	+pb	

	PAS-I Analysis	Manual Analysis		Agreement	
72.	ND51 (PC 3)	B91-93	(PC 3)	=n	
73.		B94	(CHK (EQ C3 0))	+mn	-C
74.		B95	(PC 2)	+mn	-C
75.	(=ND35) (AV R) → (AEQ R 7)	B95.1	(CHK (AEQ R 7))	+mn	-C,GPBG
76.	(=ND33) (GN R) → (PEQ R 9)	B96-97	(GN R) → (PEQ 7 9)	+mn	-C
77.	(no backup)		(backup)	+mb	-C
78.		B98	(PC 3) → (EVEN E)	+mn	LP
79.		B98.1	(GN E) → (MEQ 2 4 6 8)	+mn	LP
80.		B99	(TD E) (during (GN E))	+mn	LP
81.			(backup)	+mb	LP

APPENDIX I

I.4. Annotation of Disparities between PAS-I and Manual Analysis for S3 B1-100 on D+G=R.

<u>Item</u>

1. <u>Excluded</u>. PAS-I did not recognize any of the conversation with the experimenter; Manual created a node labeled EXP. Not a relevant difference.

10. <u>Extra Manual Node</u>. PAS-I does not have notion of a goal (here GET). Goals play an anomolous role in manual PBG (is it an operator?), though they are well defined at the level of the production system.

12. <u>Extra Manual Backup</u>. PAS-I merges the second (PC 2) → (ODD R) with the first one (ND9), thus doesn't see the backup.

13. <u>Extra Manual Node</u>. Part of error described in item 12.

22. <u>Extra Manual Backup</u>. Manual analysis took the non-use of 1,3,7,9, as opposed to (ODD R) as an indication that line of attack has been abandoned. Remark of EXP, ignored by PAS-I, was taken as reinforcing this interpretation.

25. <u>Extra Manual Backup</u>. PAS-I pooled the two branches of the PBG, which were still separate semantic elements, into a single protogroup, using one (B33-35) as evidence for knowledge elements and the other (B36) as evidence for (PC 6).

26. <u>Extra Manual Node</u>. Part of error described in item 25.

32. <u>Extra Manual Node</u>. PAS-I does not have the notion of a goal (see item 10).

33. <u>Extra Manual Node</u>. PAS-I does not have the notion of (ASK X).

34. <u>Extra Manual Backup</u>. Since PAS-I does not recognize interaction with the experimenter, it does not recognize a negative answer as termination of a branch, hence as backup.

Item

35. Excluded. Both PAS-I and Manual recognize the input of (EQ C7 0) and its effect on the existing situation. PAS-I does not have the notion of experimenter input (as in item 33 as well) and attributes this to (RECALL C7) -- which is as good as can be expected under the circumstances. Actually, proper assessment of PAS-I interpretation on this item requires taking items 36 (TD) and 37 (FN) into account, which show that the interpretation is in essential agreement with the manual analysis.

39. Excluded. Manual backs up in two stages, inserting state (but not an operator) at the point of realization of $R \neq 3$. PAS-I does the back up all in one step. In the dual representation of the PBG, where nodes are states of knowledge, it seemed to make sense to interpolate this extra stage, but it is an anomolous feature of the manual PBG. PAS-I, on the contrary, simply makes the return all in one step. Not a relevant difference.

42. Extra PAS-I Backup. The return from $C7 \neq 1$ (hence $R \neq 3$) terminates before the node where (GN R) → (PEQ R 3), but after the node (GN R) → (PEQ R 1), since this is not contradicted. The next step reveals that $R > 5$, hence a further backup to before (GN R) → (PEQ R 1) is required. In the manual analysis return is made immediately to the latter point, since (GN R) is a single node. The difficulty is that to obtain a correct handling of (GN R) in co-occurrence with (TD R) PAS-I has lost the unifying character of all the GN's, namely, that they constitute a single process and that one cannot branch out of the middle it. (The coroutine structure with TD is a different situation altogether).

Item

48. Extra Manual Node. The Linguistic Processor failed to obtain any information for B62-65, thus obtaining no indication of (PC 2) → (EQ L 3) followed by (TD L 3) → (NEQ). The manual analysis interpreted this by means of a rather substantial inference, since B62-65 is entirely free from content references.

49. Extra Manual Node. Part of error described in item 48.

50. Extra Manual Backup. Part of error described in item 48.

51. Extra Manual Node. Actually part of error described in item 48, though it shows up later. TD is reflected in the verbalizations only when a contradiction is detected; when the assignment is all right, usually no overt indication occurs. The occurrence of confirming TD's is inferred at the level of the production system by needing a uniform decision rule in order to get TD evoked for all the contradictory cases. However, in this case the manual analysis posited a TD that failed (B66-71); hence, when the path was being retraced, it posited a TD that was confirming, even though there was no direct evidence in the verbal behavior.

54. Extra PAS-I Node. At least two joint errors occurred. The Linguistic Processor misinterprets B72 and the Semantic Processor combines this with the reference to L in B73 to derive the notion that because L is not equal to something it is equal to something(else). This is combined with inappropriate protogrouping, which puts (BECAUSE 3 + 3 IS 6) in one protogroup and (+ 1 is 7) in the next. Bad topic segmenting would appear to be involved as well (which, of course, is taken as a given in this analysis). Thus, PAS-I grows an unknown operator (OP ?) with output (EQ L 3). A final chance for assimilating the operator to the prior operator (PC 2) which also produced (EQ L 3) is not seized, since the

Item

mechanism for responding to redundancies in the PBG (here two productions of (EQ L 3) in a row) is inadequate.

59. PAS-I and Manual Disagree on a Node. Partly the difficulty is that PAS-I does not have goals, so cannot set up (GET C6). However, the alternative that PAS-I comes up with (FC O) is directly at variance with any interpretation involving working backward from C6. The failure of PAS-I (since there is no evidence at all for a concern with O) stems in part from not being concerned about the origin of the letter in (FC *L). It should insist that there is a prior generation of a letter *L before a subject will do (FC *L).

60. PAS-I and Manual both Agree and Disagree on a Node. They agree in positing (PC 5) and in obtaining a result about E = 0. They disagree in that PAS-I says the result is (PEQ E 0)(PEQ E 9) and Manual says the result is (EQ E 0). Assessment of B80, on which (EQ E 0) is based, indicates that it could just as well have been (PEQ E 0). However, the (PEQ E 9) arises from the merging of (PC 5) → (PEQ E 9) from B83-85 with the (PC 5) → (PEQ E 0) from B78 and from B79-80. Thus PAS-I merged all three indications of (PC 5) into a single node, whereas Manual kept them all separate and detected backups (items 61 and 63) between each of them. The difficulty is identical to that discussed in item 12, since it developed nodes on the origin list for all three operations.

61. Extra Manual Backup. Part of error described in item 60.

62. Extra Manual Node. Part of error described in item 60.

63. Extra Manual Backup. Part of error described in item 60.

Item

64. PAS-I and Manual Disagree. PAS-I, having combined all its processing of COL 5 to yield (PEQ E 0)(PEQ E 9), interprets the occurrence of (EQ C6 1) at B85 as implying that E was determined to be 9, hence that this occurred by an assignment (AV E 9), rather than by (PC 5). The production of (EQ C5 1) was taken to be by (AV C5 1), partly because there was no (PC 5) to impute it to, but also because PAS-I does not take into account partial outputs within a protogroup. That is, the PBG is grown in pulses where each ORIGIN-LIST-full is determined by processing all the items from a protogroup. Thus, it did not determine the origin of (EQ C5 1) given (EQ E9) (whence it would have posited another (PC 5)), but did it in parallel with (EQ E 9), hence posited (AV C5 1). This latter difficulty is remediable, but without clearing up the former error (producing (PEQ E 9)) would still not behave correctly at this stage.

65. Extra PAS-I Node. Part of error described in item 64.

67. Extra Manual Backup. PAS-I merges the (PC 5) at B89 with the (PC 5) at B85 (ND46), thus not being able to see the backup, as well as not seeing the second (PC 5). This is identical to the error at item 12.

68. Extra Manual Node. Part of the error described in item 67.

69. Extra PAS-I Node. PAS-I, in order to understand that the subject said that C3 $\neq$ 1 (B94), posited a TD operation, working on (EQ C3 0), which was produced by (IG C3). There are several difficulties here. First, PAS-I uses TD with respect to carries and the manual analysis did not. Second, since PAS-I was prepared to use (EQ C3 0) as input to (TD C3), it is not clear that it should require TD to produce the answer again in order to get (EVEN E). The statement of (NEQ C3 1) could be taken simply to be subsumed within (PC 3). Third, however, PAS-I does not treat (IG X) correctly, since its product (here (EQ C3 0)) should be assumed to be a silent value, not one that is then continuously available. Thus, (IG C3) → (EQ C3 0) should at least be posited again. Of course,

Item

there is evidence (B94 again) that C3 obtains the value 0 not by ignoring it but by a deliberate act of processing. This leads to an alternative hypothesis, namely that the subject realizes that he has been assuming the C3 = 0 and that this leads him to check it, so that he does (PC 2) with (AEQ R 7) to obtain (EQ C3 0) again. This is the inference made in the manual analysis, using the language of B95 as additional evidence.

71. Excluded. An extra node is included in the new path, which is a copy of (TD C3 1), in order to have that information included in the new total knowledge state. This is true generally in repetition of nodes at a backup; this is the only situation that occurred where more than one node had to be copied.

73. Extra Manual Node. PAS-I does not have goals, hence could not posit (CHK X). However, this is also part of the error described in item 69.

74. Extra Manual Node. Part of the error described in item 69.

75. Extra Manual Node. PAS-I does not have goals, hence could not posit (CHK ...). However, PAS-I also identifies the semantic elements concerned with R with the prior elements in the PBG, thus not creating any current action concerned with R.

76. Extra Manual Backup. Part of error described in 75 and 73, since a backup cannot occur without the intervening elements existing.

78. Extra Manual Node. The Linguistic Processor fails to extract enough information, though it does get (EQ E *D). In B98 it does get a knowledge element concerned with E, but then fails to infer from the article AN that the knowledge element is undoubtedly (EVEN E). This is the only example in the present material of a specifically grammatical clue. Since there is no way to complete the extracted knowledge element (i.e.,

Item

to determine the unknown *D), the element is simply abandoned by the Origin Mechanism. The two following topic segments (B99, B100) simply provide a guarantee that some operators were applied. PAS-I has no provision currently for extracting such information at the linguistic stage. Actually, the key knowledge for the manual analysis comes from B101-103, which is beyond the limits of the test run.

79. Extra Manual Node. Part of the error described in 78.

80. Extra Manual Node. Part of the error described in 78 (though the manual inference is quite tentative in any event).

I.5. Analysis of Linguistic and Semantic Processors by PAS-I for S3 B1-100 on D+G=R (see Figures 16, 17, and 18), where se: semantic element, cn: connection, and sp: separation.

