STUDIES IN PROBLEM SOLVING:
SUBJECT 3 ON THE CRYPT-ARITHMETIC TASK DONALD + GERALD = ROBERT

Allen Newell
July, 1967

 The research was supported by Research Grant MH-07722-01 from the National Institutes of Health.

Carnegie Institute of Technology

Studies in Problem Solving: Subject 3 on the Crypt-arithmetic*

Task DONALD + GERALD = ROBERT

Allen Newell

This paper provides another item in a larger effort to develop information processing theories of human problem solving. The conceptual foundations of the theory stem from work in constructing computer programs to accomplish tasks requiring intelligence. Since the general issues, methodology, and prior work in applying these tools to theories of human behavior have been covered elsewhere by several investigators, we will only bring out those aspects necessary to put the current work in context.**

Information processing theories lend themselves to the development of programs that simulate segments of individual behavior. Typically, a subject is put to a task requiring complex reasoning -- playing chess, proving theorems, discovering complicated concepts -- and asked to think aloud while he works. The raw data available for analysis, usually called the protocol, consists of a tape recording of the subject's verbal behavior plus other notations necessary to record pertinent behavior, such as writing. The protocol includes the experimenter's behavior as well, although his participation is usually minimized intentionally.

---

* I am indebted to my colleague H. A. Simon for his contribution to this work. This research was supported by Research Grant MH-0772201 from the National Institutes of Health.

** The collection of papers in Feigenbaum and Feldman [7] provides many examples of work in artificial intelligence and computer simulation of cognitive processes. More extended discussion of the methodology is to be found in Reitman [24], Newell and Simon [20] and Miller, Galanter and Pribram [12]. The current study is used extensively in a more general paper on protocol analysis [15].

The task of data analysis consists of at least two steps. First is the production of the transcript from the raw recording and measurements. This invariably loses information, since the total range of verbal behavior is not rendered with fidelity by standard secretarial-level transcription, even when done carefully. Pauses, pace, intonations, and much paralinguistic material are lost. Since no one has yet concerned himself with this aspect of the data analysis in connection with simulation studies, we will have nothing more to say about it here directly (but see [23]). Even without it, a protocol of a typical session lasting a quarter of an hour provides a wealth of data.

The second stage of analysis is the production of a computer program that "simulates" the behavior as revealed in the protocol. Simulation means putting the behavior of the program (whose record is often called the trace) in one-one correspondence, at some level, with the human's behavior.

There are difficulties in assessing this correspondence. The most important relates to qualitiative comparison on task information. In chess, for example, both trace and protocol identify not only moves made or considered, but features of the board that are noticed and evaluated, decision points in the analysis, etc. Numerical measures are not easily adequate to these comparisons, since the full range of task content is involved. This difficulty is a direct consequence of the richness of the theory -- of the kinds of things it is able to assert about the human behavior. Still, it makes the apparatus of statistical testing difficult to apply.

There are other difficulties as well. Errors in correspondence are of different sorts. One expects the verbal output to be uneven -- to pass in silence over many things of importance. Therefore, not everything in the trace

would be expected to show in the protocol. Also, one must specify (but, very seldom does precisely) at what level the program's behavior is to be taken seriously. As one moves from gross behavior towards machine code, both the organization and the instructions reflect increasingly the structure of current programming languages (and ultimately, computers).

The paradigm presented above suggests the development of a unique program to correspond to each protocol. In truth, of course, one wishes to postulate a common set of mechanisms to describe the behavior of the same person over many tasks, or even to describe the behavior of many people. This communality is to be tempered by the fact that large individual differences do exist, and that it may be possible to evoke quite distinct sets of behaviors from the same person by changing the situation sufficiently. In any event, there have been some attempts to develop general program structures, within which individual variants can be formed to account for individual protocols [9, 19]. A difficulty plagues the analyst in doing this. Programs do not lend themselves to parameterization in the same way that standard mathematical systems do. More precisely, the variations cannot always be represented easily by replacing a constant by a variable in the expressions of the theory, where the variable ranges over a simple domain (such as the real numbers). In programs the variation is often over a class of data structures, or a class of programs -- e.g., problem solving methods.

Actually, the close fit of the individual program to a single segment of behavior looks more like a data point than a theory. That is, each program is a completely particularized version of a more general theory not yet formulated precisely. Hopefully, developing a large number of examples may

lead to inducing more clearly the common structure and the form of individual differences. The particularized program may be the appropriate description of the raw behavior, from which properties of revelance can be extracted. One cannot read significant features of the behavior without first describing it this way, any more than one can read signal out of noise without appropriate statistical processing.

The present study continues the accumulation of specific examples and the development of techniques for protocol analysis. It is devoted exclusively to the analysis of a single protocol. It follows up a technique used in a prior study of a chess protocol [21], in which an attempt was made to provide a rather exhaustive analysis prior to construction of a program. No attempt was made there to take the final step to the program. A similar approach will be taken here, making use of a different task.

We start with an analysis of the task, introducing the technical apparatus needed to describe the subject's behavior. Then we give a gross description of the protocol, followed by the detailed analysis.

Analysis of the Task

Crypt-arithmetic task. The following problem is presented to the subject

```
   DONALD   D = 5
+GERALD
--------
   ROBERT
```

In the above expression, each letter represents a digit; i.e., 0, 1, ... 9. For example, you know that D is 5. Each letter is a distinct digit. For example, no other letter than D may equal 5. What digits should be assigned to the letters such that, when the letters are replaced

by their corresponding digits, the above sum is satisfied.

This form of puzzle has been christened "crypt-arithmetic" by Maxey Brooke, who has collected a large number of examples [4]. Apparently, the only prior use of the task in psychology has been Bartlett's [2], which stimulated our use of the task.

Problem Spaces. To analyse the behavior of the subject we introduce the notion of a problem space. This consists of a set of positions (or nodes) each of which represents a state of knowledge about the problem. There is also a set of operators that apply to states of knowledge to produce new states of knowledge. A problem is posed in this space by giving an initial state of knowledge and requiring that a path be found to a final state of knowledge that includes the answer to the problem.*

Many problem spaces can be defined for a single problem. Each is to be defined by giving the class of expressions that can represent the states of knowledge, and then defining the set of operators in terms of these expressions. To do this conveniently we will make use of Backus Normal Form terminology [1]. This permits us to construct schemes for expressions, and assign suitably restricted domains for them. For instance, consider:

e := 1←d

1 := A|B|D|E|G|L|N|O|R|T

d := 0|1|2|3|4|5|6|7|8|9

---

* These notions are essentially those introduced in constructing heuristic programs, sometimes going under the name of the maze-model of problem solving [17], sometimes under the name of heuristic search [16]. Our use of the term problem space is consistent with these; our use of "state of knowledge" for the node expresses a preferred interpretation.

The lower case underlined alphabetic symbols represent classes, jd is the class of digits; 1^ is the class of all letters in the problem. The vertical bar, |, is a metasymbol, used to separate alternative expressions, or classes of expressions, for a class. Likewise, the colon-equal, :=, is used to separate the class name from its definitions. £ is the class of all expressions where *1* is replaced by a member of **1^** (i.e., a letter); *6* is replaced by a digit; and the assignment arrow, remains. Examples of je are: **A<-6, D<-5,** etc. Non-members of *e* are **U-6, 5<-D, G<-10, H<HL, D=5.**

With these definitions in hand we can define a simple problem space for the DONALD + GERALD task. First, we define £, the set of knowledge states:

s, := ej

We use the metacharacter, ^, to indicate the null expression. The last component of the definition is recursive. Thus it includes £ and £,£ and

and so on; that is, £ consists of lists of assignments. The interpretation is clear: an es expression is the association of a digit to a letter; an js expression is the conjunction of its terms. Therefore, duplicates from £ iare strictly redundant.

Next we define ^, the set of operators:

{}> := Make *e*

Thus, an operator makes an assignment, adding it to the state of knowledge. If s were the current state and Q were the operator A<-2, then

« Q(s) = s,A<-2

We will call the problem space just defined the legal problem space. It is about the simplest one in which the total problem can be defined. Simplic ity refers here to the ease with which such a problem space can be constructed

on the basis of the instructions given about the problem. Let us state the instructions for the problem in this space:

legal problem space:

$s_o := \phi$

s* := s such that

x in l implies x in s exactly once

x in d implies x in s exactly once

D←5 in s

DONALD
+GERALD
ROBERT

Our purpose is not to provide a complete formalization of the specification of the task. In particular s* is not really defined in the space at all. That is, a description is given so that a specific state, s, can be recognized as an s*. It is possible to consider definitions of a problem space in which the symbolic expression defining s* is an admissible expression in the space. They would be linguistically much more elaborate than the legal problem space we have just defined.

Problem Behavior Graphs. Given the operators and the starting node, one can lay out trees of search that might either solve the problem or represent the subject's search (or both). Figure 1, for example, shows the search tree formed by a problem solving system proceeding under the following rules:

1. Search according to the "depth first" strategy; that is, when at a position s, select an operator Q, find Q(s) = s', determine if s' is a solution;

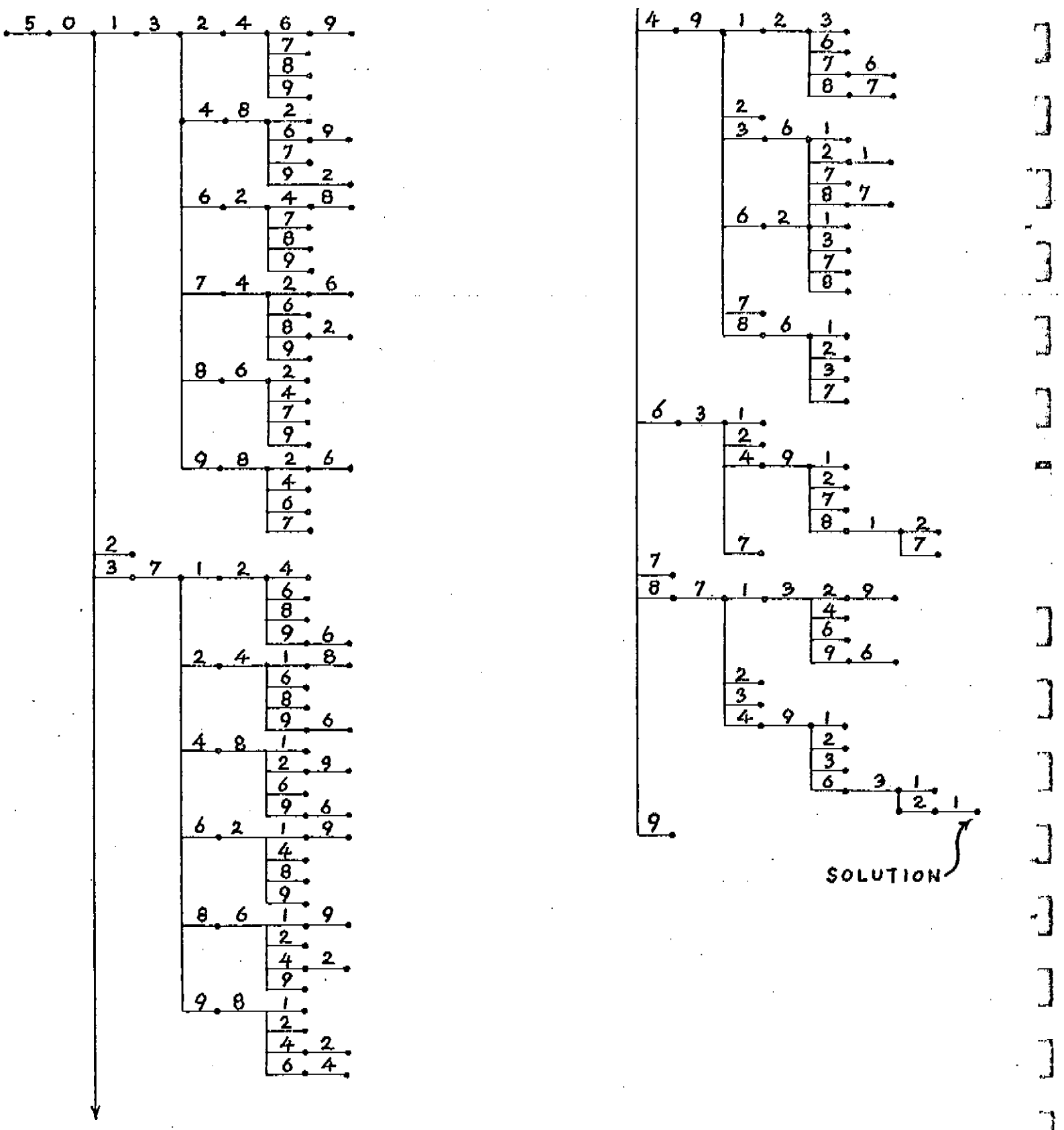


Figure 1: Depth first search.

if so terminate (success). If not, search s' according to the "depth-first" strategy (that is, recurse). If the search terminates from s' with success, then terminate with success. If not, then select the next operator at s, and repeat. If there are no more operators at s, then terminate with failure.