Linguistic Processor

B1	Ok, complete, null
B2	Excluded
B3	Ok, complete, null
B4	Ok, complete, null
B5	Ok, incomplete, se
B6	Ok, incomplete, se
B7	Ok, complete, el
B8	Ok, incomplete, se
B9	Ok, incomplete, cn
B10	Ok, complete, el
B11	Ok, complete, el
B12	Ok, incomplete, cn
B13	Ok, complete, el
B14	Ok, incomplete, sp
B15	Ok, complete, el
B16	Ok, incomplete, se
B17	Ok, complete, null
B18	Ok, complete, el
B19	Ok, complete, el
B20	Ok, complete, el
B21	Ok, incomplete, sp
B22	Ok, complete, el
B23	Ok, complete, el
B24	Ok, incomplete, se
B25	Ok, incomplete, cn
B26	Ok, complete, el
B27	Ok, complete, el
B28	Ok, complete, el
B29	Ok, complete, el
B30	Ok, complete, el

Semantic Processor

B1-4	Ok, complete, null, =
B5	Ok, complete, el, +
B6-7	Ok, complete, el, +
B8	Ok, complete, null, +
B9	Ok, incomplete, cn, =
B10	Ok, complete, el, =
B11	Missed, se, -
B12	Ok, incomplete, cn, =
B13	Ok, complete, el, =
B14	Ok, incomplete, sp, =
B15	Ok, complete, el, =
B16-18	Missed, se, -
B19	Ok, incomplete, cn, -
B20	Ok, complete, el, =
B21-22	Ok, incomplete, cn, -
B23	Ok, complete, el, =
B24	Ok, incomplete, se, =
--	Missed, cn, - (creating B24-25)

Linguistic Processor		Semantic Processor	
		B25-30	Ok, complete, el, =
		B25-30	Ok, complete, el, =
		B25-30	Ok, complete, el, =
		B25-30	Ok, complete, el, =
		B25-30	Ok, complete, el, =
		B25-30	Ok, complete, el, =
B31	Excluded		
B32	Ok, complete, el		
		B31-32	Missed, se, -
B33	Ok, complete, el		
B34	Ok, complete, el		
B35	Ok, incomplete, se		
		B33-35	Ok, complete, el, +
B36	Ok, complete, el		
		B36	Ok, complete, el, +
B37	Ok, incomplete, sp		
B38	Ok, complete, el		
		B37-38	Ok, complete, el, +
B39	Ok, incomplete, sp		
B40	Ok, complete, null		
B41	Ok, complete, null		
B42	Ok, complete, el		
		B39-42	Ok, incomplete, cn, =
B43	Ok, complete, el		
B44	Ok, complete, el		
B45	Ok, incomplete, se		
		B43-45	Ok, complete, el, +
B46	Missed, se		
		B46	Missed, se, =
B47	Ok, complete, el		
		B47	Ok, complete, el =
B48	Missed, se		
B49	Ok, incomplete, cn		
		B48-49	Missed, se, =
B50	Missed, se		
		B50	Missed, se, =
B51	Ok, complete, el		
B52	Excluded		
B53	Ok, complete, el		
B54	Ok, complete, null		
B55	Ok, complete, el		
		B51-55	Missed, se, -
B56	Ok, complete, el		
B57	Ok, complete, el		
		B56-57	Ok, complete, el, =
B58	Ok, complete, el		
B59	Ok, complete, el		
B60	Ok, complete, el		
B61	Ok, incomplete, se		
		B58-61	Ok, complete, el, =
		B58-61	Ok, complete, el, =
		B58-61	Ok, complete, el, =
		B58-61	Ok, complete, el, =

Linguistic Processor

B62	Ok, complete, el
B63	Ok, complete, el
B64	Ok, complete, el
B65	Ok, incomplete, cn
B66	Ok, complete, el
B67	Ok, incomplete, se
B68	Ok, complete, el
B69	Ok, incomplete, se
B70	Ok, complete, el
B71	Ok, complete, el
B72	Missed, se
B73	Missed, se
B74	Ok, incomplete, sp
B75	Ok, complete, el
B76	Ok, complete, el
B77	Ok, complete, el
B78	Ok, incomplete, se
B79	Missed, se
B80	Ok, complete, el
B81	Missed, se
B82	Ok, complete, el
B83	Ok, complete, el
B84	Ok, complete, el
B85	Ok, complete, el
B86	Excluded
B87	Ok, incomplete, se
B88	Ok, incomplete, se
B89	Ok, incomplete, se
B90	Ok, complete, el
B91	Ok, incomplete, sp

Semantic Processor

B62-65	Missed, se, -
B66-69	Ok, complete, el, +
B70	Ok, complete, el, =
B71	Ok, complete, el, =
B72-73	Missed, se, =
B74-77	Ok, complete, el, =
B74-77	Ok, complete, el, =
B74-77	Ok, complete, el, =
B74-77	Ok, complete, el, =
B78	Ok, complete, el, +
B79	Missed, se, =
B80	Ok, complete, el, =
B81-82	Missed, se, =
B83	Ok, complete, el, =
B84-85	Ok, incomplete, se, -
B86-88	Missed, se, -
B89	Ok, complete, el, +
--	Missed, cn, -
B90-91	Ok, complete, null, +

Linguistic Processor

B92	Ok, complete, el
B93	Ok, complete, el
B94	Ok, complete, el
B95	Ok, incomplete, cn
B96	Ok, incomplete, cn
B97	Missed, sp
B98	Ok, incomplete, se
B99	Missed, sp
B100	Missed, cn

Semantic Processor

B92	Ok, complete, el, =
B93-94	Ok, complete, el, =
B95	Ok, complete, el, +
B96	Ok, complete, el, +
B97	Missed, sp, =
B98	Ok, incomplete, se, =
B99-100	Missed, se, =

APPENDIX I

I.6. Determination of unknowns by PAS-I
for S3 B1-100 on D+G=R (see Table 10).

```
 1.  *COL  +(DU)    B12-14   ((IN 5 *COL))
                              (IN D  6  )

 2.  *COL  +(DU)             ((THEN) (IN 5 *COL))
                                     (IN D   6   )

 3.  *C    +(OM)    B19-22   ((BECAUSEOF ((EQ *C 1))  ((ODD R)))
                                         (EQ C1 1)

 4.  *D    +(DU)    B23-30   ((EVEN (PLUS *D *D)))
                                    (PLUS  L  L)

 5.  *D    *(DU)

 6.  *X    *(DU)             ((BECAUSEOF ((ODD (PLUS  *X 1)))  ((PEQ R 1))))
                                          (ODD  R)

 7.  *X    +(DU)             ((BECAUSEOF ((ODD (PLUS *X 1)))  ((PEQ R 3))))
                                          (ODD  R)

 8.  *X    +(DU)             ((BECAUSEOF ((ODD (PLUS *X 1)))  ((PEQ R 5))))
                                          (ODD  R)

 9.  *X    +(DU)             ((BECAUSEOF ((ODD (PLUS *X 1)))  ((PEQ R 7))))
                                          (ODD  R)

10.  *X    +(DU)             ((BECAUSEOF ((ODD (PLUS *X 1)))  ((PEQ R 9))))
                                          (ODD  R)

11.  *C    +(DU)    B33-38   ((COND ((EQ C6 1))  ((EQ *C *D))))
                                                  (EQ C6  1)

12.  *D    +(DU)

13.  *COL  +(OM)    B43-45   (PC *COL) from (PLUS 1 1)
                             (PC   2  )

14.  *L    *(SPG)   B47      (PC 6) from (EQC (PLUS 5 *L) 3)

15.  *L    -(LP)    B50      ((NEQ *L *D) (LETTER R) (QUES))
                              (NEQ  D *D)

16.  *D    -(none)

17.  *L    +(DU)    B56-61   ((OPIO (FN ((EQ *L 13)))  ((NEQ *L 13))  ((NEQ R 3))))
                                        (EQ C7 1 )      (NEQ C7  1)

18.  *L    +(DU)

19.  *L    +(DU)             ((BECAUSEOF ((GREATER R 5))  ((PEQ *L 7))))
                                                           (PEQ  R 7)
```

```
20.  *L    +(DU)              ((BECAUSEOF ((GREATER R 5)) ((PEQ *L 9))))
                                                           (PEQ  R 9)

21.  *L    +(DU)     (B56-61) ((MEQ *L 7 9))
                               (MEQ  R 7 9)

22.  *L    +(DU)              ((BECAUSEOF ((MEQ *L 7 9)) ((AEQ *L 7))))
                                           (MEQ  R 7 9)   (AEQ  R 7)

23.  *L    +(DU)

24.  *COL  +(OM)     B66-70   (PC *COL) from (EQC (PLUS 3 3) 6)
                              (PC   2  )

25.  *L    +(DU)              ((BECAUSEOF ((EQ R 7)) ((EQ *L 7))))
                                                      (EQ  R 7)

26.  *X    -(none)   B71-73   (PC *COL) from (EQC (PLUS *X 1) 7)
                              (OP   ?  )

27.  *D    -(none)            ((BECAUSEOF ((NEQ L *D)) ((EQ L *D))))
                                           (NEQ L *D)   (EQ L  3)

28.  *D    -(LP)

29.  *L    -(none)   B74-78   ((COND ((GREATER *L 9)) ((EQ G 1))))
                                      (GREATER *L 9)

30.  *L    -(none)            ((COND ((NGREATER *L 9)) ((EQ G 2))))
                                      (NGREATER *L 9)

31.  *COL  +(DU)              ((IN O *COL))
                               (IN O  5  )

32.  *L    +(OM)     B79-80   (PC *COL) from (EQC (PLUS *L *L) *D)
                              (PC  5  )

33.  *L    +(OM)

34.  *D    +(OM)

35.  *C    +(OM)     B83-85   ((BECAUSEOF (AND ((EQ E 9)) ((EQ *C 1))) ((EQ *C 1))))
                                                           (EQ C5 1)    (EQ C6 1)

36.  *C    +(DU)

37.  *L    +(SPG)    B89      (PC 5) from (EQC (PLUS O *L) O)

38.  *C    +(OM)     B92-96   ((BECAUSEOF ((NEQ *C 1)) ((EVEN E))))
                                           (NEQ C3 1)

39.  *D    -(none)   B98      ((IF) (EQ E *D))
                                    (EQ E *D)
```

APPENDIX II

II.1. PAS-I Linguistic and Semantic Processor Output for S3 B101-143 on D+G=R.

```
B101. E HAS TO BE AN EVEN NUMBER
           (EVEN E)

         <B101>. ((EVEN E))


B102. AND E + 0 = 0 --
           (AND) (EQC (PLUS E 0) 0)

         <B102>. ((AND) (EQC (PLUS E 0) 0))


B103. E CANNOT BE 9 .
           (NEG) (EQ E 9)

         <B103>. ((NEQ E 9))


B104. EXP : WHAT ARE YOU THINKING NOW ?
           (EXP)

         <B104>. ((?))


B105. I 'M GOING BACK OVER THESE L 'S HERE AND TRY TO THINK WHAT WOULD HAPPEN IF THEY ARE NI --
           (ASSUME) (EQ L 9)

         <B105>. ((AEQ L 9))


B106. RATHER --
           (CORRECTION)
B107. LET 'S SEE , HOW DID I ARRIVE AT THE POINT OF THAT ?
           (QUES)
B108. THIS IS GOING TO BE A' LITTLE CONFUSING TO START TRYING TO TRACE BACK HERE .
           (?)
B109. WHAT 'S THE REASONING HERE ?
           (QUES)

      <B106-109>. ((?))


B110. I 'M THINKING IN THE BACK OF MY MIND WHAT THIS R WAS .
           (EQ R *D)

         <B110>. ((EQ R *D))


B111. I DECIDED THAT R HAD TO BE GREATER THAN 5 ,
           (GREATER R 5)

         <B111>. ((GREATER R 5))


B112. BECAUSE THAT WAS GIVEN
           (BECAUSE)

         <B112>. ((?))
```

```
B113. AND R + G ,
           (AND) (PLUS R G)

         <B113>. ((AND) (PLUS R G))


B114. OR RATHER , D + G = R .
           (CORRECTION) (EQC (PLUS D G) R)

         <B114>. ((CORRECTION) (EQC (PLUS D G) R))


B115. I KNOW YOU 'RE WONDERING WHAT I 'M THINKING .
           (?)

         <B115>. ((?))


B116. I 'M STILL TRYING TO LOOK AT THIS SECOND COLUMN HERE , WHERE E + O = O ,
           (EQC (PLUS E O) O) (PLACE SECOND)

         <B116>. ((EQC (PLUS E O) O))


B117. AND A + A = E .
           (AND) (EQC (PLUS A A) E)

         <B117>. ((AND) (EQC (PLUS A A) E))


B118. THEN AGAIN , THAT 'S ASSUMING THAT N IS LESS THAN 3 ,
           (IF) (SMALLER N 3)
B119. BECAUSE I DO N'T WANT TO BE CARRYING 1 INTO THAT E + O COLUMN .
           (BECAUSE) (NEG) (CIN 1 (PLUS E O))

      <B118-119>. ((BECAUSEOF ((NEQ C5 1)) ((SMALLER N 3))))


B120. I THINK I 'LL TRY ONCE MORE HERE --
           (?)
B121. JUST TRYING TO SORT OF BLUFF MY WAY THROUGH THIS .
           (?)

      <B120-121>. ((?))


B122. THAT IS , JUST ASSUME SOME VALUE FOR A ,
           (ASSUME) (LETTER A)

         <B122>. ((OPIO (FA (EQ E *D)) NIL ((EQ A *D))))


B123. SO I CAN GET THAT E .
           (THEREFORE) (LETTER E)

         <B123>. ((AEQ A *D))


B124. I CAN DO BETTER THAN THAT .
           (?)
B125. I --
           (?)

      <B124-125>. ((?))
```

```
B126. I KNOW THAT E + O HAS TO EQUAL O ,
          (EQC (PLUS E O) O)

        <B126>. ((EQC (PLUS E O) O))


B127. AND , AT MOST , O IS GOING TO BE 9 ;
          (AND) (ASSUME) (EQ O 9)
B128. IN WHICH CASE E WOULD BE ZERO .
          (THEREFORE) (EQ E 0)

     <B127-128>. ((BECAUSEOF ((AEQ O 9)) ((EQ E 0))))


B129. IF E IS ZERO .
          (IF) (EQ E 0)

        <B129>. ((IF) (EQ E 0))


B130. A + A --
          (PLUS A A)

        <B130>. ((PLUS A A))


B131. BUT A CAN N'T EQUAL 5 --
          (NEG) (EQ A 5)

        <B131>. ((NEQ A 5))


B132. THAT IS , A + A WOULD EQUAL E
          (EQC (PLUS A A) E)

        <B132>. ((EQC (PLUS A A) E))


B133. AND IF E WERE ZERO ,
          (AND) (IF) (EQ E 0)
B134. A WOULD HAVE TO EQUAL 5 ;
          (EQ A 5)

     <B133-134>. ((COND ((EQ E 0)) ((EQ A 5))))


B135. BUT A CAN N'T EQUAL 5 .
          (NEG) (EQ A 5)

        <B135>. ((NEQ A 5))


B136. AND --
          (AND)
B137. SEE --
          (?)

     <B136-137>. ((?))


B138. I DECIDED THAT R HAD TO BE AN ODD NUMBER ,
          (ODD R)
B139. AND HAS TO BE GREATER THAN 5 ,
          (AND) (GREATER *L 5)
```

```
B140. WHICH LEAVES ONLY 7
        (THEREFORE) (DIGIT 7)
B141. AND 9 .
        (PLS 9)

    <B138-141>. ((BECAUSEOF (AND ((ODD R)) ((GREATER *L 5))) ((PEQ *L 7))))

    <B138-141>. ((BECAUSEOF (AND ((ODD R)) ((GREATER *L 5))) ((PEQ *L 9))))

    <B138-141>. ((MEQ *L 7 9))


B142. I THINK THAT REASONING IS CORRECT .
        (?)
B143. WELL , AT WORST I HAVE ONLY TWO SOLUTIONS TO WORK ON IN THAT CASE , STARTING FROM THAT POINT .
        (?)

    <B142-143>. ((?))
```

II.2. PAS-I Group Processor and PBG Mechanism Output for S3 B101-143 on D+G=R.

```
B101-103 PROTOGROUP

ELEMENTS : ( ((EVEN E))
             ((AND))
             ((NEQ E 9)) )
OPERATOR : (PC 5)   FROM (EQC (PLUS E 0) 0)

    ORIGIN LIST : ( (ND51 ((NEQ C3 1)) (EVEN E)) )

        ((PC 3) ((NEQ C3 1)) (EVEN E)) RECOGNIZED AS ND51

    ORIGIN LIST : ()

    ORIGIN LIST : ( ((TE (NEQ E 9) H) ((EVEN E)) (NEQ E 9)) )

        (TE (NEQ E 9)) GROWN AS ND52
        (NEQ E 9) GROWN AS K47

        (PC 5) GROWN AS ND53


B105 PROTOGROUP

ELEMENTS : ( ((AEQ L 9)) )
OPERATOR : ()

    ORIGIN LIST : ( ((AV L H) NIL (AEQ L 9)) )

        (AV L) GROWN AS ND54
        (AEQ L 9) GROWN AS K48

        PBG CONFLICT : (AEQ L 9) VS (EQ L 3)

        PBG CONFLICT : (AEQ L 9) VS (EQ L 3)


B110-111 PROTOGROUP

ELEMENTS : ( ((EQ R *D))
             ((GREATER R 5)) )
OPERATOR : ()

    ((EQ R *D)) IS ((EQ R 7))

    ORIGIN LIST : ( ((AV R H) ((MEQ R 7 9)) (AEQ R 7)) )

        (AV R) GROWN AS ND56
        (AEQ R 7) GROWN AS K49

    ORIGIN LIST : ( (ND31 ((EQ D 5) (EQ C7 0)) (GREATER R 5)) )

        ((PC 6) ((EQ D 5) (EQ C7 0)) (GREATER R 5)) RECOGNIZED AS ND31


B113 PROTOGROUP

ELEMENTS : ( ((AND)) )
OPERATOR : (PC *COL)   FROM (PLUS R G)

    ORIGIN LIST : ()
```

```
B114 PROTOGROUP

ELEMENTS : ( ((CORRECTION)) )
OPERATOR : (PC 6)   FROM (EQC (PLUS D G) R)

    ORIGIN LIST : ()

        (PC 6) GROWN AS ND57


B116 PROTOGROUP

ELEMENTS : ()
OPERATOR : (PC 5)   FROM (EQC (PLUS E O) O)

        (PC 5) GROWN AS ND58


B117-119 PROTOGROUP

ELEMENTS : ( ((AND))
             ((BECAUSEOF ((NEQ C5 1)) ((SMALLER N 3)))) )
OPERATOR : (PC 3)   FROM (EQC (PLUS A A) E)

    ORIGIN LIST : ()

    ORIGIN LIST : ( ((OP ?) ((NEQ C5 1)) (SMALLER N 3)) )

       (NEQ C5 1) ORIGIN UNKNOWN
        (OP ?) GROWN AS ND59
        (SMALLER N 3) GROWN AS K50

        (PC 3) GROWN AS ND60


B122-123 PROTOGROUP

ELEMENTS : ( ((OPIO (FA (EQ E *D)) NIL ((EQ A *D))))
             ((AEQ A *D)) )
OPERATOR : ()

    ((OPIO (FA (EQ E *D)) NIL ((EQ A *D)))) IS
    ((OPIO (FA (EQ E *D)) NIL ((EQ A *D))))

    ORIGIN LIST : ( ((FA (EQ E *D)) NIL (EQ A *D)) )

        (FA (EQ E *D)) GROWN AS ND61
        (EQ A *D) GROWN AS K51
        (FA (EQ E *D)) CANNOT BE GROWN

    ((AEQ A *D)) IS ((AEQ A *D))

    ORIGIN LIST : ()


B126-129 PROTOGROUP

ELEMENTS : ( ((BECAUSEOF ((AEQ O 9)) ((EQ E 0))))
             ((IF) (EQ E 0)) )
OPERATOR : (PC 5)   FROM (EQC (PLUS E O) O)

    ORIGIN LIST : ( ((AV O H) NIL (AEQ D 9))
                    ((AV G H) NIL (AEQ G 2 H))
                    (ND1 NIL (EQ D 5))
                    ((PC 6 H) ((AEQ G 2 H) (EQ D 5) (EQ R 7)) (EQ C6 8 H))
                    ((PC 5) ((EQ C6 8 H) (AEQ O 9)) (EQ C5 8 H))
                    ((PC 5) ((EQ C5 8 H) (AEQ O 9)) (EQ E 8)) )

          (AV O) GROWN AS ND62
          (AEQ O 9) GROWN AS K52

          (AV G) GROWN AS ND63
          (AEQ G 2) GROWN AS K53

          ((RECALL D) NIL (EQ D 5)) RECOGNIZED AS ND1

          (PC 6) GROWN AS ND64
          (EQ C6 8) GROWN AS K54

          (PC 5) GROWN AS ND65
          (EQ C5 8) GROWN AS K55

          (PC 5) GROWN AS ND66
          (EQ E 8) GROWN AS K56

      ORIGIN LIST : ( (ND66 ((AEQ O 9) (EQ C5 8)) (EQ E 8)) )

          ((PC 5) ((AEQ O 9) (EQ C5 8)) (EQ E 8)) RECOGNIZED AS ND66


  B138-135 PROTOGROUP

  ELEMENTS : ( ((NEQ A 5))
               ((EQC (PLUS A A) E))
               ((COND ((EQ E 8)) ((EQ A 5))))
               ((NEQ A 5)) )
  OPERATOR : (PC 3)   FROM (PLUS A A)

      ORIGIN LIST : ( (ND1 NIL (EQ D 5))
                      ((TD A 5 H) ((EQ D 5)) (NEQ A 5)) )

          ((RECALL D) NIL (EQ D 5)) RECOGNIZED AS ND1

          (TD A 5) GROWN AS ND67
          (NEQ A 5) GROWN AS K57

      ORIGIN LIST : ()

      ORIGIN LIST : ( ((AV C4 H) NIL (AEQ C4 1 H))
                      ((PC 3) ((AEQ C4 1 H)) (COND ((EQ E 8)) (EQ A 5))) )

          (AV C4) GROWN AS ND68
          (AEQ C4 1) GROWN AS K58

          (PC 3) GROWN AS ND69
          (COND ((EQ E 8)) (EQ A 5)) GROWN AS K59

      ORIGIN LIST : ( (ND67 ((EQ D 5)) (NEQ A 5)) )

          ((TD A 5) ((EQ D 5)) (NEQ A 5)) RECOGNIZED AS ND67


  B138-141 PROTOGROUP

  ELEMENTS : ( ((BECAUSEOF (AND ((ODD R)) ((GREATER *L 5))) ((PEQ *L 7))))
               ((BECAUSEOF (AND ((ODD R)) ((GREATER *L 5))) ((PEQ *L 9))))
               ((MEQ *L 7 9)) )
  OPERATOR : ()
```

```
    ((BECAUSEOF (AND ((ODD R)) ((GREATER *L 5))) ((PEQ *L 7)))) IS
    ((BECAUSEOF ((ODD R) (GREATER R 5)) ((PEQ R 7))))

    ORIGIN LIST : ( (ND32 ((ODD R) (GREATER R 5)) (PEQ R 7)) )

        ((GN R) ((ODD R) (GREATER R 5)) (PEQ R 7)) RECOGNIZED AS ND32

    ((BECAUSEOF (AND ((ODD R)) ((GREATER *L 5))) ((PEQ *L 9)))) IS
    ((BECAUSEOF ((ODD R) (GREATER R 5)) ((PEQ R 9))))

    ORIGIN LIST : ( (ND33 ((PEQ R 7) (GREATER R 5) (ODD R)) (PEQ R 9)) )

        ((GN R) ((PEQ R 7) (GREATER R 5) (ODD R)) (PEQ R 9)) RECOGNIZED AS ND33

    ((MEQ *L 7 9)) IS ((MEQ R 7 9))

    ORIGIN LIST : ( (ND34 ((PEQ R 9) (PEQ R 7)) (MEQ R 7 9)) )

        ((GN R) ((PEQ R 9) (PEQ R 7)) (MEQ R 7 9)) RECOGNIZED AS ND34

*** PAS-I FINISHED ***
```

APPENDIX II

II.3. Comparison between PBG of PAS-I and Manual Analysis for S3 B101-143 on D+G=R.