2. Generate operators, l←d, by generating the columns from right to left, and l within a column from top to bottom. Generate d in the order 0 through 9.

The test for terminating a branch is the construction of a state that cannot possibly lead to the solution: one having two different digits assigned to the same letter, having the same digit assigned to two letters, or having a false sum. These last steps that reveal the contradiction are not shown in the figure. If no checking for failure had been done until the end, then a tree with 10! terminals would have been generated (or 9! if D←5 taken into account).

The search tree is a way of displaying behavior that permits inferences back to the program used by the subject. It is an empirical question whether the subject will generate a tree in the course of solution, or will use some quite different solution technique. Likewise, it needs to be shown that a significant amount of the tree can be inferred from the protocol. But prior work and the remainder of this paper settles these doubts.

The example of Figure 1 is overly simple in one important respect: no part of the tree is searched more than once. Thus the time sequence of generation can be inferred from the total tree generated. In general this will not

be the case; the subject (or program) will wander over the same ground repeatedly. Thus we will introduce a modification, which we will call the problem behavior graph (PBG), which will retain the full information about the dynamics of search. The rules for PBG's are:

Rules for Problem Behavior Graph (PBG):

A state of knowledge is represented by a node.

The application of an operator to a state of knowledge is represented by a horizontal arrow to the right; the result is the node at the head of the arrow.

A return to the same state of knowledge as node X is represented by another node below X and connected to it by a vertical line.

A repeated application of the same operator to the same state of knowledge is indicated by doubling the horizontal line.

Time runs to the right and down; thus, the graph is linearly ordered by time of generation.

These rules are illustrated in Figure 2. The subject starts in node 1 in the upper left-hand corner. The first operator applied is Q1, leading the state of knowledge indicated by node 2. Then Q2 is applied, leading to node 3. At this point the subject returns to the same state of knowledge as in node 2; this is shown in node 4. The act of returning was not done via an operation in the problem space -- that is, by one of the Q's -- but by some other operation, such as recalling the prior state, or abandoning the information produced by Q2. Thus, the move from node 3 to node 4 does not show as an operation. At this point, Q1 is applied to node 5 and then a return is made to the state of knowledge represented by node 1. Node 6 must go on the line below nodes 4 and 5, since it occurred later in time. Q1 was again applied, as indicated by the

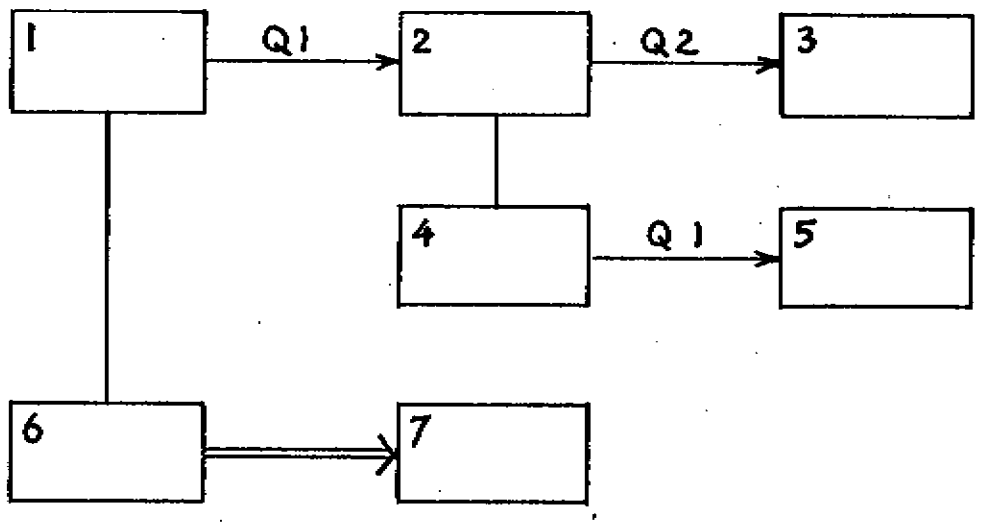


Figure 2: Basic definition of Problem Behavior Graph (PBG)

double line emanating from node 6. The connection between node 7 and nodes 2 and 4 is not indicated in the graph.

The problem solver is viewed as always being located at some node in the PBG, searching for a solution by generating yet other nodes. Yet the act of search itself generates information in addition to that represented by the state of knowledge at the current node. One variety is path information: the subject knows something about how he got to the node. Note that there may be many paths to the same node (as 3+5 = 8, 4+4 = 8, 2*4 = 8, 24/3 = 8, etc.), so that path information is not necessarily derivable from the state of knowledge. In our earlier example, given the state it is possible to derive which operators were applied; but it is not possible to determine either their order or whether they were applied more than once. Another variety of additional information is about past attempts. At node 4 (Figure 2) the problem solver must know that he has been in this state before and that Q2 was applied. Otherwise, he should apply Q2, rather than Q1. It is of course possible to be back in the same situation without knowing the full history of past attempts; this is quite possible at node 7, for instance. Both these varieties of information are functions of a node, so that we should think of the state of knowledge in each node being expanded to include them.

Initially two nodes are given directly to the problem solver, s_o and s* (often, as we noted, only a test for s* is given). During the course of search additional nodes may be retained in such a way that they are available to the problem solver "directly" -- that is, without regeneration within the problem space. Some of these may be tied to the path leading to the current node, but others may become independent of the current position. The stock,

as we will call it, constitutes the pool of states that is available for the continuation of problem solving if the current node is abandoned (as it must be eventually on every dead branch). It plays an important role in problem solving.

Additional information of a permanent nature (available independent of the current position) may be extracted from the states of knowledge as the problem solver searches through them. No special representation of such blackboard information* is included in the PBG, despite its importance.

Besides these varieties of knowledge, there is associated with each node several processes:

Evaluation:

Does s=s*?
Does s have information that should be saved?
Should s be added to the stock?
Should search continue from s?
Should the problem space be abandoned?

Select next operator to be applied

Is it desirable?
Will it work?
Has it been used before?

Apply operator to produce new state of knowledge

(May not always be successful)

Select new node from stock

(Given decision not to continue)

All these processes may be highly complex and extended in time. They may involve further search in problem spaces of their own, and extensive variation

---

* "Blackboard" refers to the analogy of many people putting information on a common blackboard where it is available for all to use, independent of its order of recording and source.

in the state of knowledge about facets of the total problem not represented by this problem space. Likewise, several of the processes may be absent in special cases, or independent of the current node and so better viewed as a common process -- e.g., the selection of a new node. We will still write a node at the head of every arrow, even though no new state of knowledge was produced; thus, a node may be vacuous. This convention is convenient, since the inference that a result is not actually produced may not occur until late in the data analysis.

Varieties of problem solving. We are concerned in this section with how problem solving proceeds in a given problem space. The next section will discuss multiple problem spaces for the same problem.

The basic structure of problem spaces dictates that problem solving takes the form of search. It also determines where the opportunities (and necessities) for further choice occur, and hence where intelligence can enter. The three items just given -- evaluation, operator selection, and node selection -- summarize these entry points. However, they cannot be dealt with independently, since problem solving appears to proceed according to various strategies or methods that dictate coordination between all three categories. Thus, we will organize our comments around generalized methods.

Forward search. The simplest way of searching involves generating the operators in some fixed order (according to their own structure, such as trying the digits in numerical order), and testing for whether a solution (or progress) is obtained. The search given in Figure 1 was so generated. We will term any method that uses an operator generator that is basically independent of the position, a forward search method.

Even with a fixed generator there are still various strategies of search available. One extreme is the depth-first strategy, already illustrated, in which a position once generated and accepted immediately becomes the starting point for further search, and all search that will occur from that position is completed before returning to any positions prior to it. Another extreme is the breadth-first strategy, in which all positions at a given depth from the initial position are generated before going on to any that are deeper. Both these strategies have substantial, though differing, memory requirements for the stock of positions. The depth-first strategy demands perfect retention of all the positions in the path leading to the current position; the breadth-first strategy demands perfect retention of all positions at a given depth (whose number generally will increase exponentially with increasing depth). Other strategies can be constructed that have less stringent memory requirements; these involve repeated searches over the same ground. An example is the progressive-deepening strategy discussed in the analysis on chess already mentioned [21]. One version of this requires only that the initial position be retained, since the search always returns to the initial position. However, some other blackboard information must be retained to gradually move the search through the space.

Means-ends analysis. In the definition of a problem a desired object, s*, is given (or determined). In the forward search method a process must exist that tests if s is either s* or constitutes progress toward s*. If, in addition, s* is used in the selection (or ordering) of the operators to be applied at s, then we term the method, means-ends analysis. That is, the means (i.e., the operators) are chosen with a view towards the end (i.e., s*).

Both forward search and means-ends analysis provide direction to thinking. But, to use some analogies, the direction shown by forward search is like that of water flowing down a hill: it would spread in all directions, and it is the shape of the environment that makes it appear to seek the bottom of the hill (the tests of progress). The direction of means-ends analysis is like that of the servo: a comparison is continually made with the final end, and action taken to diminish the difference.

In general means-ends analysis requires the detection of a difference between the current state of knowledge and the desired one, $d(s,s^*)$, and this difference is used to make the selection, $Q = f(d)$. Of course, only selection functions leading to operators that reduce or eliminate differences would be of much use. Only in limiting cases will the selection be dependent only on s^*. These mechanisms are well illustrated in GPS [19], where the difference is obtained by a match process and the selection function (f, above) is given explicitly as a table of connections.

In terms of the PBG there is no need to represent the final state in the graph, since it is common information available at all nodes. Consequently, the existence of means-ends analysis will be revealed only by the kind of lawful behavior shown in the selection of operators.

Goals and goal hierarchies. As we have just noted, in case there is a single goal there is no need to represent it in the PBG. However, if the goal changes from time to time, then it is important to know what goal (or goals) is being attempted from a given state of knowledge. We could view this information as being part of the state of knowledge. This would require, however, that we augment the set of operators with some that set and abandon goals,

for we have identified the states of knowledge as constructable on the basis of the operators: if we are at s and apply Q, then the new state of knowledge is s' = Q(s). If there were to be an unrecorded change of goals, this would no longer be the case.

What sorts of goals are possible within the framework of a problem space? First, one can seek states of knowledge as yet unattained. That is, one may say "get s." This implies that s is obtained through the application of operators to states of knowledge that already exist. It need not be the case that the desired s is fully known; the s* in our legal problem space is only specified as a member of a set of s. Exactly what sets are possible of expression depends on the particular problem space. It will have to be specified as clearly as we have specified operators, in order to define the goal setting operators. It is also possible to "check s"; that is, to verify some knowledge already obtained.