	PAS-I Analysis			Manual Analysis			
1.	B101-103	(=ND51	(PC 3) → (EVEN E)	B101-102	(PC 5) → (PEQ E 0 9)	=≠n	OM
2.		ND52	(TE (NEQ E 9)) → (NEQ E 9)	B103	(TD E 9) → (NEQ E 9)	=n	
3.		ND53	(PC 5)			=≠n	OM,GPBG
4.	B105	ND54	=ND55 (backup)	B103.1	(backup)	=b	
5.		ND55	(AV L) → (AEQ L 9)	B104-105	(AV L) → (AEQ L 9)	=n	
6.				B106	(PC 2) → (EQ R 9)	+mn	LP
7.				B106.1	(TD R 9) → (NEQ R 9)	+mn	LP
8.				B106.2	(backup)	+mb	LP
9.				B107	(backup)	+mb	LP
10.	B110-111	ND56	(AV R) → (AEQ R 7)	B110	(GET R)	≠n	LP,-C
		(=ND31)	(PC 6) → (GREATER R 5)				
11.	B114	ND57	(PC 6)	B11	(PC 6) → (GREATER R5)	=≠n	-C
12.			(no backup-up)	B114.1	(backup-up)	+mb	-C,LP
13.				B115	(PC 3) → (EVEN E)	=≠n	OM
14.	B116	ND58	(PC 5)	B116	(PC 5) → (EQ E 0)(?EQ C5 0)	=≠n	OM
15.				B117.1	(GET (EQ C5 0))	+mn	-C
16.	B117-119	ND59	(OP ?) → (SMALLER N 3)	B118	(PC 4) → (SMALLER N 3)	=≠n	OM
17.		ND60	(PC 3)			=≠n	OM
18.			(no backup)	B119.1	(backup)	+mb	GPBG
19.	B120-123	ND61	(FA (EQ E *D))	B120-122	(AV A) → (AEQ A x)	=≠n	-C,OM
20.				B123	(PC 3) → (EQ E y)	+mn	-C,OM

	PAS-I Analysis			Manual Analysis			
21.				B123.1	(PC 5) → (? 0)	+mn	-C,OM
22.	B126-129		(no backup)	B124	(backup)	+mb	GPBG
23.	B126-129	ND62	(AV 0) → (AEQ 0 9)	B125-127	(AV 0) → (AEQ 0 9)	=n	
24.		ND63	(AV G) → (AEQ G 2)			+pn	OM
25.		ND64	(PC 6) → (EQ C6 0)			+pn	OM
26.		ND65	(PC 5) → (EQ C5 0)			+pn	OM,GPBG
27.		ND66	(PC 5) → (EQ E 0)	B128	(PC 5) → (EQ E 0)	=n	
28.				B129	(PC 3) → (EQ A 5)	+mn	OM
29.	B130-135	ND67	(TD A 5) → (NEQ A 5)	B131	(TD A 5) → (NEQ A 5)	=n	
30.			(no backup)	B131.1	(backup)	+mb	GPBG
31.		ND68	(AV C4) → (AEQ C4 1)			+pn	OM
32.		ND69	(PC 3) → (COND ((EQ E 0))((EQ A 5)))	B132	(PC 3) → (EQ A 5)	≠n	Minor
33.		(=ND67)	(TD A 5) → (NEQ A 5)	B135	(TD A 5) → (NEQ A 5)	+mn	GPBG
34.			(no backup)	B136	(backup)	+mb	GPBG
35.	B138-141			B137-138	(PC 2) → (ODD R)	+mn	-C,GPBG
36.				B139	(PC 6) → (GREATER R 5)	+mn	-C,GPBG
		(=ND32)	(GN R) → (PEQ R 7)				
		(=ND33)	(GN R) → (PEQ R 9)				
37.		(=ND34)	(GN R) → (MEQ R 7 9)	B140-141	(GN R) → (MEQ R 7 9)	+mn	-C,GPBG
38.				B142-143	(AV R) → (AEQ R 7)(AEQ R 9)	+mn	-C,GPBG
				B143.1	(backup-up)		

II.4. Annotation of Disparities between PAS-I and Manual Analysis for S3 B101-143 on D+G=R.

1. B101-102. There is agreement on the occurrence of (PC 5), though PAS-I does not get much else.

(1) The error in order, putting ND53 (PC 5) after ND52 (TE) comes from the fact that PAS puts all hypothesized elements before the element for which there is direct information. PAS does not see any connection between the PC and the TE, so has no way to order them.

(2) The manual inference that (PC 5) → (PEQ E 0 9) is based on four pieces of evidence:

(a) (EVEN E) is obtained from (PC 3) (PAS gets this).

(b) B102 implies (PC 5) (PAS gets this).

(c) B103 implies S3 got a contradition, hence TD (PAS gets this too, although it calls it TE, rather than TD).

(d) The contradiction comes from (EVEN E). But why select out 9? Only if one had concluded from (PC 5) that E could be 9. Therefore the right inference was to (PEQ E 9). There is no evidence that the subject actually at that point derived (PEQ E 0 9). It is possible that he derived (EQ E 9), but this is not in accord with reality, nor with later acceptance of (EQ E 0). In fact, one can give an argument that (PEQ E 0 9) is probably wrong. If that had been the inference, then what should have followed was the inference to (EQ E 0). Instead, the subject backed way down.

(3) PAS hypothesizes the TE in order to account for the (NEQ E 9); however, it does not define TE (nor TD) in such a way as to require inputs to it, which would force it to see (PC 5) as generating some information. Hence (PC 5) hangs unattached.

(4) PAS does not in general ask the question: given that an operator occurred; what might it have produced that would fit into what is happening.

2. B103. TE is simply a more generalized version of TD (as it was originally defined for PAS); hence there is agreement here.

3. ND53. See discussion under line 1.

4. B103.1. This is an example where leaving the node at the end of the exploration is wrong. (AEQ L 9) is not done and then it is discovered that it conflicts with (AEQ L 3); 9 becomes free due to (NEQ E 9) and the situation is backed up to start over.

5-9. B106-109. The manual analysis argues as follows:

(1) (AEQ L 9) leads to processing column 2, (PC 2), which leads to (EQ R 9), which is in contradiction to (AEQ L 9), so that it will be seen by S3 via TD.

B106 RATHER -- is confirming evidence that the contradiction was seen.

(2) This is a peculiar contradiction, in that it leads to rejecting (EQ R 9), which then relieves the contradiction, making 9 free again. In addition the subject meant to assign 9 to R (AEQ R 9) and not to L. But R being 9 has just been implicated in a contradiction.

(3) All the above leaves the subject somewhat confused (no model of what it means internally to be confused, just a judgment that the situation is confusing).

B107 and B108 clearly indicate confusion on the subject's part, which is taken as confirming evidence.

The topic segments B106, B107 and B108 are too indefinite for the Linguistic Processor to obtain anything. Therefore, PAS simply attempts no other inferences. To obtain (PC 2) (TD R 9) it must work forward assuming that certain processing is obvious (see Newell and Simon, 1972, for a discussion of the notion of obviousness). This it is not yet prepared to do. To get it by working backward, it would have to detect a contradiction directly from RATHER and then try to work backwards to the origin. This is a conceivable route, but is not yet available.

10. B110-111. There are two failures here. One is that PAS does not have the concept of a goal, thus cannot obtain (GET R). Related to this is that the Linguistic Processor does not pick up the question character of B110, and so creates only (EQ R *D). The second error stems from the failure to get TD on (AEQ L 9); thus, PAS sees R as determined to be 7 (from PEQ R 7 9) and (AEQ L 9)). Thus hypothesizes an (AV R) where it should not have (and would not have).

11. B111-B114. Both PAS and the manual analysis get (PC 6), though PAS does not hang onto (GREATER R 5) but assimilates it back to a prior node. This reveals a general deficiency of PAS, that it does not consider that a subject may repeat a path that is still a separate processing. This relates to the recognition mechanism being too undiscriminating. The manual analysis glosses over the the slip at B113 (R + G) and sees B111-114 as a single occurrence of PC; PAS, on the other hand, takes this as a separation.

12. B114.1 The manual analysis sees that subject as having gone back, reviewed the derivation of R and then returned to the advanced point in the PBG. This kind of movement is not yet admitted by PAS. A small part of this is reflected in the recognition mechanism, but the most important part is simply that PAS still assumes continuous movement on the tree, with the only breaks being backing down.

13. B115. The manual analysis takes B115-117 together to infer (PC 3) and (PC 5), deciding on the order on the basis of B118-119. PAS gets both of these PC's

(at lines 17 and 14 respectively), but takes the order as it is given in the topic segments. Actually, the evidence isn't really good either way, e.g., it could have gone (PC 5), (PC 3), (PC 5) and then (PC 4) just as well as (PC 3) (PC 5) (PC 4). It is a little difficult to get from (PC 3) to (PC 4), since the connection is via C5 and the (PC 4) involves C5 as well.

14-15. B116-117.1. Both get the (PC 5). There is nothing direct in the topic segment that indicates the output of PC. The manual analysis works backward from B118: N < 3 implies C5 = 0 implies E = 0 implies this must have come from (PC 5). Since (below) PAS fails to get the operator, getting on (OP ?) instead of (PC 4), it does not run through the same path seeking the origin of the inputs to (PC 4) that produced (SMALLER N 3), etc.

16. B117-119. It is clear that PAS had available the information needed to get (PC 4), but currently PAS insists on stringent criteria of explanation. Given column 4 is the only column with N and given that an output was produced, PAS still will not propose (PC 4) if it cannot see exactly how the inference came about. This deficiency is easily corrected, though it is unclear how many errors of the other kind (stating a given column when it shouldn't be) would be generated, or whether it is possible to develop intermediate criteria.

PAS also declares (NEQ C5 1) to be of unknown origin. This is due to a lack of hypotheses about where negative information comes from. PAS should know (but doesn't) to try transforming NEQs on carries to EQs on their complements (if the values are 0 or 1).

17. ND60. See discussion on item 13.

18. B119.1. The manual analysis infers a backup from the language of B120 and the fact that nothing in the subsequent period rests on (EQ E 0), (EQ C5 0) or (SMALLER N 3). PAS misses the language entirely. This is first of all an deficiency in the Linguistic Processor, but more generally, there are no mechanisms yet in PAS for taking account of explicit backup indicators. That is,

PAS infers backup only from implicit information about contradictions and similarities in the PBG.

19-21. B120-123. The manual analysis interpreted the subject as going through a processing with symbolic variables, essentially a planning sort of operation. PAS found a way of saying the essential matter, namely, the subject's being concerned that A being equal to some digit is the determiner of E being equal to some digit: (FA (EQ E *D)) → (EQ A *D). This rates some credit. On the other hand, it clearly didn't get as far as the manual analysis, namely to see what processing was producing that, namely (PC 3). The assumption that (PC 5) also occurred is based on assuming that the subject will proceed with the processing he has been doing (and this is explicitly stated in B120). Actually, the utterances from B126-127 would appear to be support. PAS, as discussed earlier, does not run forward in time, and could only get to (PC 5) if something has issued from it that would have demanded an origin. This is a deficiency of the current scheme. A second concern about the behavior of PAS here is why it did not get (AV A), since B122 is quite clear. This goes back to a rule in the Semantic Processor that introduces (OPIO ...) from (ASSUME) (LETTER A), (THEREFORE) (LETTER E). This maintains the knowledge about assumption in B123, (AEQ A *D), which subsequently disappears when there is not way to determine the *D. (Recall that PAS perfers to throw away information if it cannot make absolutely definite sense out of it -- a somewhat extreme and inappropriate form of behavior.) It is not clear that the translation into (OPIO ...) is totally appropriate.

22. B124-129. For the manual analysis B124 provides direct linguistic indication of backup (I CAN DO BETTER THAN THAT), which, as noted already, PAS is insensitive to. In addition, for the manual analysis there is the repeated processing of (PC 5) and (PC 3) that occurs subsequently; these are in the opposite order

as before (when 3 was first, then 5), but this is to be explained by the reversal on what is assumed (A then, O here). PAS misses this, but does pick up the indication of inconsistency in the assignment of 9 and 0 when it is already assigned to L. Thus, it gets the backup, but for the wrong reason, since (AEQ L 9) should not be there.

23-27. B126-129. Both the manual analysis and PAS pick up the production of (EQ E 0) by (PC 5) (item 27). The manual analysis sees this as following directly upon (AEQ O 9). PAS, on the other hand, cannot generate (EQ E 0) from (PC 5) and (AEQ O 9) alone without hypothesizing the values of a carry, whose origin must then also be explained. This leads to a fantasy in which the key link is (EQ C6 0) (from which follows (EQ C5 0) by (PC 5)), involving (PC 6) and (AV G). Actually, PAS is right in not deriving (EQ E 0) and the manual analysis is deficient. The probably correct path is:

(PC 5) → (PEQ E 0 9)

(GN E) → (PEQ E 9)

(TD E 9) → (NEQ E 9) (because (EQ O 9)

(GN E) → (EQ E 0)

That is, mixed up in the processing of (PC 5) is the knowledge of exclusion of 9 for E from 9 from O and this is not part of the subprocess that is PC.

28-33. B130-135. The utterances are quite clear on what happened: the subject takes (EQ E 0) and processes column 3, getting (EQ A 5), which leads to a contradiction since (EQ D 5). The subject repeats the calculation, and this is done deliberately enough so there is little doubt that it is a repetition and not just a second reporting. Each step of the way has a topic segment clearly devoted to it. PAS has no difficulty with the general interpretation of the parts, but scrambles everything together. There are several deficiences involved.

(1) Currently TD does not insist that its negated element, here (NEQ A 5), be generated by some operator. In essence it is enough that from (EQ D 5) and TD (NEQ A 5) follows.

(2) There is no mechanism yet for detaching the conditional expression, given that its antecedent is satisfied.

(3) PAS simply puts together all the identical elements in the vicinity, so that the two occurrences of (NEQ A 5) are not taken as separate, and the occurrence of (PLUS A A) and (EQC (PLUS A A)) are not taken as separate. In general such order and repetition indifference is functional, since the subject often says things in varying orders and with inadvertent repetitions. Here the strategy backfires.

Items (1) and (2) are relatively minor; item (3) is more major, since changing the ordering and repetition rules will have any side effects.