Besides seeking new states of knowledge, it is possible to seek to apply operators. A given operator need not be applicable to an arbitrarily given state of knowledge. One can then have a goal to "apply Q to s." Such a goal can be attained only if something changes, since, by hypothesis, Q cannot be applied to s. One can either seek an s', such that Q can be applied to s'; or seek a Q' such that Q' does apply to s (one might fiddle with both, of course). In fact the former appears much the more likely, since the problem space is already devoted to obtaining new s's. To find new Q's in any way besides relatively fixed operator selection processes, requires a problem space that has operators for states of knowledge and metaoperators for searching through these.

These ideas lead to PBG's like that shown in Figure 3. At s1 the act of setting a goal is selected, so that at s2 the search is on for s*. Operator Q1 is applied, yielding a new state, s3. From s3 operator Q2 is attempted but fails, and the goal of applying Q2 is selected instead. Thus at s4 the search is on for how to do Q2. Q3 is applied, leading to s5, which is apparently the end of the search for the acting goal, since Q2 is applied, leading to s6. Next, Q4 is applied

Figure 3 is ambiguous about what goal is being sought at s6. One assumption, which defines some of the path information kept by the problem solver, is the following:

> Goal stack assumption: The goals occurring in the path to the current state of knowledge are stacked in the order of their occurrence (most recent on top). The top goal in the stack is the current goal. When it is attained or abandoned the stack is popped, and the next goal down becomes the new current goal.

It is an empirical matter whether this goal stack assumption is justified for a particular problem solver. If it is (and it will be true to a first approximation for the protocol of this paper), then Figure 3 becomes an adequate representation of the search; and Q4 is being applied in the search for s*.

We have discussed the representation and mechanics of goal formation, but have not dealt with the various rules for when to change goals and what new goals to introduce. For instance, one rule used by GPS is always to set up the subgoal of applying an operator if there is difficulty in so doing. In fact, the simplest versions of GPS can be completely described as a means-ends analysis problem solver obeying the goal stack assumption and always

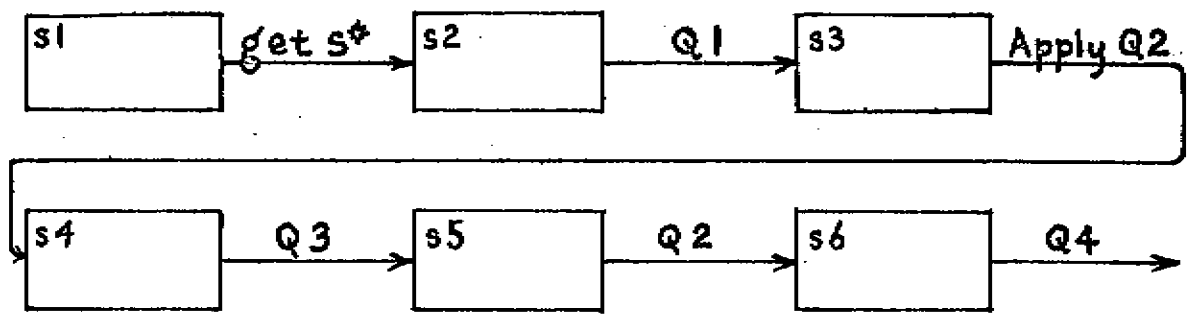


Figure 3: Example of goals as operators.

setting up apply operator subgoals.*

Desired states, as we have already seen, may be expressed as a set of possible states to be attained. Desired operators may also be only partially specified. For example, "Adding some odd digit" is a possible operator in the DONALD+GERALD task. In both cases it is possible that further actions could be taken to specify the goal. Thus Figure 4 shows a series of specifying actions, both of the desired state and of the operator to attain it. In terms of problem spaces we could say that there exists a problem space whose states of knowledge are the various expressable subclasses of operators. The operators of this space are acts of specification that produce ever smaller subclasses. Likewise, we could talk of a similar problem space for the specification of desired states. In some cases we may feel that such problem spaces are necessary to describe the subject's problem solving. More often, as will be the case in the analysis of this paper, the specification acts can be adequately characterized as fixed processes of the same kind as operator selection and state evaluation processes.

Several problem spaces. The foregoing comments have tried to make clear some of the different ways problem solving might proceed in a given space and how it might be reflected in the PBG. Even there, as we enriched the kinds of things talked about -- to desired states and desired operators -- we had to decide whether to introduce several spaces or stay with one. More generally, problem solving can proceed with the aid of several related spaces, as long

---

* This is the information expressed in the diagram showing the three main methods [see, for example 19, Figure 3].

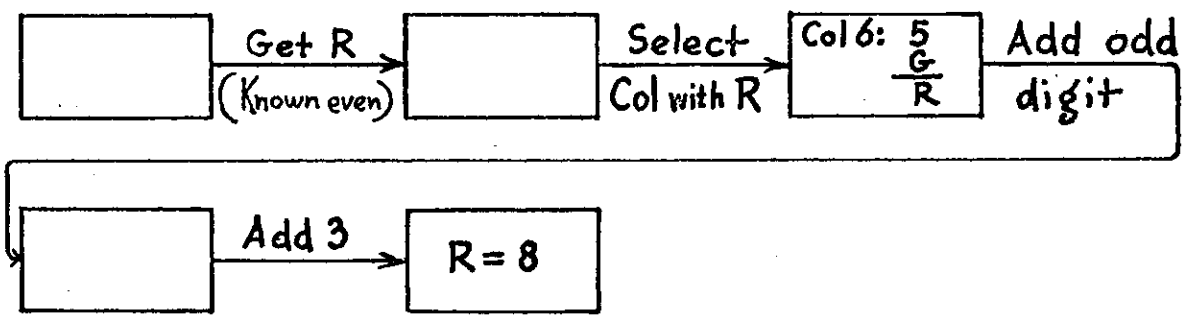


Figure 4: Example of gradual specification of operators.

as operations exist for passing information back and forth between them. Of course, given two problem spaces ($\underline{S}$, $\underline{Q}$) and ($\underline{S}'$, $\underline{Q}'$) it is always possible to consider that there is only a single space, whose states are the pair ($\underline{S}$, $\underline{S}'$) and whose operators are the union $\underline{Q}u\underline{Q}'$; where it is understood that an operator only acts on its own subspace. Factorization into two spaces makes sense where interaction between them -- expressed by the number of cross-over links -- is low enough for the spaces to have separate identities. Let us consider some examples.

Working forward and working backward. Given a problem space, ($\underline{S}$, $\underline{Q}$), we can construct another, ($\underline{S}$, $\underline{Q}'$), that has the same states of knowledge, but uses the operators inverse to those in $\underline{Q}$. Working with $\underline{Q}$ is working forward; working with $\underline{Q}'$ is working backward. It is possible for a problem solver to wander freely from one of these spaces to the other -- closing the gap from either direction. Often, both for computer programs and for humans, solving occurs exclusively in one or the other. If a problem solver worked alternately in one for awhile and then the other for awhile, one would still want to keep the two spaces separate. Sometimes the difficulty in working backward stems from the additional generality required. Thus, in the legal problem space the desired state, s*, is given as a set of states. Consequently the inverse operators of $\underline{Q}$ cannot work on it. To work backwards would require creating operators that derive an expression for the states that lead to classes of states. Once this was done, one could apply these extended operators both forwards and backwards and have a more powerful problem solver. But none of the subjects we have observed created such operators in DONALD+GERALD.

External and internal responses. The amount of information obtained from subjects about an instance of problem solving can vary. At the one extreme we can obtain only the final solution, if found. Alternatively, we might provide some way for the subject to indicate something about where he is during the solution process. In the DONALD+GERALD task a natural way to do this is to permit (or require) the subject to write down the various digits associated with the letters as he determines them. We can view the subject as working in a problem space that has a written display of the problem with various letters replaced by digits. Figure 5 describes this a little more formally, where the display is indicated by rows (r) and columns (c).

We view the subject as doing considerable internal processing, so that the act of writing (including the decision of what to write) is not a simple operation, but is itself developed after a search in some internal problem space (or spaces). We use verbal behavior to tap into this internal space. Now, the verbal comment

"Well, N might be 7. If it were.."

is as much an overt response as the command "Will you write down N equals 7 on the board." However, rather than erect yet another problem space, we will view the verbal behavior as a direct sampling of the internal space.

The question at the moment is how to represent the action taking place in the two spaces. One view is given in Figure 6. The PBG in the internal space is represented as if it were the main one. At those nodes where a decision is made to apply an operator in the external space, a notation is made. In our figure this is shown by a vertical line down to the lower plane. The external PBG is much distorted, since the position of the operators is dictated by the internal PBG.

External Problem Space

```
r := r1|r2|r3                          rows (r dropped if clear)

c := C1|c2|c3|c4|c5|c6|c7              columns (c dropped if clear)

w := ljd                               characters in display

i := l|lcllcr                          instances of a letter

     7654321
      wwwwww  r1
X :=  wwwwww  r2
      wwwwwww r3

WO := d/i|l/i                          write operators
```

board

```
        DONALD                  50NAL5
XI:     GERALD        XI' :     GERAL5
        ROBERT                  ROBERT
```

5/D12 write 5 for D at c1,r2
9/E5 write 9 for E at c5
2/G write 2 for G
L/L21 rewrite L at c2,r1

Figure 5: External problem space definition.

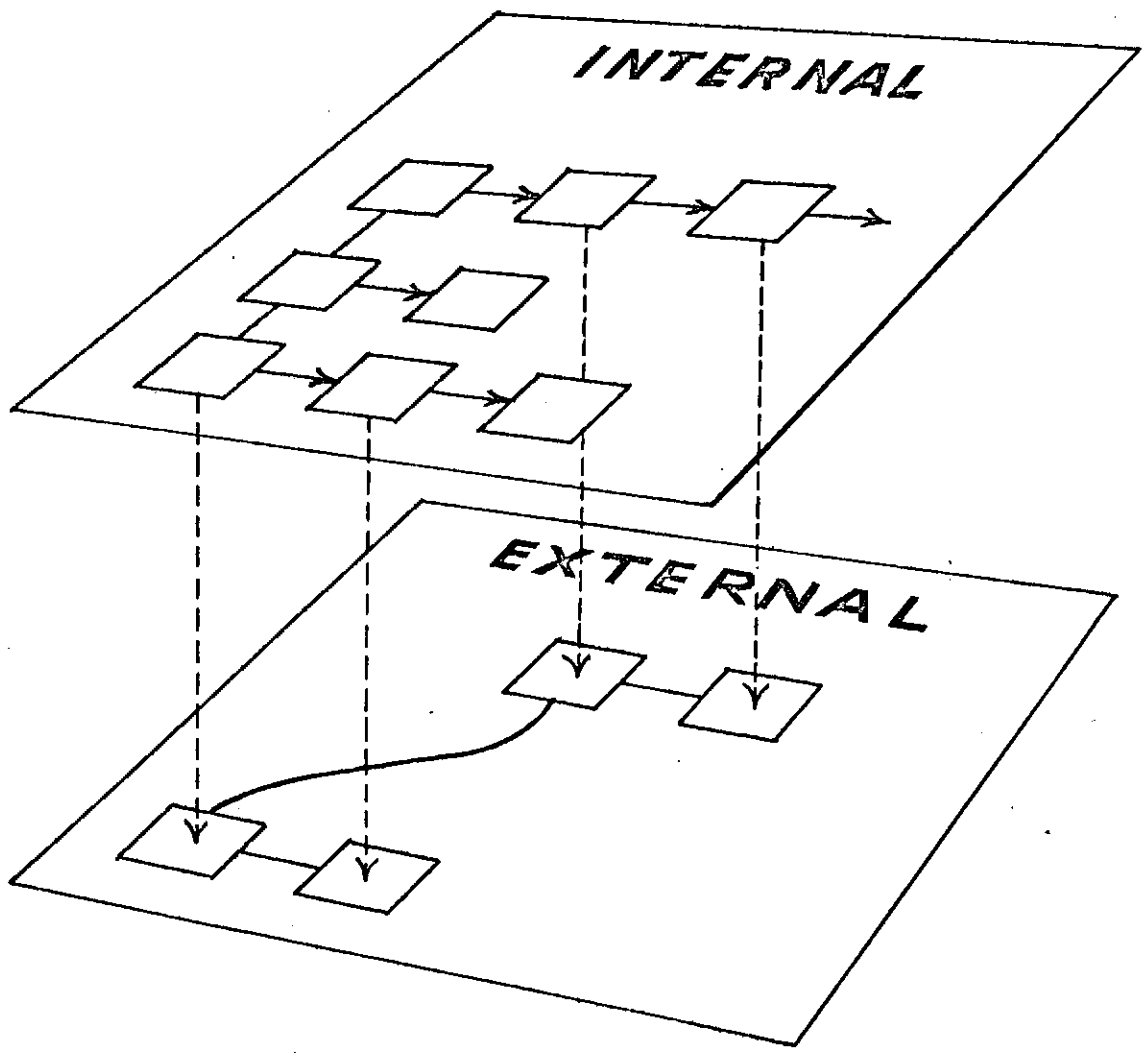


Figure 6: Internal-external PBG, type 1.