34-38. B136-B141. PAS ends its PBG at the end of this branch, while the manual analysis sees a backup to a review of the derivation of R and then a backup-up to return to the consideration of E. PAS picks up the information from the text adequately, and sees the relations between its parts:

```
((BECAUSEOF ((ODD R) (GREATER R 5))((PEQ R 7))))
((BECAUSEOF ((ODD R) (GREATER R 5))((PEQ R 9))))
```

Its failure not to take this to the same conclusion as the manual analysis stem from two deficiencies:

(1) It does not see the use of (ODD R) and (GREATER R 5) as requiring regeneration. Thus, PAS does not have the idea of reviewing and recomputing data already in the PBG.

(2) Though it does imply the occurrence of (GN R) for the above, corresponding to the manual analysis (line 37), it recognizes it as ND34 and assimilates it into the tree. The identification is in fact correct, but, again, the lack of a notion of repeating a processing keeps it from becoming explicit.

APPENDIX III

III.1. PAS-I Linguistic and Semantic Processor Output for S4 B1-128 on D+G=R.

```
B1. EXP : NOW -- NOW THAT YOU ARE BECOMING SO -- SO GOOD AT THIS --
        (EXP)
B2. HU --
        (?)
B3. EXP : YOU MUST REMEMBER THAT WE WANT TO HEAR EVERYTHING THAT YOU THINK .
        (EXP)
B4. UM -- HM .
        (?)
B5. EXP : ESPECIALLY WHEN YOU DECIDE TO -- NOT TO DO SOMETHING AND GO ON TO SOMETHING ELSE .
        (EXP)
B6. YES .
        (YES)
B7. EXP : HERE IS ANOTHER PROBLEM . I WILL GIVE YOU THAT D IS 5 IN THIS PROBLEM . PLEASE TALK .
        (EXP)

     <B1-7>. ((?))


B8. WELL D --
        (LETTER D)

        <B8>. ((LETTER D))


B9. GIVING D 5
        (EQ D 5)
B10. AUTOMATICALLY MAKES T A' ZERO ,
        (THEREFORE) (EQ T 0)

     <B9-10>. ((BECAUSEOF ((EQ D 5)) ((EQ T 0))))


B11. COULD YOU MAKE T A' ZERO ?
        (EQ T 0) (QUES)

     <B11>. ((EQ T 0))


B12. EXP : T IS A' ZERO .
        (EXP)

     <B12>. ((?))


B13. BECAUSE 5 PLUS 5 IS EQUAL TO 10 ,
        (BECAUSE) (EQC (PLUS 5 5) 10)

     <B13>. ((BECAUSE) (EQC (PLUS 5 5) 10))


B14. AND THAT 'S SIMPLE FROM THE PROBLEM .
        (AND)
B15. AND LOOKING AT THE LEFT MOST COLUMN ,
        (AND) (PLACE LEFT)

     <B14-15>. ((?))


B16. YOU CAN SEE THAT R IS EITHER 1 OR 2 GREATER THAN D ,
        (MEQR R (PLUS D 1) (PLUS D 2))
        <B16>. ((MEQR R (PLUS D 1) (PLUS D 2)))


B17. BUT THAT DOES N'T SEEM TO HELP VERY MUCH AT THIS POINT .
        (NEG)

        <B17>. ((?))


B18. IN THE SECOND COLUMN HAVING THE TWO L 'S EQUAL ,
        (EQ L *D) (PLACE SECOND)

        <B18>. ((EQ L *D))


B19. AND ALSO THE TWO A 'S EQUAL IN THE THIRD COLUMN ,
        (AND) (EQC (PLUS A A) *D) (PLACE THIRD)

        <B19>. ((AND) (EQC (PLUS A A) *D))


B20. DOES N'T SEEM TO HELP TOO MUCH AT THIS POINT EITHER .
        (NEG)

        <B20>. ((?))


B21. KNOWING THAT THE TWO L 'S ARE EQUAL ,
        (EQC (PLUS L L) *D)

        <B21>. ((EQC (PLUS L L) *D))


B22. IF I JUST TOOK A' GUESS AT ONE OF THE L 'S ,
        (IF) (ASSUME) (LETTER L)

        <B22>. ((IF) (AEQ L *D))


B23. THIS --
        (?)
B24. THIS MIGHT GIVE ME SOME INSIGHT INTO HOW ABOUT --
        (?)
B25. HOW TO GO ABOUT THE PROBLEM .
        (?)

      <B23-25>. ((?))


B26. SO JUST TO GUESS L A' 3 COULD BE SOME HELP .
        (THEREFORE) (ASSUME) (EQ L 3)

        <B26>. ((THEREFORE) (AEQ L 3))


B27. COULD YOU MAKE L A' 3 ?
        (EQ L 3) (QUES)

        <B27>. ((EQ L 3))


B28. EXP : L IS 3 .
        (EXP)
```

```
          <B28>. ((?))


     B29. THAT WOULD MAKE R A' 6 .
               (THEREFORE) (EQ R 6)

          <B29>. ((THEREFORE) (EQ R 6))


     B30. WOULD YOU MAKE R A' 6 ?
               (EQ R 6) (QUES)

          <B30>. ((EQ R 6))


     B31. EXP : R IS 6 .
               (EXP)

          <B31>. ((?))


     B32. AND THAT MUST MAKE G A' 1 ,
               (AND) (THEREFORE) (EQ G 1)
     B33. BECAUSE G MUST BE EITHER A' ZERO
               (BECAUSE) (EQ G 0)
     B34. OR A' 1 ,
               (OR) (DIGIT 1)

       <B32-34>. ((BECAUSEOF ((MEQ G 0 1)) ((EQ G 1))))


     B35. FOR D PLUS G TO BE EQUAL TO R .
               (EQC (PLUS D G) R)

          <B35>. ((EQC (PLUS D G) R))


     B36. AND YOU KNOW THAT T IS ZERO .
               (AND) (EQ T 0)
     B37. SO THAT MAKES G A' 1 .
               (THEREFORE) (EQ G 1)

       <B36-37>. ((BECAUSEOF ((EQ T 0)) ((EQ G 1))))


     B38. EXP : G IS 1 .
               (EXP)

          <B38>. ((?))


     B39. AND TO HAVE O PLUS E EQUAL TO O ,
               (AND) (EQC (PLUS O E) O)

          <B39>. ((AND) (EQC (PLUS O E) O))


     B40. THIS MEANS THAT E IS 1 LESS --
               (EQR *L (PLUS E 1))

          <B40>. ((EQR *L (PLUS E 1)))


     B41. IT MEANS THAT E MUST BE 9 .
               (THEREFORE) (EQ E 9)
```

```
         <B41>. ((THEREFORE) (EQ E 9))


B42. WOULD YOU MAKE E A' 9 ?
          (EQ E 9) (QUES) .

         <B42>. ((EQ E 9))


B43. EXP : E IS 9 .
          (EXP)
B44. ALSO IN ROBERT .
          (?)

      <B43-44>. ((?))


B45. AND THAT MAKES THE GUESS OF L A' 3 A' BAD GUESS ,
          (AND) (NEG) (EQ L 3)

         <B45>. ((AND) (NEQ L 3))


B46. BECAUSE THE TWO A 'S CANNOT BE EQUAL TO 9
          (BECAUSE) (NEG) (EQC (PLUS A A) 9)

         <B46>. ((BECAUSE) (NEG) (EQC (PLUS A A) 9))


B47. SINCE THEY MUST BE A' WHOLE DIGIT .
          (BECAUSE)

         <B47>. ((?))


B48. SO L MUST BE SOME OTHER DIGIT GREATER THAN 5 ,
          (THEREFORE) (GREATER L 5)

         <B48>. ((THEREFORE) (GREATER L 5))


B49. IN ORDER TO HAVE THE TWO A 'S EQUAL TO 9 .
          (IF) (EQC (PLUS A A) 9)

         <B49>. ((IF) (EQC (PLUS A A) 9))


B58. THE TWO --
          (?)

         <B58>. ((?))


B51. A MUST BE 4
          (EQ A 4)

         <B51>. ((EQ A 4))


B52. IN ORDER TO HAVE THE TWO ADD TO 9 .
          (IF) (IN 9 *COL)

         <B52>. ((IF) (IN 9 *COL))
```

```
B53. WOULD YOU MAKE A A' 4 ?
          (EQ A 4) (QUES)

        <B53>. ((EQ A 4))


B54. EXP : A IS 4 .
          (EXP)

        <B54>. ((?))


B55. AND KNOWING THAT THE TWO L 'S ADD TO SOME NUMBER GREATER THAN 10 ,
          (AND) (GREATER (PLUS L L) 10)

        <B55>. ((AND) (GREATER (PLUS L L) 10))


B56. L MUST BE EITHER 6 ,
          (EQ L 6)
B57. 7 ,
          (DIGIT 7)
B58. OR 8 .
          (OR) (DIGIT 8)

     <B56-58>. ((PEQ L 6))

     <B56-58>. ((PEQ L 7))

     <B56-58>. ((PEQ L 8))

     <B56-58>. ((MEQ L 6 7 8))


B59. AND GIVEN THAT THE D IS 5 ,
          (AND) (EQ D 5)

        <B59>. ((AND) (EQ D 5))


B60. AND KNOWING THAT THE TWO O 'S --
          (AND) (NUM O 2)

        <B60>. ((AND) (NUM O 2))


B61. O MUST BE --
          (EQ O *D)

        <B61>. ((EQ D *D))


B62. YOU CAN N'T TELL --
          (NEG)

        <B62>. ((?))


B63. KNOWING THAT O PLUS 9 IS EQUAL TO O ,
          (EQC (PLUS O 9) O)

        <B63>. ((EQC (PLUS O 9) O))
```

```
B64. MUST MAKE N PLUS R SOME NUMBER GREATER THAN 10 ,
       (THEREFORE) (GREATER (PLUS N R) 10)

     <B64>. ((THEREFORE) (GREATER (PLUS N R) 10))


B65. IN ORDER TO MAKE THE FIFTH COLUMN COME OUT CORRECTLY .
       (IF) (PLACE FIFTH)

     <B65>. ((?))


B66. SO WE COULD TRY MAKING R --
       (THEREFORE) (ASSUME) (EQ R *D)

     <B66>. ((THEREFORE) (AEQ R *D))


B67. R MUST BE 1 GREATER THAN D --
       (EQR R (PLUS D 1))

     <B67>. ((EQR R (PLUS D 1)))


B68. NO ,
       (NEG)

     <B68>. ((?))


B69. R DOES N'T HAVE TO BE 1 GREATER THAN D .
       (NEG) (EQR R (PLUS D 1))

     <B69>. ((NEQR R (PLUS D 1)))


B70. R MUST BE SOME NUMBER BETWEEN 6 AND 9
       (BETW R 6 9)

     <B70>. ((PEQ R 6))

     <B70>. ((PEQ R 7))

     <B70>. ((PEQ R 8))

     <B70>. ((PEQ R 9))

     <B70>. ((MEQ R 6 7 8 9))


B71. AND THE OTHER --
       (AND)

     <B71>. ((?))


B72. THE ONLY ODD NUMBER IN THAT RANGE IS 7 .
       (EQ *L 7)
B73. SO COULD YOU CHANGE R TO A' 7 ?
       (THEREFORE) (EQ R 7) (QUES)

   <B72-73>. ((BECAUSEOF ((EQ *L 7)) ((EQ R 7))))


B74. EXP : CHANGE R FROM 6 TO 7 . R IS 7 .
```

```
           (EXP)
B75. CAN SEE NOW THAT THE FIRST THREE COLUMNS ARE CORRECT ,
           (?)
B76. BUT THAT DOES N'T NECESSARILY MEAN THAT THEY 'RE THE CORRECT DIGITS FOR THE PROBLEM .
           (NEG)

        <B74-76>. ((?))


B77. N NOW MUST BE --
           (EQ N *D)
B78. THE ONLY NUMBERS LEFT ARE 2 ,
           (DIGIT 2)
B79. 6 ,
           (DIGIT 6)
B80. AND 8 .
           (PLS 8)

        <B77-80>. ((PEQ N 2))

        <B77-80>. ((PEQ N 6))

        <B77-80>. ((PEQ N 8))

        <B77-80>. ((MEQ N 2 6 8))


B81. AND THEY MUST GO TO O ,
           (AND) (LETTER O)

           <B81>. ((AND) (LETTER O))


B82. N ,
           (LETTER N)

           <B82>. ((LETTER N))


B83. AND B .
           (PLS B)

           <B83>. ((LETTER B))


B84. I CAN NOW SEE AN ERROR IN THE THIRD COLUMN .
           (PLACE THIRD)

           <B84>. ((?))


B85. E ,
           (LETTER E)

           <B85>. ((LETTER E))


B86. COULD YOU CHANGE E TO AN 8 ,
           (EQ E 8)

           <B86>. ((EQ E 8))


B87. BECAUSE 4 PLUS 4 IS -- IS NOT EQUAL TO 9 .
           (BECAUSE) (NEG) (EQC (PLUS 4 4) 9)
        <B87>. ((BECAUSE) (NEG) (EQC (PLUS 4 4) 9))


B88. EXP : I WILL CHANGE E FROM 9 TO 8 . E IS NOW 8 .
        (EXP)

        <B88>. ((?))


B89. SO THE NUMBERS NOW REMAINING ARE 2 ,
        (THEREFORE) (REMAIN 2)
B90. 6 ,
        (DIGIT 6)
B91. AND 9 .
        (PLS 9)

     <B89-91>. ((THEREFORE) (REMAIN 2 6 9))


B92. AND THEY MUST GO TO THE LETT --
        (AND)

        <B92>. ((?))


B93. THE LETTERS O ,
        (LETTER O)

        <B93>. ((LETTER O))


B94. B ,
        (LETTER B)

        <B94>. ((LETTER B))


B95. AND N .
        (PLS N)

        <B95>. ((LETTER N))


B96. AND SEEING THAT O PLUS 8 IS EQUAL TO O --
        (AND) (EQC (PLUS O 8) O)

        <B96>. ((AND) (EQC (PLUS O 8) O))


B97. MUST MAKE E A' 9 ,
        (THEREFORE) (EQ E 9)

        <B97>. ((THEREFORE) (EQ E 9))


B98. FOR REASONING WHICH I GAVE PREVIOUSLY .
        (?)
B99. SO THAT MUST MAKE SOME OTHER DIGIT WRONG .
        (THEREFORE)

     <B98-99>. ((?))


B100. SO WE COULD CHANGE E BACK TO A' 9
```

```
             (THEREFORE) (EQ E 9)

          <B100>. ((THEREFORE) (EQ E 9))


 B101. AND MAKE L AN 8 .
             (AND) (EQ L 8)

          <B101>. ((AND) (EQ L 8))


 B102. COULD YOU CHANGE E TO 9 AGAIN ,
             (EQ E 9)

          <B102>. ((EQ E 9))


 B103. AND MAKE L AN 8
             (AND) (EQ L 8)

          <B103>. ((AND) (EQ L 8))


 B104. IN ORDER TO HAVE THE FIRST TWO COLUMNS COME OUT CORRECTLY ?
             (IF) (QUES)
 B105. EXP : CHANGE E FROM 8 TO 9 . E IS NOW 9 .
             (EXP)

       <B104-105>. ((?))


 B106. L FROM 3 TO 8 .
             (DIGIT 3) (DIGIT 8) (LETTER L)

          <B106>. ((DIGIT 3) (DIGIT 8) (LETTER L))


 B107. EXP : L FROM 3 TO 8 . L IS NOW 8 .
             (EXP)

          <B107>. ((?))


 B108. SO THE REMAINING DIGITS ARE NOW 2 ,
             (THEREFORE) (REMAIN 2)
 B109. 3 ,
             (DIGIT 3)
 B110. AND 6 .
             (PLS 6)

       <B108-110>. ((THEREFORE) (REMAIN 2 3 6))


 B111. IF N WERE 6 ,
             (IF) (EQ N 6)
 B112. THEN B WOULD BE 3 ,
             (THEN) (EQ B 3)
 B113. AND CARRYING 1 INTO THE NEXT COLUMN ,
             (AND) (EQ *C 1)

       <B111-113>. ((COND ((EQ N 6)) (AND ((EQ B 3)) ((EQ *C 1)))))


 B114. WOULD ONLY LEAVE 2 FOR THE O .
             (EQ O 2)
```

```
        <B114>. ((EQ O 2))


B115. AND 2 PLUS 9 IS EQUAL TO 11 ,
          (AND) (EQC (PLUS 2 9) 11)

        <B115>. ((AND) (EQC (PLUS 2 9) 11))


B116. PLUS THE CARRY IS EQUAL TO 12 ,
          (EQ *C 12)

        <B116>. ((EQ *C 12))


B117. MAKING THE SECOND O THE REQUIRED 2 .
          (EQ O 2)

        <B117>. ((EQ O 2))


B118. AND ADDING 1 TO THE LAST COLUMN ,
          (AND) (EQ *C 1)

        <B118>. ((AND) (EQ *C 1))


B119. AND MAKING THE PROBLEM COME OUT CORRECTLY .
          (AND)

        <B119>. ((?))


B120. SO IF YOU MAKE N A' 6 ,
          (THEREFORE) (IF) (EQ N 6)
B121. B A' 3 ,
          (EQ B 3)
B122. AND O A' 2 ,
          (AND) (EQ O 2)

      <B120-122>. ((COND ((EQ N 6)) (AND ((EQ B 3)) ((EQ O 2)))))


B123. I BELIEVE THAT SOLVES IT .
          (?)
B124. EXP : N IS 6 . B IS 3 . O IS 2 . IS THAT YOUR SOLUTION ?
          (EXP)
B125. YES .
          (YES)
B126. EXP : RIGHT . THAT IS THE CORRECT SOLUTION .
          (EXP)
B127. OK .
          (?)
B128. EXP : I AM GOING TO CHANGE THE TAPE .
          (EXP)

      <B123-128>. ((?))
```

III.2. PAS-I Group Processor and PBG Mechanism Output for S4 B1-128 on D+G=R.

```
B8-11 PROTOGROUP

ELEMENTS : ( ((LETTER D))
             ((BECAUSEOF ((EQ D 5)) ((EQ T 0))))
             ((EQ T 0)) )
OPERATOR : ()

    ORIGIN LIST : ()

    ORIGIN LIST : ( ((RECALL D) NIL (EQ D 5))
                    ((RECALL C1) NIL (EQ C1 0))
                    ((PC 1 H) ((EQ D 5) (EQ C1 0)) (EQ T 0)) )

        (RECALL D) GROWN AS ND1
        (EQ D 5) GROWN AS K1

        (RECALL C1) GROWN AS ND2
        (EQ C1 0) GROWN AS K2

        (PC 1) GROWN AS ND3
        (EQ T 0) GROWN AS K3

    ORIGIN LIST : ( (ND3 ((EQ C1 0) (EQ D 5)) (EQ T 0)) )

        ((PC 1) ((EQ C1 0) (EQ D 5)) (EQ T 0)) RECOGNIZED AS ND3


B13 PROTOGROUP

ELEMENTS : ( ((BECAUSE)) )
OPERATOR : (PC 1)   FROM (EQC (PLUS 5 5) 10)

    ORIGIN LIST : ()

        ((PC 1) NIL NIL) MERGED WITH ND3


B16 PROTOGROUP

ELEMENTS : ( ((MEQR R (PLUS D 1) (PLUS D 2))) )
OPERATOR : ()

    ORIGIN LIST : ( ((OP ?) NIL (MEQR R (PLUS D 1) (PLUS D 2))) )

        (OP ?) GROWN AS ND4
        (MEQR R (PLUS D 1) (PLUS D 2)) GROWN AS K4


B18-19 PROTOGROUP

ELEMENTS : ( ((EQ L *D))
             ((AND)) )
OPERATOR : (PC 3)   FROM (EQC (PLUS A A) *D)

    ((EQ L *D)) IS ((EQ L *D))

    ORIGIN LIST : ()

    ORIGIN LIST : ()
```

```
        (PC 3) GROWN AS ND5


B21-22 PROTOGROUP

ELEMENTS : ( ((IF) (AEQ L *D)) )
OPERATOR : (PC 2)   FROM (EQC (PLUS L L) *D)

    ((IF) (AEQ L *D)) IS ((IF) (AEQ L *D))

    ORIGIN LIST : ()

        (PC 2) GROWN AS ND6


B26-27 PROTOGROUP

ELEMENTS : ( ((THEREFORE) (AEQ L 3))
             ((EQ L 3)) )
OPERATOR : ()

    ORIGIN LIST : ( ((AV L H) NIL (AEQ L 3)) )

        (AV L) GROWN AS ND7
        (AEQ L 3) GROWN AS K5

    ORIGIN LIST : ( (ND7 NIL (AEQ L 3)) )

        ((AV L) NIL (AEQ L 3)) RECOGNIZED AS ND7


B29-30 PROTOGROUP

ELEMENTS : ( ((THEREFORE) (EQ R 6))
             ((EQ R 6)) )
OPERATOR : ()

    ORIGIN LIST : ( ((IG C2 H) NIL (EQ C2 0 H))
                    (ND7 NIL (AEQ L 3 H))
                    ((PC 2 H) ((EQ C2 0 H) (AEQ L 3 H)) (EQ R 6)) )

        (IG C2) GROWN AS ND8
        (EQ C2 0) GROWN AS K6

        ((AV L) NIL (AEQ L 3 H)) RECOGNIZED AS ND7

        (PC 2) GROWN AS ND9
        (EQ R 6) GROWN AS K7

    ORIGIN LIST : ( (ND9 ((AEQ L 3) (EQ C2 0)) (EQ R 6)) )

        ((PC 2) ((AEQ L 3) (EQ C2 0)) (EQ R 6)) RECOGNIZED AS ND9


B32-37 PROTOGROUP

ELEMENTS : ( ((BECAUSEOF ((MEQ G 0 1)) ((EQ G 1))))
             ((BECAUSEOF ((EQ T 0)) ((EQ G 1)))) )
OPERATOR : (PC 6)   FROM (EQC (PLUS D G) R)

    ORIGIN LIST : ( ((RECALL C7) NIL (EQ C7 0))
                    (ND1 NIL (EQ D 5))
                    ((PC 6) ((EQ C7 0) (EQ D 5) (EQ R 6)) (MEQ G 0 1))
                    ((GN G H) ((MEQ G 0 1) (EQ T 0)) (EQ G 1)) )
        (RECALL C7) GROWN AS ND10
        (EQ C7 0) GROWN AS K8

        ((RECALL D) NIL (EQ D 5)) RECOGNIZED AS ND1

        (PC 6) GROWN AS ND11
        (MEQ G 0 1) GROWN AS K9

        (GN G) GROWN AS ND12
        (EQ G 1) GROWN AS K10

     ORIGIN LIST : ( (ND12 ((EQ T 0) (MEQ G 0 1)) (EQ G 1)) )

        ((GN G) ((EQ T 0) (MEQ G 0 1)) (EQ G 1)) RECOGNIZED AS ND12

  B39-42 PROTOGROUP

  ELEMENTS : ( ((AND))
               ((EQR *L (PLUS E 1)))
               ((THEREFORE) (EQ E 9))
               ((EQ E 9)) )
  OPERATOR : (PC 5)   FROM (EQC (PLUS O E) O)

     ORIGIN LIST : ()

     ((EQR *L (PLUS E 1))) IS ((EQR *L (PLUS E 1)))

     ORIGIN LIST : ()

     ORIGIN LIST : ( ((AV E H) NIL (AEQ E 9)) )

        (AV E) GROWN AS ND13
        (AEQ E 9) GROWN AS K11

     ORIGIN LIST : ( (ND13 NIL (AEQ E 9)) )

        ((AV E) NIL (AEQ E 9)) RECOGNIZED AS ND13

        (PC 5) GROWN AS ND14

 B45-46 PROTOGROUP

 ELEMENTS : ( ((AND) (NEQ L 3))
              ((BECAUSE) (NEG)) )
 OPERATOR : (PC 3)   FROM (EQC (PLUS A A) 9)

    ORIGIN LIST : ( ((PC 3) ((EQ E 9)) (EQ C3 1 H))
                    ((PC 2 H) ((EQ C3 1 H)) (NEQ L 3)) )

       (PC 3) GROWN AS ND15
       (EQ C3 1) GROWN AS K12

       (PC 2) GROWN AS ND16
       (NEQ L 3) GROWN AS K13

       PBG CONFLICT : (NEQ L 3) VS (AEQ L 3)

    ORIGIN LIST : ()

B48-49 PROTOGROUP

ELEMENTS : ( ((THEREFORE) (GREATER L 5))
          ((IF)) )
OPERATOR : (PC 3)   FROM (EQC (PLUS A A) 9)

    ORIGIN LIST : ( ((OP ?) NIL (GREATER L 5)) )

        (OP ?) GROWN AS ND20
        (GREATER L 5) GROWN AS K14

    ORIGIN LIST : ()

        ((PC 3) NIL NIL) MERGED WITH ND18


B51-53 PROTOGROUP

ELEMENTS : ( ((EQ A 4))
             ((IF) (IN 9 *COL))
             ((EQ A 4)) )
OPERATOR : ()

    ORIGIN LIST : ( ((IG C4 H) NIL (EQ C4 0 H))
                    ((PC 3 H) ((EQ C4 0 H) (EQ E 9) (EQ C3 1)) (EQ A 4)) )

        (IG C4) GROWN AS ND21
        (EQ C4 0) GROWN AS K15

        (PC 3) GROWN AS ND22
        (EQ A 4) GROWN AS K16

    ((IF) (IN 9 *COL)) IS ((IF) (IN E 5))

    ORIGIN LIST : ( ((FC E H) NIL (IN E 5)) )

        (FC E) GROWN AS ND23
        (IN E 5) GROWN AS K17

    ORIGIN LIST : ( (ND22 ((EQ C3 1) (EQ E 9) (EQ C4 0)) (EQ A 4)) )

        ((PC 3) ((EQ C3 1) (EQ E 9) (EQ C4 0)) (EQ A 4)) RECOGNIZED AS ND22


B55-59 PROTOGROUP

ELEMENTS : ( ((AND))
             ((PEQ L 6))
             ((PEQ L 7))
             ((PEQ L 8))
             ((MEQ L 6 7 8))
             ((AND) (EQ D 5)) )
OPERATOR : (PC 2)   FROM (GREATER (PLUS L L) 10)

    ORIGIN LIST : ()

    ORIGIN LIST : ( ((GN L H) ((NEQ L 3) (GREATER L 5)) (PEQ L 6)) )

        (GN L) GROWN AS ND24
        (PEQ L 6) GROWN AS K18

    ORIGIN LIST : ( ((GN L H) ((NEQ L 3) (GREATER L 5) (PEQ L 6)) (PEQ L 7)) )

        (GN L) GROWN AS ND25
        (PEQ L 7) GROWN AS K19

    ORIGIN LIST : ( ((GN L H) ((NEQ L 3) (GREATER L 5) (PEQ L 6) (PEQ L 7)) (PEQ L 8)) )
```

```
          (GN L) GROWN AS ND26
          (PEQ L 8) GROWN AS K20

      ORIGIN LIST : ( ((GN L H) ((PEQ L 6) (PEQ L 7) (PEQ L 8)) (MEQ L 6 7 8)) )

          (GN L) GROWN AS ND27
          (MEQ L 6 7 8) GROWN AS K21

      ORIGIN LIST : ( (ND1 NIL (EQ D 5)) )

          ((RECALL D) NIL (EQ D 5)) RECOGNIZED AS ND1

          (PC 2) GROWN AS ND28

 B60-61 PROTOGROUP

 ELEMENTS : ( ((AND))
              ((EQ O *D)) )
 OPERATOR : (PC 5)   FROM (NUM O 2)

     ORIGIN LIST : ()

     ((EQ O *D)) IS ((EQ O *D))

     ORIGIN LIST : ()

         (PC 5) GROWN AS ND29

 B63 PROTOGROUP

 ELEMENTS : ()
 OPERATOR : (PC 5)   FROM (EQC (PLUS O 9) 0)

         ((PC 5) NIL NIL) MERGED WITH ND29

 B64 PROTOGROUP

 ELEMENTS : ( ((THEREFORE)) )
 OPERATOR : (PC 4)   FROM (GREATER (PLUS N R) 10)

     ORIGIN LIST : ()

         (PC 4) GROWN AS ND30

 B66-67 PROTOGROUP

 ELEMENTS : ( ((THEREFORE) (AEQ R *D))
              ((EQR R (PLUS D 1))) )
 OPERATOR : ()

     ((THEREFORE) (AEQ R *D)) IS ((THEREFORE) (AEQ R *D))

     ORIGIN LIST : ()

     ORIGIN LIST : ( ((OP ?) NIL (EQR R (PLUS D 1))) )

         (OP ?) GROWN AS ND31
         (EQR R (PLUS D 1)) GROWN AS K22

 B69-70 PROTOGROUP
```

```
ELEMENTS : ( ((NEQR R (PLUS D 1)))
             ((PEQ R 6))
             ((PEQ R 7))
             ((PEQ R 8))
             ((PEQ R 9))
             ((MEQ R 6 7 8 9)) )
OPERATOR : ()

    ORIGIN LIST : ( ((OP ?) NIL (NEQR R (PLUS D 1))) )

        (OP ?) GROWN AS ND32