An alternative view is shown in Figure 7. This is the view from the external space. Each writing act is shown having an entire PBG in the internal space devoted to its determination. Such a subgraph might be devoted to selecting the operator, to testing its feasibility before "actually" applying it, etc.

Each of these two views can be correct under different assumptions about the processing. Figure 6 assumes that the act of writing produces no change in the state of knowledge. The external space is epiphenomenal, so to speak. However, it still might play a role with respect to the recording of information -- functioning as the blackboard or the stock of nodes to which the subject can return. Contrariwise, Figure 7 assumes that all the processing in the internal space is entirely subservient to the external act, so that after the external act is made, no other information is available to the subject than that shown by the now external state. A completely stimulus bound organism would be represented by Figure 7. If neither of these extremes were approximated, one could treat the two spaces as one, showing operators of both kinds intermixed in one large PBG.

Individual operators and class operators. Let us start with the legal problem space (s, Q) for DONALD+GERALD. If we take the kinds of information that make up its states of knowledge -- the assignments, l←d -- as primitive, then we can construct a new problem space whose states of knowledge are disjunctions of these primitive individuals. That is, it may be known only that R is 1, 3, 5, 7 or 9 or that E is 0 or 9. For these states of knowledge to be operational there must exist operators that take such states of knowledge and

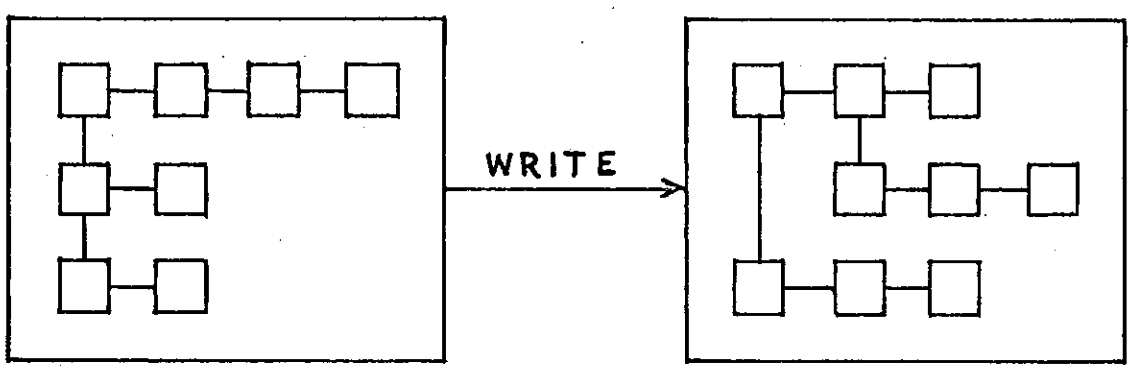


Figure 7: Internal-external PBG, type 2.

produce new ones. For example, knowing that two letters were assigned odd digits, we could conclude that their sum is even, without ever considering the exact digits that were assigned. Thus, class information need not be given as an explicit enumeration, but according to some system of properties (in this case even-oddness).

Much additional power comes through the ability to work with classes rather than individuals. Basically, it provides the ability to do in one search what would take a vast examination of cases if the members of the set were enumerated. (When dealing with infinite sets, of course, such enumeration is ruled out in principle.) Thus in the even-odd example, the same conclusion about the sum can be reached by considering 5x5 = 25 cases (since we are working only with the ten digits).

It should be clear that there are not just two spaces, the primitive one admitting only elementary facts, and the one admitting all degrees of classification. Rather, there is a series of increasingly powerful spaces depending on what operators exist for producing new states of knowledge. In general one would expect a problem solver to operate in the most powerful space for which he has the operators. However if the more powerful operators are too costly in some way (such as having to take time to write on paper), then the subject might alternate between a less powerful space (in his head) and a more powerful one (on paper).

Planning. To drive across the country one can first consult a map and then, having planned the trip, take to the car and actually drive it. It is clear that problem solving in two spaces is being carried out: in the space of the map and in the space of the actual country. Figure 8 shows the relationship.

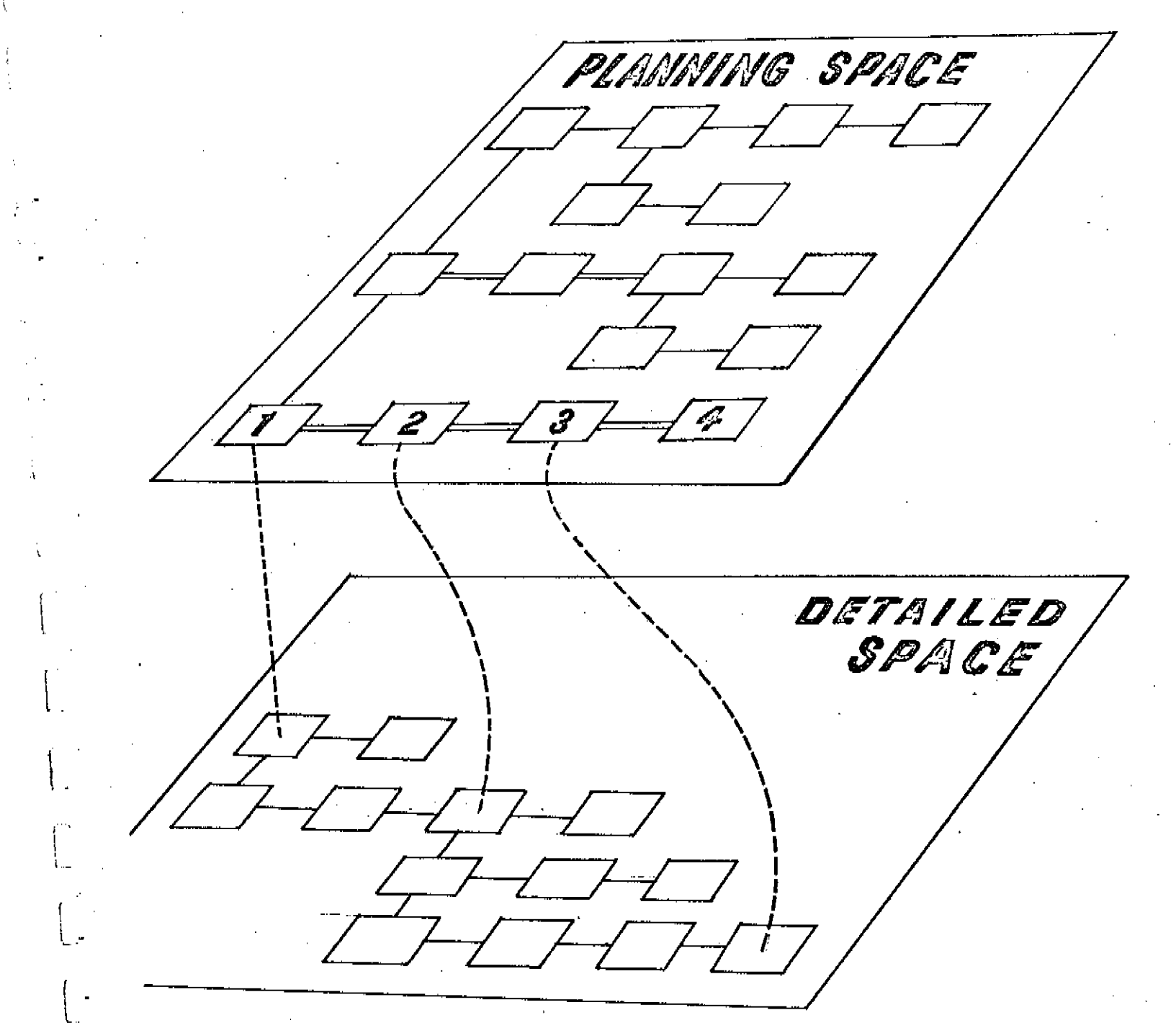


Figure 8: PBG for planning.

The top plan is the map, the bottom the country. Problem solving first takes place in the map. When a solution ("the plan") is found there, as indicated by the first sequence of nodes, labeled (1) through (4), problem solving ("implementation") can start in the more detailed space. Each of the nodes in the plan sets up a goal in the lower space, which requires some search to attain it.

The map-country example involves two spaces that are quite distinct in their characteristics. However, planning and implementation often can occur in the same space. The requirement is a rich enough set of operators so that a solution can be found that ignores or assumes some kinds of information. Then this solution path can become the plan for working out the details. Actually, in the DONALD+GERALD protocol that we will examine no extensive planning occurs. However, a conceivable type of planning is one that endeavors to determine what order to consider the letters, as in the following line of reasoning:

> Since D is determined, then T is determined. Thus, the carry into column 2 is known. Hence something is known about R. But then column 6 should tell us something about G, D being known and R being partly known. But the carry into column 6 isn't known, so the value of O and maybe E had better be found first. ..."

The operators here are akin to the counting of equations and unknowns. The result is a good order for the generation of operations, as in Figure 1. (It is doubtful that a plan at this level could discover the optimal order.)

First Phase Analysis of the Protocol

Overview of the protocol. The previous section has given us enough tools to start the protocol analysis. The protocol is reproduced in full at the end of the paper. It was taken in the Spring of 1960; little is known about the subject except that he was a male college student at Carnegie Institute of Technology. No separate record is available on what was written down, although it is usually clear from the protocol when something is written, as in B9. Likewise, no separate timing information is available. However, there are a total of 2186 words. Since other protocols under similar conditions average about two words a second, we estimate the length of this one at about 20 minutes. Timing information is not critical, since we have no way of making use of it.

The protocol has been broken up into short phrases, labeled B1, B2, ..., B321. This includes the remarks of both the subject and the experimenter. The phrasing is based on a naive assessment of what constitutes a single task assertion or reference. It is meant to ease reference and does not affect the analysis explicitly. However, the total number of phrases, 321, does give some indication of how many data are present -- one phrase every three to four seconds. Considering how much most of us think can go on in a few seconds this may seem a rather low density.

The most basic question about such data is to what extent the phrases have an unambiguous meaning. The verbalizations are free, and so we can expect -- and get -- exchanges such as B169 - B176 which includes "I still feel as though I'm baring my soul." In general, however, the task provides an extremely narrow context which makes interpretation relatively easy.