        (NEQR R (PLUS D 1)) GROWN AS K23

        PBG CONFLICT : (NEQR R (PLUS D 1)) VS (EQR R (PLUS D 1))

    ORIGIN LIST : ( ((RECALL C7) NIL (EQ C7 8))
                    (ND1 NIL (EQ D 5))
                    ((PC 6 H) ((EQ C7 8) (EQ D 5)) (PEQ R 6)) )

        (RECALL C7) GROWN AS ND34
        (EQ C7 8) GROWN AS K24

        ((RECALL D) NIL (EQ D 5)) RECOGNIZED AS ND1

        (PC 6) GROWN AS ND35
        (PEQ R 6) GROWN AS K25

    ORIGIN LIST : ( (ND1 NIL (EQ D 5))
                    (ND34 NIL (EQ C7 8))
                    ((PC 6 H) ((EQ D 5) (EQ C7 8)) (PEQ R 7)) )

        ((RECALL D) NIL (EQ D 5)) RECOGNIZED AS ND1

        ((RECALL C7) NIL (EQ C7 8)) RECOGNIZED AS ND34

        ((PC 6 H) ((EQ D 5) (EQ C7 8)) (PEQ R 7)) MERGED WITH ND35
        (PEQ R 7) GROWN AS K26

    ORIGIN LIST : ( (ND1 NIL (EQ D 5))
                    (ND34 NIL (EQ C7 8))
                    ((PC 6 H) ((EQ D 5) (EQ C7 8)) (PEQ R 8)) )

        ((RECALL D) NIL (EQ D 5)) RECOGNIZED AS ND1

        ((RECALL C7) NIL (EQ C7 8)) RECOGNIZED AS ND34

        ((PC 6 H) ((EQ D 5) (EQ C7 8)) (PEQ R 8)) MERGED WITH ND35
        (PEQ R 8) GROWN AS K27

    ORIGIN LIST : ( (ND1 NIL (EQ D 5))
                    (ND34 NIL (EQ C7 8))
                    ((PC 6 H) ((EQ D 5) (EQ C7 8)) (PEQ R 9)) )

        ((RECALL D) NIL (EQ D 5)) RECOGNIZED AS ND1

        ((RECALL C7) NIL (EQ C7 8)) RECOGNIZED AS ND34

        ((PC 6 H) ((EQ D 5) (EQ C7 8)) (PEQ R 9)) MERGED WITH ND35
        (PEQ R 9) GROWN AS K28

    ORIGIN LIST : ( ((GN R H) ((PEQ R 6) (PEQ R 7) (PEQ R 8) (PEQ R 9)) (MEQ R 6 7 8 9)) )

        (GN R) GROWN AS ND36
        (MEQ R 6 7 8 9) GROWN AS K29
```

```
B72-73 PROTOGROUP

ELEMENTS : ( ((BECAUSEOF ((EQ *L 7)) ((EQ R 7)))) )
OPERATOR : ()

    ((BECAUSEOF ((EQ *L 7)) ((EQ R 7)))) IS
    ((BECAUSEOF ((EQ R 7)) ((EQ R 7))))

    ORIGIN LIST : ( ((AV R H) ((MEQ R 6 7 8 9)) (AEQ R 7)) )

        (AV R) GROWN AS ND37
        (AEQ R 7) GROWN AS K38


B77-83 PROTOGROUP

ELEMENTS : ( ((PEQ N 2))
             ((PEQ N 6))
             ((PEQ N 8))
             ((MEQ N 2 6 8))
             ((AND) (LETTER O))
             ((LETTER N))
             ((LETTER B)) )
OPERATOR : ()

    ORIGIN LIST : ( ((GN N H) NIL (PEQ N 2)) )

        (GN N) GROWN AS ND38
        (PEQ N 2) GROWN AS K31

    ORIGIN LIST : ( ((GN N H) NIL (PEQ N 6)) )

        ((GN N H) NIL (PEQ N 6)) MERGED WITH ND38
        (PEQ N 6) GROWN AS K32

    ORIGIN LIST : ( ((GN N H) NIL (PEQ N 8)) )

        ((GN N H) NIL (PEQ N 8)) MERGED WITH ND38
        (PEQ N 8) GROWN AS K33

    ORIGIN LIST : ( ((GN N H) ((PEQ N 2) (PEQ N 6) (PEQ N 8)) (MEQ N 2 6 8)) )

        (GN N) GROWN AS ND39
        (MEQ N 2 6 8) GROWN AS K34

    ORIGIN LIST : ()

    ORIGIN LIST : ()

    ORIGIN LIST : ()


B85-87 PROTOGROUP

ELEMENTS : ( ((LETTER E))
             ((EQ E 8))
             ((BECAUSE) (NEG)) )
OPERATOR : (PC 3)   FROM (EQC (PLUS 4 4) 9)

    ORIGIN LIST : ()

    ORIGIN LIST : ( ((AV E H) NIL (AEQ E 8)) )
```

```
          (AV E) GROWN AS ND40
          (AEQ E 8) GROWN AS K35

          PBG CONFLICT : (AEQ E 8) VS (AEQ E 9)

      ORIGIN LIST : ()

          (PC 3) GROWN AS ND42


B89-91 PROTOGROUP

ELEMENTS : ( ((THEREFORE) (REMAIN 2 6 9)) )
OPERATOR : ()

      ORIGIN LIST : ( ((OP ?) NIL (REMAIN 2 6 9)) )

          (OP ?) GROWN AS ND43
          (REMAIN 2 6 9) GROWN AS K36


B93-97 PROTOGROUP

ELEMENTS : ( ((LETTER O))
             ((LETTER B))
             ((LETTER N))
             ((AND))
             ((THEREFORE) (EQ E 9)) )
OPERATOR : (PC 5)   FROM (EQC (PLUS O 8) O)

      ORIGIN LIST : ()

      ORIGIN LIST : ()

      ORIGIN LIST : ()

      ORIGIN LIST : ()

      ORIGIN LIST : ( ((AV E H) NIL (AEQ E 9)) )

          (AV E) GROWN AS ND44
          (AEQ E 9) GROWN AS K37

          PBG CONFLICT : (AEQ E 9) VS (AEQ E 8)

          (PC 5) GROWN AS ND46


B100-103 PROTOGROUP

ELEMENTS : ( ((THEREFORE) (EQ E 9))
             ((AND) (EQ L 8))
             ((EQ E 9))
             ((AND) (EQ L 8)) )
OPERATOR : ()

      ORIGIN LIST : ( (ND45 NIL (AEQ E 9)) )

          ((AV E) NIL (AEQ E 9)) RECOGNIZED AS ND45

      ORIGIN LIST : ( ((AV L H) NIL (AEQ L 8)) )

          (AV L) GROWN AS ND47
          (AEQ L 8) GROWN AS K38
```

```
     ORIGIN LIST : ( (ND45 NIL (AEQ E 9)) )

         ((AV E) NIL (AEQ E 9)) RECOGNIZED AS ND45

     ORIGIN LIST : ( (ND47 NIL (AEQ L 8)) )

         ((AV L) NIL (AEQ L 8)) RECOGNIZED AS ND47


B106 PROTOGROUP

ELEMENTS : ( ((DIGIT 3) (DIGIT 8) (LETTER L)) )
OPERATOR : ()

     ORIGIN LIST : ()


B108-118 PROTOGROUP

ELEMENTS : ( ((THEREFORE) (REMAIN 2 3 6))
             ((COND ((EQ N 6)) (AND ((EQ B 3)) ((EQ *C 1)))))
             ((EQ O 2))
             ((AND))
             ((EQ *C 12))
             ((EQ O 2))
             ((AND) (EQ *C 1)) )
OPERATOR : (PC *COL)   FROM (EQC (PLUS 2 9) 11)

     ORIGIN LIST : ( ((OP ?) NIL (REMAIN 2 3 6)) )

         (OP ?) GROWN AS ND48
         (REMAIN 2 3 6) GROWN AS K39

     ((COND ((EQ N 6)) (AND ((EQ B 3)) ((EQ *C 1))))) IS
     ((COND ((EQ N 6)) ((EQ B 3) (EQ C5 1))))

     ORIGIN LIST : ( (ND1 NIL (EQ D 5))
                     (ND2 NIL (EQ C1 0))
                     ((PC 1 H) ((EQ D 5) (EQ C1 0)) (EQ C2 1 H))
                     ((PC 2 H) ((EQ C2 1 H) (EQ L 8)) (EQ R 7 H))
                     ((IG C4 H) NIL (EQ C4 0 H))
                     ((PC 4 H) ((EQ R 7 H) (EQ C4 0 H)) (COND ((EQ N 6)) (EQ B 3)))
                     ((PC 5 H I) NIL (COND ((EQ N 6)) ((EQ C5 1)))) )

         ((RECALL D) NIL (EQ D 5)) RECOGNIZED AS ND1

         ((RECALL C1) NIL (EQ C1 0)) RECOGNIZED AS ND2

         (PC 1) GROWN AS ND49
         (EQ C2 1) GROWN AS K40

         (PC 2) GROWN AS ND50
         (EQ R 7) GROWN AS K41

         (IG C4) GROWN AS ND51
         (EQ C4 0) GROWN AS K42

         (PC 4) GROWN AS ND52
         (COND ((EQ N 6)) (EQ B 3)) GROWN AS K43

         (PC 5 H I) GROWN AS ND53
         (COND ((EQ N 6)) ((EQ C5 1))) GROWN AS K44

    ORIGIN LIST : ( ((PC 4 H) ((EQ R 7) (EQ C4 0)) (COND ((EQ N 6)) (EQ B 3)))
                    ((PC 4 H) ((EQ R 7) (EQ C4 0)) (COND ((EQ N 6)) (EQ C5 1))) )
```

```
        ((PC 4 H) ((EQ R 7) (EQ C4 0)) (COND ((EQ N 6)) (EQ B 3))) MERGED WITH ND52
        (COND ((EQ N 6)) (EQ B 3)) GROWN AS K45

        ((PC 4 H) ((EQ R 7) (EQ C4 0)) (COND ((EQ N 6)) (EQ C5 1))) MERGED WITH ND52
        (COND ((EQ N 6)) (EQ C5 1)) GROWN AS K46

    ORIGIN LIST : ( ((AV O H) NIL (AEQ O 2)) )

        (AV O) GROWN AS ND54
        (AEQ O 2) GROWN AS K47

    ORIGIN LIST : ()

    ((EQ *C 12)) IS ((EQ C6 1))

    ORIGIN LIST : ( ((PC 5 H) ((EQ E 9) (EQ O 2)) (EQ C6 1)) )

        (PC 5) GROWN AS ND55
        (EQ C6 1) GROWN AS K48


B128-122 PROTOGROUP

ELEMENTS : ( ((COND ((EQ N 6)) (AND ((EQ B 3)) ((EQ O 2))))) )
OPERATOR : ()

    ORIGIN LIST : ( ((PC 4 H) ((EQ R 7) (EQ C4 0)) (COND ((EQ N 6)) (EQ B 3)))
                    (ND54 NIL (COND ((EQ N 6)) ((EQ O 2)))) )

        ((PC 4 H) ((EQ R 7) (EQ C4 0)) (COND ((EQ N 6)) (EQ B 3))) MERGED WITH ND52
        (COND ((EQ N 6)) (EQ B 3)) GROWN AS K49

        ((AV O) NIL (COND ((EQ N 6)) ((EQ O 2)))) RECOGNIZED AS ND54


*** PAS-I FINISHED ***
```

APPENDIX III

III.3. Comparison between PBG of PAS-I and Manual Analysis for S4 B1-128 on D+G=R.

	PAS-I Analysis			Manual Analysis		Agreement
1.				B1-7	(Exp instructions)	x
2.	B8-11	ND1	(RECALL D) → (EQ D 5)	B8-14		s
3.		ND2	(RECALL C1) → (EQ C1 0)			s
4.		ND3	(PC 1) → (EQ T 0)		(PC 1) → (EQ T 0)	=n
5.	B16	ND4	(OP ?) → (MEQR (+ D 1) (+ D 2))	B15-17	(PC 6) → (MEQ (+ D 1) (+ D 2))	≠n
6.				B18	(PC 2)	+mn
7.	B18-19	ND5	(PC 3)	B19-20	(PC 3)	=n
8.			(no backup)		(backup)	+mb
9.	B21-22	ND6	(PC 2)	B21	(PC 2)	=n
10.				B22	(AV L)	+mn
11.			(no backup)		(backup)	+mb
12.	B26-27	ND7	(AV L) → (AEQ L 1)	B26-27	(AV L) → (AEQ L 1)	=n
13.	B29-30	ND8	(IG C2) → (EQ C2 0)	B29-30		s
14.		ND9	(PC 2) → (EQ R 6)		(PC 2) → (EQ R 6)	=n
15.	B32-37	ND10	(RECALL C7) → (EQ C7 0)	B32		s
16.					(PC 6) → (EQ G 1)	+mn
17.			(no backup)		(backup)	+mb
18.		ND11	(PC 6) → (MEQ G 0 1)	B33	(PC 6) → (MEQ G 0 1)	=n
19.					(GN G) → (AEQ G 0)	+mn
20.				B36	(TD G 0) → (NEQ G 0)	+mn
21.			(no backup)		(backup)	+mb
22.		ND12	(GN G) → (EQ G 1)	B37	(GN G) → (EQ G 1)	=n

		PAS-I Analysis		Manual Analysis		Agreement
23.	B39-42	ND13	(AV E) → (AEQ E 9)	B39-42		+pn
24.		ND14	(PC 5)		(PC 5) → (EQ E 9)	=≠n
25.	B45-46	ND15	(PC 3) → (EQ C3 1)	B44	(PC 3) → (NEQ C3 0)	=≠n
26.		ND16	(PC 2) → (NEQ L 3)	B45	(FA C3) → (NEQ L 3)	≠n
27.			(backup)		(backup)	=b
28.		ND17	= ND13		(PC 5) → (EQ E 9)	x
29.		ND18	= ND15		(PC 3) → (EQ C3 1)	x
30.		ND19	= ND16			x
31.	B48-49	ND20	(OP ?) → (GR L 5)	B48	(PC 2) → (GR L 5)	≠n
32.	B51-53	ND21	(IG C4) → (EQ C4 0)	B50-53		s
33.		ND22	(PC 3) → (EQ A 4)		(PC 3) → (EQ A 4)	=n
34.		ND23	(FC E) → (IN E 5)			+pn
35.	B55-59	ND24	(GN L) → (PEQ L 6)	B55-58		s
36.		ND25	(GN L) → (PEQ L 7)			s
37.		ND26	(GN L) → (PEQ L 8)			s
38.		ND27	(GN L) → (MEQ L 6 7 8)		(GN L) → (MEQ L 6 7 8)	=n
39.		ND28	(PC 2)			+pn
40.				B59	(PC 6)	+mn
41.	B60-63	ND29	(PC 5)	B60	(PC 5) → (FREE 0)	=≠n
42.			(no backup)		(backup)	+mb
43.				B63	(PC 5) → (EQ C5 1)	+mn
44.	B64	ND30	(PC 4)	B64	(PC 4) → (GET R)	=≠n
45.				B66	(AV R)	+mn
46.			(no backup)		(backup)	+mb
47.	B66-67	ND31	(OP ?) → (EQR R (+ D 1))	B67	(PC 6) → (EQ R (+ D 1))	≠n

	PAS-I Analysis			Manual Analysis		Agreement
48.	B69-70	ND32	(OP ?) → (NEQR R (+ D 1))			s
49.			(backup)	(backup)		=b
50.		ND33	=ND32			x
51.		ND34	(RECALL C7) → (EQ C7 0)			s
52.		ND35	(PC 6) → (PEQ R 6) (PEQ R 7) (PEQ R 8) (PEQ R 9)	B70	(PC6) → (MEQ R 6 7 8 9)	=n
53.		ND36	(GN R) → (MEQ R 6 7 8 9)			s
54.				B71	(PC 2) → (ODD R)	+mn
55.	B72-73	ND37	(AV R) → (AEQ R 7)	B72-73	(GN R) → (EQ R 7)	≠n
56.				B75-76	(PC 1) → OK	+mn
57.					(PC 2) → OK	+mn
58.					(PC 3) → OK	+mn
59.				B77	(PC 4) → (GET N)	+mn
60.	B77-83	ND38	(GN N) → (PEQ N 2) (PEQ N 6) (PEQ N 8)			s
61.		ND39	(GN N) → (MEQ N 2 6 8)	B78	(GN FREE) → (REMAIN 2 6 8)	=≠n
62.				B81	(AV O N B)	+mn
63.					(PC 4)	+mn
64.	B85-87	ND40	(AV E) → (AEQ E 8)	B84	(PC 3) → (NEQ E 9)	=≠n
65.			(backup)		(backup)	=b
66.		ND41	= ND40			x
67.		ND42	(PC 3)	B85-86	(PC 3) → (EQ E 8)	=≠n
68.	B89-91	ND43	(OP ?) → (REMAIN 2 6 9)	B89-91	(GN FREE) → (REMAIN 2 6 9)	≠n
69.				B92-95	(AV O N B)	+mn
70.	B93-97	ND44	(AV E) → (AEQ E 9)	B96	(PC 5) → (EQ E 9)	=≠n
71.				B97	(TD E 9) → (NEQ E 8)	+mn

		PAS-I Analysis		Manual Analysis		Agreement
72.				B99	(FA E)	+mn
73.			(backup)		(backup)	=b
74.		ND45	= ND44			x
75.		ND46	(PC 5)			=≠n
76.				B100	(PC 3) → (EQ E 9) (EQ C3 1)	+mn
77.	B100-103	ND47	(AV L) → (AEQ L 8)	B101	(PC 2) → (EQ L 8)	≠n
78.	B108-118	ND48	(OP ?) → (REMAIN 2 3 6)	B108-110	(GN FREE) → (REMAIN 2 3 6)	≠n
79.		ND49	(PC 1) → (EQ C2 1)			+pn
80.		ND50	(PC 2) → (EQ R 7)			+pn
81.		ND51	(IG C4) → (EQ C4 0)			+pn
82.				B111	(AV N) → (AEQ N 6)	+mn
83.		ND52	(PC 4) → (COND (EQ N 6) (EQ B 3)) (COND (EQ N 6) (EQ C5 1))	B112-113	(PC 4) → (EQ B 3)	=≠n
84.		ND53	(PC 5)			+pn
85.		ND54	(AV O) → (AEQ O 2)	B114	(GN O) → (EQ O 2)	≠n
86.		ND55	(PC 5) → (EQ C6 1)	B115-117	(PC 5) → (EQ O 2)(EQ C6 1)	=n
87.				B118	(PC 6) → OK	+mn

Security Classification

DOCUMENT CONTROL DATA - R & D

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

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

3. REPORT TITLE
Preliminary Results with a System for Automatic Protocol Analysis

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

5. AUTHOR(S) *(First name, middle initial, last name)*
Donald A. Waterman and Allen Newell

6. REPORT DATE May 1972	7a. TOTAL NO. OF PAGES 181	7b. NO. OF REFS 27

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

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

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

13. ABSTRACT

A computer program for automatic protocol analysis (PAS-I) is described in detail. The task of protocol analysis is that of inferring from the verbalizations given by a human while solving a problem the time course of his states of knowledge about the task. PAS-I works only with the task domain of cryptarithmetic puzzles and incorporates a specific theory of problem solving (as described in Newell and Simon, Human Problem Solving). The input to PAS-I is the transcription of the human's verbalizations (as tape recorded), segmented into topics. The program does a linguistic analysis from the input text to produce a sequence of semantic elements; these are then subjected to several stages of processing, including hypothesizing the information processing operations that the subject performed to produce the semantic knowledge that he appears to have at each moment of time. The final output is a problem behavior graph (PBG) which is the trajectory of the subject through the problem space of possible knowledge states. The performance of PAS-I on three examples of behavior is presented and analysed: 100 topic segments of subject S1 on the cryptarithmetic task, DONALD+GERALD=ROBERT; 43 further topic segments on S1 on the same task; and 128 topic segments (the entire session) of subject S4 on DONALD+GERALD.