One might feel that the verbalizations should be encoded into some formal

categories, in order to make it clear what information was being used in the analysis. This was tried in the chess analysis already mentioned [21], without returns commensurate with the effort. There appear to be some objections in principle to such an encoding, especially where the content of the utterances is to be preserved. In any event, in practice those parts which are easy to code, don't need it -- e.g., B189: "I'm going to make R a 9" or B208: "that means that A + A has to equal 10." Those parts which are not, should be kept in their original form in order to extract any information they do contain -- e.g., B16 and B17: "that are each -- somewhere --" or B62 - B65: "Now if the -- Oh, I'm sorry I said something incorrect here, I'm making -- no, no, I didn't either." As a matter of fact, breaking the utterance up into small phrases goes a long ways towards isolating a series of unambiguous "measurements" of what information the subject had at particular times. These measurements carry the main burden of the analysis; the ambiguous ones that are left operate mostly as weaker checks of consistency.

We now turn to an analysis of S3's problem solving behavior. Various additional issues of interpretation will be dealt with as they come up in the context of the analysis.

The Problem Spaces of S3. An examination of the protocol shows the following characteristics, which must be encompassed in a formalization of the subject's problem spaces.

1. The subject writes some things down, but long periods go by with no writing. Thus, there are at least two problem spaces.

2. Several kinds of actions occur: the assigning of digits to letters;

inferring of relationships from the columns of the sum; and the generating of digits that satisfy certain relationships. For example:

B43. "if we assume that L is, say, 1." (assignment)

B44. "we'll have 1 + 1 that's 3 or R --" (inference)

B26. "So R can be 1, 3, not 5, 7 or 9." (generation)

3. The relations that occur are equality, inequality, and even-oddness. Equality must be kept distinct from assignment, since the subject appears to know throughout whether a digit has been inferred equal to a letter or assigned to that letter. Examples:

B134. "A would have to equal 5." (equality)

B58. "R has to be a number greater than 5." (inequality)

B22. "which will mean that R has to be an odd number." (parity)

B95. "Of course, this is all going on the assumption that R is 7 --" [after B61. 'Bo we'll start back here and make it a 7."] (assignment distinct from equality)

4. The subject is able to consider disjunctive sets. Example:

B74. "But now I know that G has to be either 1 or 2."

5. Although the subject frequently states equations, these all correspond to the reading of a column; there is no evidence of the algebraic manipulation of equations. (Such manipulations do show up clearly in protocols of other subjects.)

6. With respect to carries, the subject is able to use them in inferences, to infer them, to seek them, and (possibly) to assign them. For example:

B70. "because 3 + 3 is 6 + 1 is 7." (use)

B85. 'which would mean that I was then carrying 1 into the left hand column." (inference)

B261. "There's no place where I can get L + L to equal more than 10, so I could make -- (seek)

B221. "Suppose I would carry 2 from the column." (assign)

The last example shows that the values of the carry cannot be restricted to 0 and 1 for this subject.

7. There exist a few actions and kinds of information that lie outside the range indicated above. These occur so rarely that they must be handled in an ad hoc fashion in any event. Examples:

B40.	"Possibly the best way to get to this problem is to try different possible solutions."	(reference to a method)
B50.	"it's not possible that there could be another letter in front of this R is it? Is it or not?	(an operation to obtain information from experimenter)

We can now describe the problem space that appears to cover most of the protocol. It is an internal space, in which writing operations into an external space occur occasionally, in the manner of Figure 6. This internal space is shown in Figure 9. It is an expanded version of the legal problem space, although for completeness we have repeated things set forth earlier. Some of the distinctions made within it are dictated by considerations yet to come.

A few examples will serve to make the specification clear and to explain the few special symbols that occur in it.

Internal Problem Space

l	:= A|B|D|E|G|L|N|O|R|T	letters
d	:= 0|1|2|3|4|5|6|7|8|9	digits
d*	:= d|x|y	digits or variables for digits
t	:= t1|t2|t3|t4|t5|t6|t7	carries
v	:= l|t	variables
ls	:= l|l,ls	letter sets ls = all l fls = all d still free
ds	:= d|d,ds	digit sets ds = all d fds = all d still free
cs	:= c|c,cs	column sets cs = all c
rel	:= <|=|>|<	relations
pro	:= even|odd|free	properties
suf	:= !|?|-p	suffixes
ee	:= v|v rel d*|v rel ds|v prp	elementary expressions
e	:= ee suf	expressions
s	:= e|e,s	states
Q	:= PC(c)|GN(v)|AV(v)|TD(l,d)	operators
g	:= get v|get ee|get ls| check e|check cs	goals

Figure 9: Internal problem space definition.

0←x	the letter O is assigned a digit x (a specific one, but whose value is not yet known)
A=5-p	A=5 is not possible
G=3,4	G is 3 or 4
t6?	The value of t6 (the carry into column 6) is unknown.
get B	Obtain the value of B
get t3=1	Obtain the value of t3 to be 1
N free	N is any digit; i.e., there is no additional constraints to the selection of a value of N.

Four operators are given, which are used to generate new states of information from old. Initially, we will describe these in gross terms by the kind of information they put out and the kind of information they use as input. This will be enough to identify their occurrence in the protocol. We will discuss later what consistent algorithms can be fashioned that describe the occurrences of the operators.

PC(c) Process column c. The input to PC is all the information in the state about the three letters and two carries associated with a column. The output is an expression, e, about some of the variables (l and t) of the column. The specification of what variable to get information about may or may not be determined prior to performing PC; if so we can write PC(c,v).

Examples:	t_i	r1	+ r2	= r3	t_{i+1}	output
c1	0	D=5	D=5	T	?	T=0, t2=1
c2	1	L	L	R	?	R odd
c6	?	D=5	G	R←7	0	G=1,2; t6?

GN(v) Generate v. The input to GN is a variable with whatever information is known about it. The output is the set of admissible values, not taking into account whether or not the values are assigned to other letters.

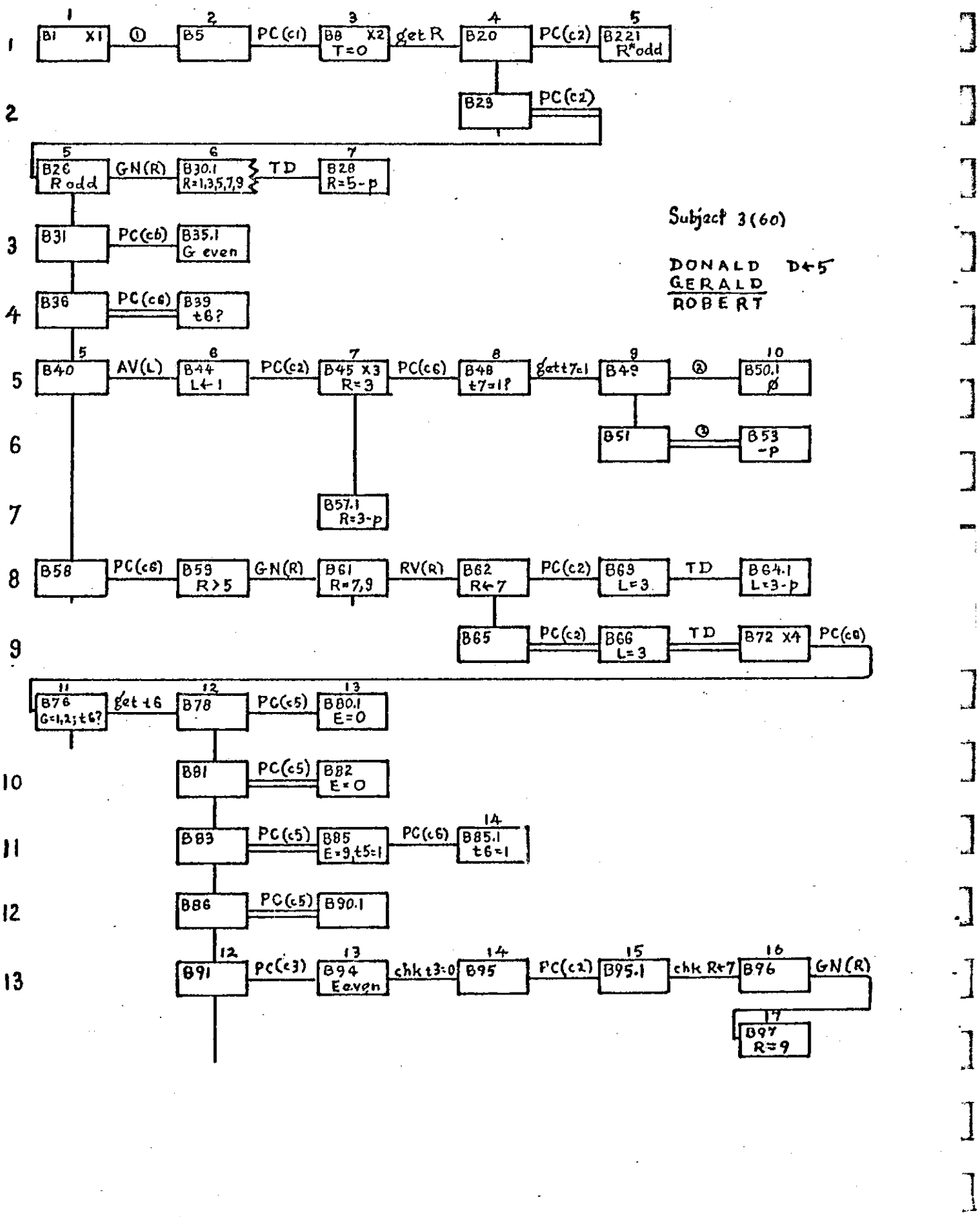
Examples	output
L (nothing known)	0,1,2, ...
R odd	1,3,5,7,9
R odd, R>5	7,9

AV(v) Assign value to v. The input to AV is a variable with whatever information is known about it. The output is an assignment of a digit to that variable. If the digit to be assigned is determined prior to performing AV, we can write AV(v,d).

Examples	output
AV(L)	L←1
AV(R)	R←9

TD(l,d) Test if d admissible for l. The input to TD is a letter with whatever is known about it, and a digit. The output is a statement either that the digit is admissible or that is not. TD takes into account 1) whether d is used for another l, or 2) whether d is outside the known restrictions for the given l.

The Problem Behavior Graph of S3. Figure 10 gives the behavior graph for the subject. The conventions are those already laid out. No operations into the external problem space are shown; instead at those points where something was written we have put an Xi in the upper right hand corner of the box, giving the name of the new external display. These are shown on the last page of the figure. Figure 10 has had to be folded rather badly to get the tree onto page-sized sheets. Figure 11 gives the outline of the total tree. It also



Subject 3(60)

DONALD D←5
GERALD
ROBERT

Figure 10: Crypt-arithmetic: Initial segment of Problem Behavior Graph.

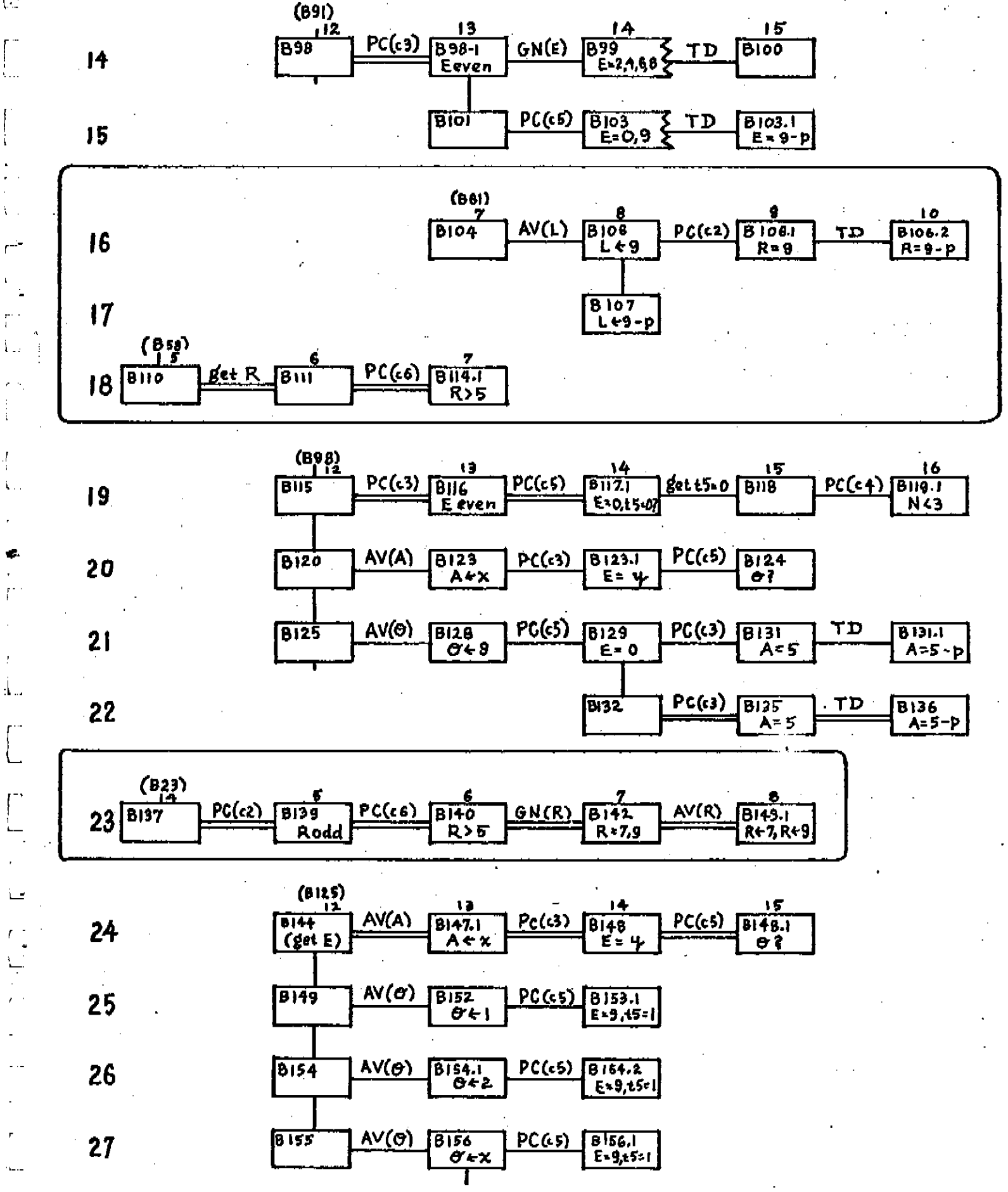


Figure 10 (continued)

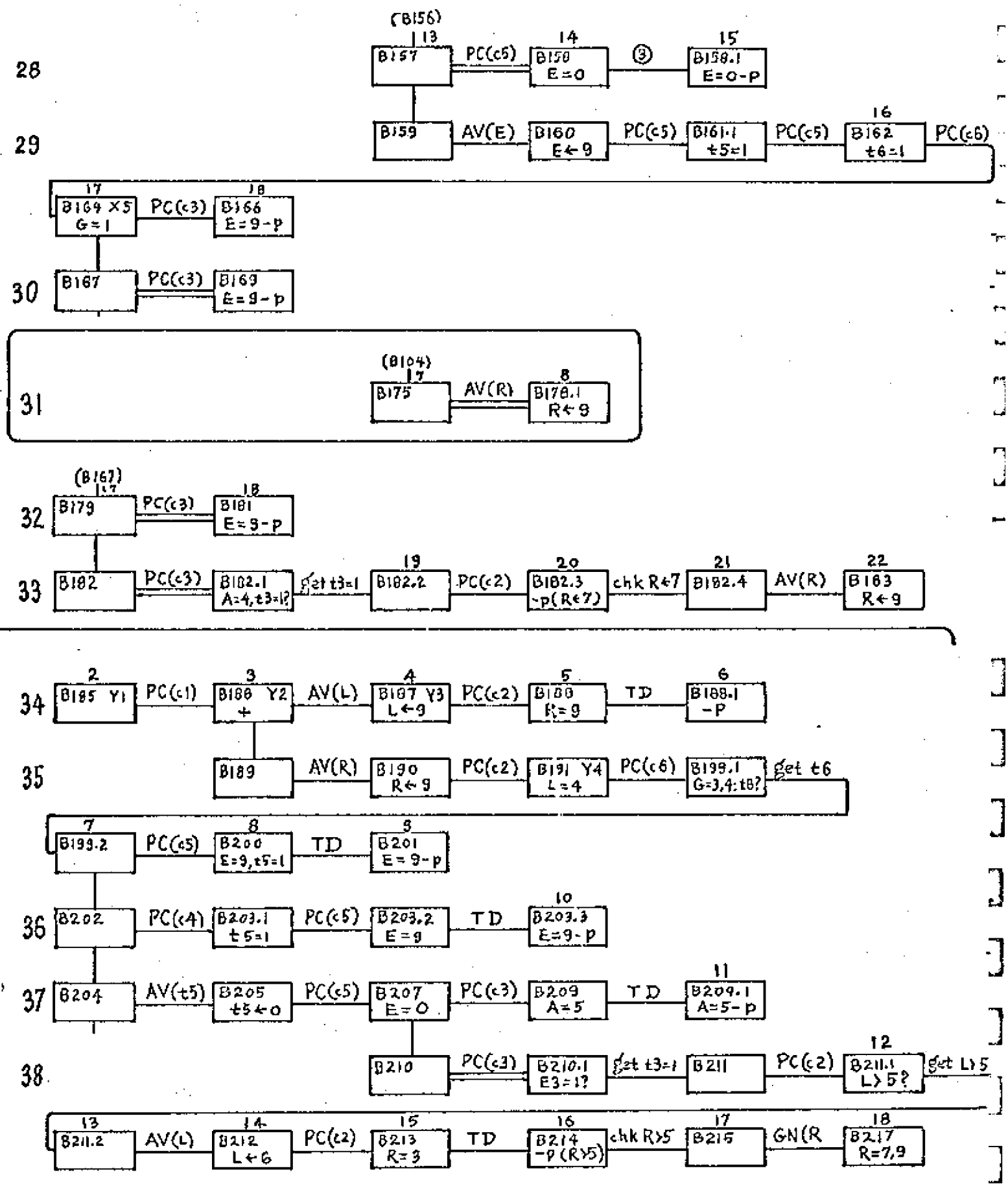


Figure 10 (continued)

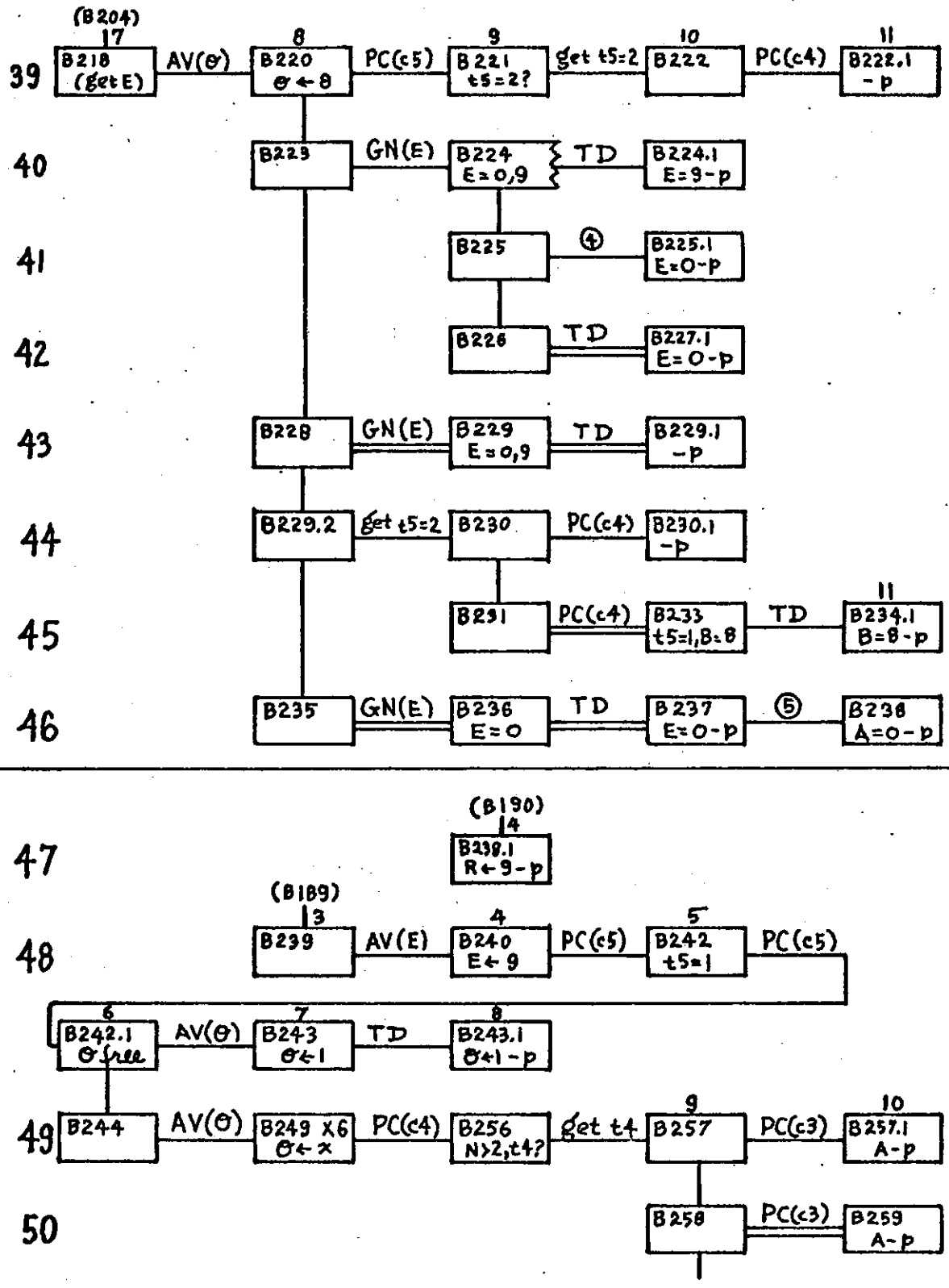


Figure 10 (continued)

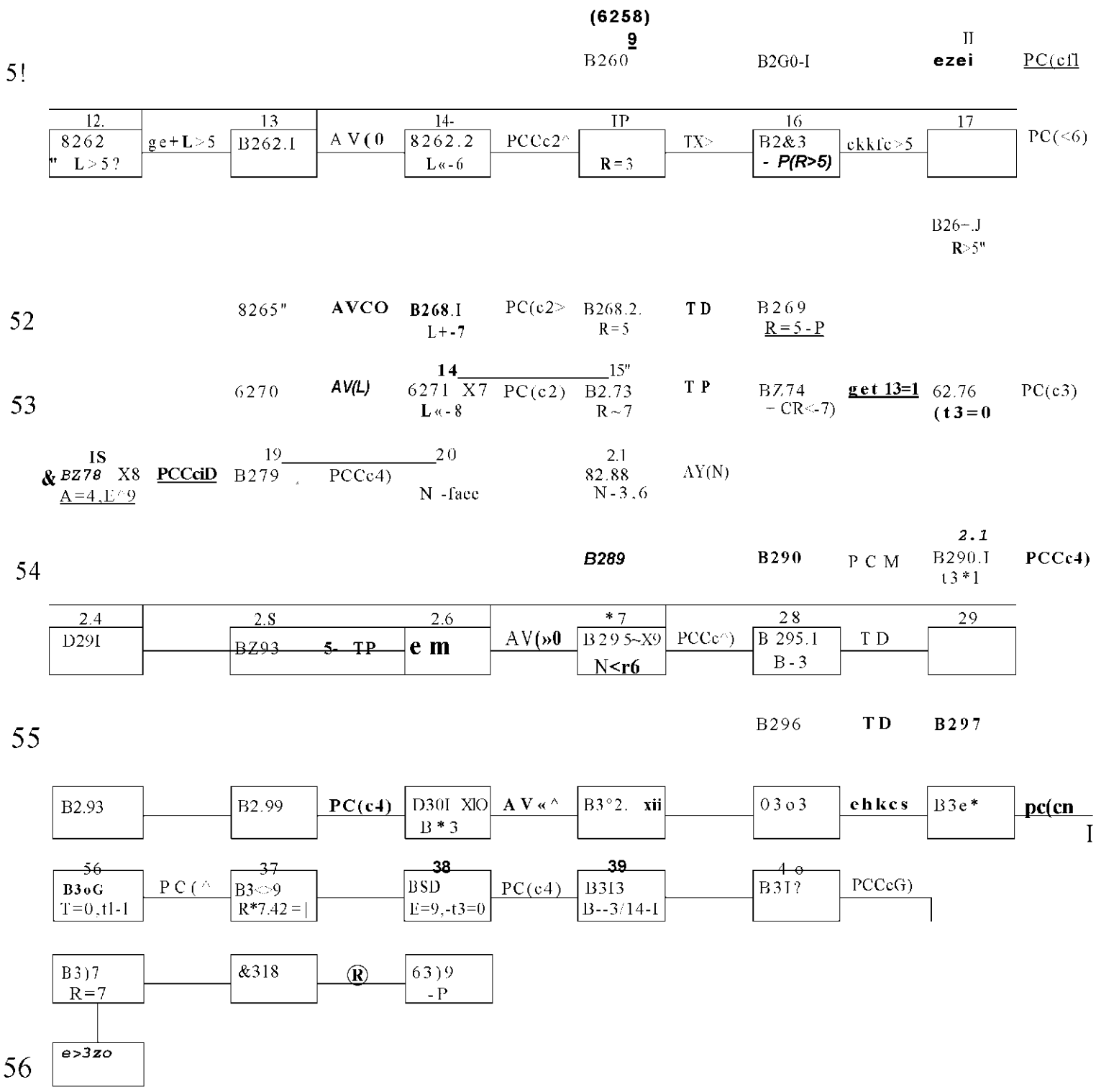


Figure 10 (continued)

B1	X1	DONALD GERALD ROBERT	B295	X9	5x6485 197485 7xB970
B8	X2	5ONAL5 GERAL5 ROBER0	B301	X10	5x6485 197485 7x3970
B45	X3	5ONA15 GE3A15 3OBE30	B302	X11	526485 197485 723970
B72	X4	5ONA35 GE7A35 7OBE70			
B164	X5	5ONA35 1E7A35 7OBE70			
B185	Y1	DONALD GERALD ROBERT			
B186	Y2	DONAL5 GERAL5 ROBER0			
B187	Y3	DONA95 GERA95 ROBER0			
B191	Y4	5ONA45 GE9A45 9OBE90			
B249	X6	5xNA35 197A35 7xB970			
B271	X7	5xNA85 197A85 7xB970			
B278	X8	5xN485 197485 7xB970			

Notes for numbered operators

(1) Interaction with experimenter to define problem.

(2) Asking experimenter about t7 -- i.e., on definition of the problem.

(3) Recall of information from B136; we have no operator for this.

(4) Experimenter interjects E=0-p(T=0!).

(5) Digit based operation (given 0-p find all 1 affected)?

(6) Given a solution, to evoke checking behavior.

(7) Given a solution, to evoke finding all solutions.

(8) Shifting assignments, but how?

Figure 10 (continued)

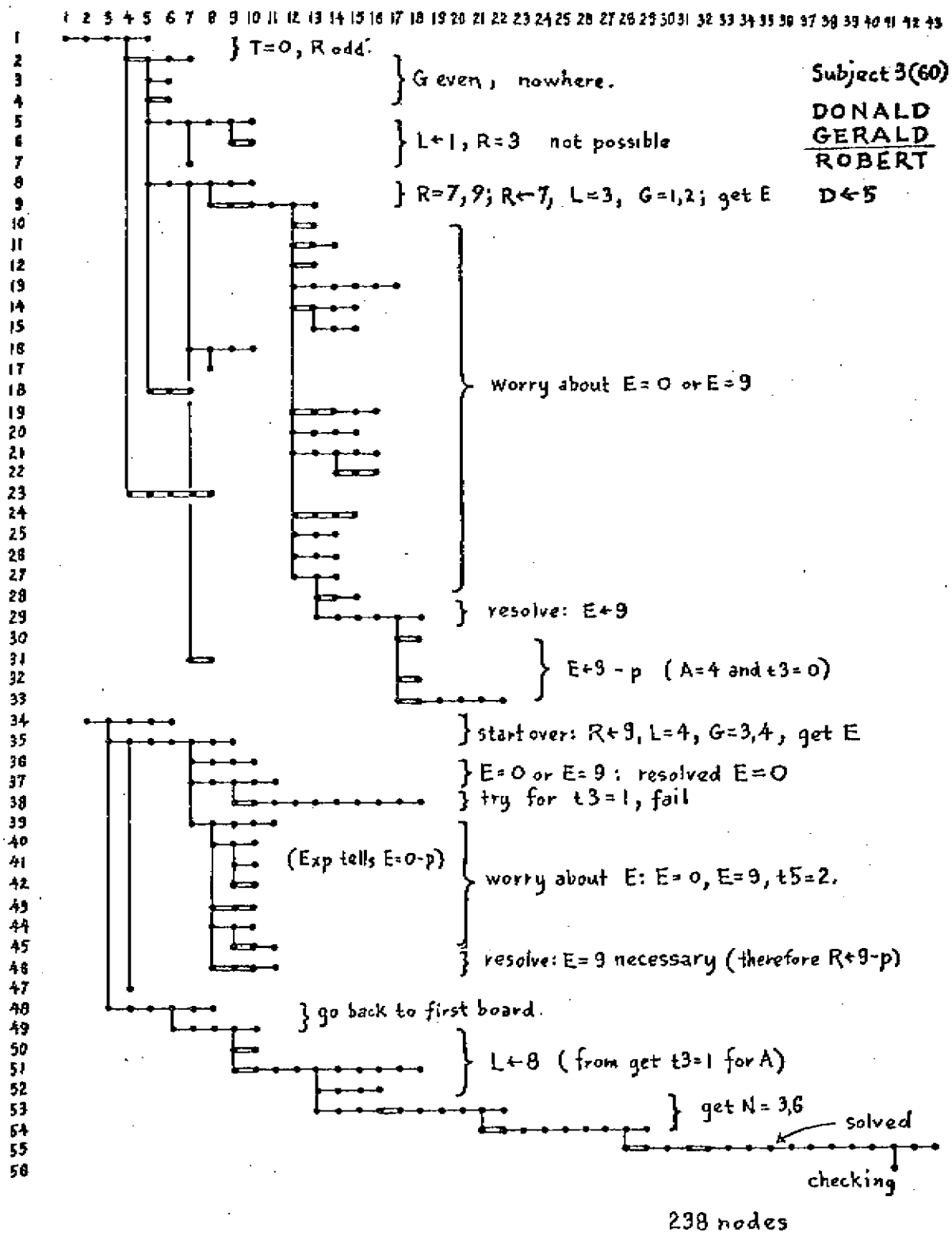


Figure 11: Crypt-arithmetic: Total Problem Behavior Graph.

provides a summary of the main problem solving episodes, a matter that we shall not take up until all the other analysis is complete.

Each of the nodes corresponds to an interval in the protocol. The B-numbers in the upper left give the approximate starting point; and lines have been drawn across the protocol at these points. Where the divisions were not fine enough, we have simply used decimal numbers -- e.g., B64.1 -- to indicate an occurrence somewhere after the phrase. Occasionally items of information occur that are sufficiently unique so that it did not seem appropriate to add them to the repertoire of operators or states; these are indicated by circled footnote numbers -- e.g., around B48, B49. The only other convention adopted can be seen at B30.1. Here each generated digit is tested by TD and one of them, R=5, is rejected. Rather than depict the entire loop between GN and TD, we have simply put a jagged right side to B30.1 to indicate that the B28 box is subordinated to the generator and executed repeatedly. Other examples of this occur at B99 and B103.

Let us see how the encoding goes. We adopt a point of view that anything that makes a difference in the state, as defined by s, should be noted and made into an operator. (The four operators given are the end result of coding, of course, and were not decided upon a priori.) Various other kinds of information, most notably what column is being considered, are not shown in this problem space.

Starting at the beginning (B1) we have an exchange that is really outside the problem space, since it involves clarification of the rules. Not wanting to include it in the first real node, we simply indicate this by a special footnote (1). Likewise we conventionally indicate that the initial board position, X1,

is already set up by the experimenter. For the second node, B5, we have a clear statement considering the two D's, asserting their value, and concluding that T is zero. The coding of this as the operator PC(c1) is clear. Some open questions are 1) when did the inference actually occur; 2) why was c1 considered; 3) was it desired to find the value of T before processing c1; and 4) was it also concluded that t2=1? About some of these questions we do not need to have answers. As to the first question, we only need the approximate ordering of processing, which is quite clear from the protocol. As to the second question we have declared the selection of columns to be internal to each node and thus irrelevant to the problem graph. (Actually, we will return to it later.) The third question is relevant, but we adopt the view that unless specific information is available on the variable desired, we will not record it. Finally, although it is plausible that t2=1 is inferred since 5 + 5 = 10, there is no immediate evidence. However, later behavior (B21) shows that in fact this information was retained.

Turning to the next node, B8, we will find it useful to consider it in conjunction with node B20. In this latter we clearly have a consideration of c2 with the inference of R odd. If we write down what happens before this we have:

B8-B9	Writing prior result
B10-B11	Searching for a next step with no result in terms of our problem space.
B12-B13	Another writing step, when the D in c6 is noticed; conceivably new information is obtained, but certainly no evidence for it appears.

B14-B19 Consideration of c2, c3, A, L and R in the apparent search for a next step. No new information obtained in our problem space.

B20-B22 Processing of c2.

The concern with R, clearly indicated in B18 and B19, lead to the inference that the decision to process c2 is based partly on the decision to obtain some information about R. Thus we code B8 with the goal of getting R. The things occurring prior to B18 all belong within a node: the operations of writing and the (attempted) selection of columns on which to work. If the inference to "get R" were less clear, we would have only a single node for B8 to B22, whose operator would be PC(c2).

It is clear that in B23 and B25 the reasoning used in B20 to B22 is repeated. Why the repetition occurred is not as apparent. It might be that the repetition is to check the processing -- to assure that the inference is correct. That a correction can occur the second time around is shown by the sequence B32 - B35, yielding G even, and the immediate repeat, B36 - B38, leading to the realization that no such inference is possible. However, the repetition may also be determined by the structuring of the experimental situation to get the subject to talk. In any event, we need to create a node, B22.1, for the result of the first PC(c2) and then back up one for the second at B23.

In B26 - B30 there is an explicit generation of the odd digits, following immediately upon the (confirmed) conclusion that R is odd. Thus the inference that GN(R) occurred is not problematic. It is also apparent that the generation does not take into account what values are already used. That is, the already used digit, 5, is generated and explicitly rejected, rather than skipped over. This supports the inference that TD was applied to the output of GN. It is not

as clear, of course, that TD was applied to 1, 3, 7 and 9, since these were OK and no special indication of their acceptability is provided. Thus, some assumptions of parsimony enter into the coding: if TD was applied sometimes and sometimes not, then a process must have existed to make this decision; but this process would have had to perform (uniformly) the same function as TD, namely, to determine if a digit were used; consequently, it is simpler to assume that TD was applied uniformly. (A comment has already been made on the special structuring of the graph for B30.1 and B28.)

B31 signals a pause, since the experimenter breaks in with a prod to talk. Since there is no evidence in what follows B32 that the refinement of the information to R=1,3,7,9 is used, rather than the more primitive, R odd, it is inferred that the search backed up. It is quite possible that some additional processing did go on from B30.1 during the pause, but since we have no evidence for it, we make no explicit note of it. If some new information were obtained, either it should show up at B31 (which it doesn't) or at some later time in terms of some facts for which there is no way to account for how the subject obtained them.

Our purpose is served in the last three pages if they give some appreciation of how one gets from the data to the behavior graph. Most of the instances discussed so far are quite transparent. Similar discussion could elaborate the rest of the graph. Since we have discussed above about 4% of the graph, another 70 pages would be required. Much of it would be equally transparent (and equally dull); a few places would raise serious issues of interpretation. Following the protocol at the end of the paper, we have added notes that discuss most of these interesting cases. They are labeled by the corresponding B-number.

Summary. The problem graph is a projection of the total behavior of the subject into a space of our own devising. Thus, in interpreting the problem graph we must take into account the various possibilities for the true situation relative to what we see in the graph. Let us state first, in strong form, what the problem graph implies: and then follow this with some possibilities that could seriously qualify these statements.

The successful encoding of the S3's behavior into the problem graph of Figure 10 implies:

1. The subject's problem solving proceeds in the set of states of knowledge given in the figure. That is, evidence exists that S3 had these various states of knowledge and that paths exist through them that go from the initial state to the solution.

2. The operators PC, GN, AV and TD account for all transitions to new states of knowledge. (Returns to prior states are governed by other processes.)

3. The operators, along with a set of processes for selecting operators, evaluating states of knowledge for termination, and selecting prior nodes to which to return, constitute a sufficient set of processes for explaining the subject's behavior.

In short, the subject's basic problem solving method is search, hence "trial and error," but in a space defined by intellectual operations of a fair degree of sophistication (PC, GN, AV, TD). These operators provide a definition of basic competence, similar to the abilities that Gagné [8] has tried to identify in somewhat simpler arithmetic situations.

The true state of affairs can deviate from the above assertions in several ways, deriving either from the sort of information actually used by the subject, or the complexity of the processes.

1. So far none of the component processes -- the operators or the selection and evalution processes -- have been spelled out in detail. It may be that the essential problem solving is done in one or more of these. If this were so, the analysis of the problem graph might be termed superficial, since although true enough, it would not explicate the important processing. Note that the basic issue is not the amount of selection performed by the components (e.g., by the operator selection process), but whether problem solving is required: either search in another space; or some other as-yet-unspecified intellectual process.

2. It may be that the analysis has gone too far -- is too disaggregated. Thus, the graph would show the means used to carry out some larger plan or method without giving any clear indication about this higher organization. An analog would be the trace of the machine instructions carried out by a computer in executing a program. Such a situation would reveal itself in the capriciousness of various selections, viewed in terms of locally available information, whenever the information about what to do next available in the structure of the plan or method manifested itself.

3. The worst case is that in which the present graph is epiphenomenal. That is, other processes using different (or at least additional) information would be actually responsible for the problem solving. As a consequence of these other processes, the subject would come to know (and reveal) the information contained in the graph. If these other processes themselves

involved a search process in some problem space, the situation would be exactly like that depicted in Figure 6; if it were of some other as-yet-unspecified nature, a picture is not so easily made.

To illustrate further this possibility, suppose that we had taken as our problem space the external space, described in Figure 5. Figure 12 shows the problem graph we would have obtained, and this indeed is epiphenomenal. It contains very few clues as to the essential processes that are determining the course of problem solving. Indications of this would arise as soon as one tried to specify the rules for operator selection in terms of the states of knowledge. For example, since there is no representation in the external space for the fact that R is 7 or 9, there is no way of saying why R was assigned the value 7 at B61 and 9 at B189. Or again, note that not until B249 does any evidence for a concern with E, O and column 5 occur, although these dominated the subject's attention from B80 to B162 and from B219 to B243, almost half the total time.

Returning to our graph, evidence that it was epiphenomenal would be expected in a failure to find any rules based on the states of knowledge of our problem space for selecting operators and evaluating and selecting nodes.

In all three of the difficulties we have mentioned, in which our chosen space is superficial, disaggregated, or epiphenomenal, the empirical question is whether simple selection and evaluation rules can be found that make use of the information given by the problem space. Examination of these rules is the task of the second phase of the analysis.

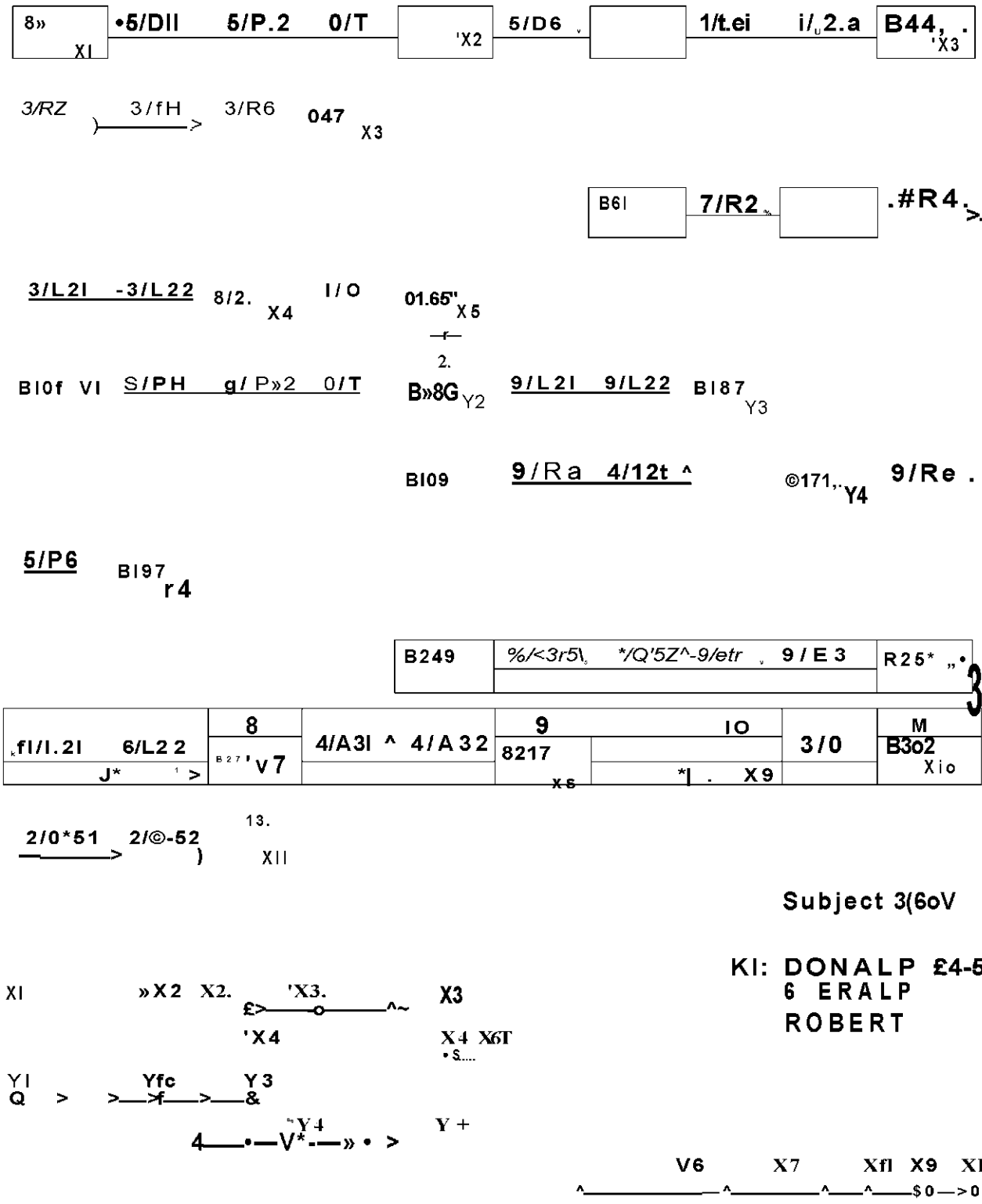


Figure 12: Crypt-arithmetic: Problem Behavior Graph in external problem space.

Second Phase Analysis of the Protocol

We now turn to a more intensive analysis of the component processes assumed by the problem graph. Looked at one way, the graph has segmented the total protocol into 238 parts, each of which has associated with it one of a finite set of behaviors (the operator with its particular inputs and outputs). Each of these nodes also has an associated state of knowledge. If there exist for the subject definite processes for selection, evaluation, and the carrying out of the operators, then within these 238 occurrences enough repetitions of essentially the same situation may occur to induct the processes and have some faith in their reality. It is clear that repetition of decision situation is the key issue, for if each of the occurrences called forth a unique process, then we could never have any verification that a proposed process was in fact the one used. Note, however, that the definition of "the same decision situation" is not given a priori. Each of the 238 states of knowledge is unique. Therefore the amount of repetition is defined after the fact by the nature of a proposed process and the proposed laws of its evocation.

If the essential problem solving is carried out in the space of Figure 10, then we would expect that simple, definite algorithms should exist for the component processes. These might utilize information not in the knowledge state, but only of either a local kind (such as where attention has been immediately before) or of a universal kind (such as properties of integers). So far, except for the operators, we have not been entirely specific about the processes to be performed, other than designating them by the functions of selection and evaluation. It is not clear that the processes should be organized in one-to-one correspondence with these functions. In fact, our first task is to describe a scheme for stating rules.

Productions. Given that our data are a set of correspondences between states of knowledge and the action that resulted, the natural form for a rule is:

cues in knowledge state ⟶ action

This is to be taken as a conditional expression, in that the action (meaning one of the operators along with its operands) will occur only if the cues occur; if these do not occur nothing is implied about what action will occur.

We can consider that the total behavior of a system is made up of a great many such conditional rules:

$$\text{Condition}_1 \longrightarrow \text{action}_1$$

$$\text{Condition}_2 \longrightarrow \text{action}_2$$

$$\ldots \longrightarrow \ldots$$

$$\text{Condition}_n \longrightarrow \text{action}_n$$

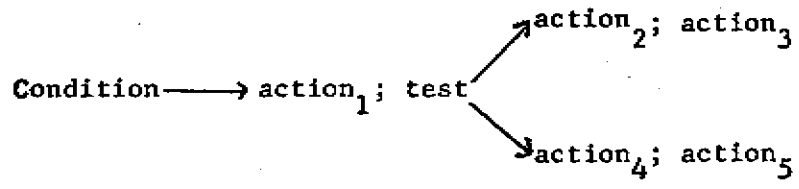
Such systems are often referred to as production systems, each rule being termed a production. They are often used in syntax analysis [6], but also have been made the base of a general theory of algorithms, called Markov algorithms [11].

To get a viable system from a set of productions it is necessary to add some principle to resolve conflicts when the conditions of several productions are concurrently satisfied. The simplest such scheme, corresponds to a simple priority. Execute the first satisfied production in sequence, starting from the top; then repeat again from the top. Other, more complex, principles are possible, such as associating with each production a link to the next production to be considered. But we will try the simplest first. From the viewpoint of inducing productions from the problem graph, the priority scheme is the last feature to become evident since it depends on the total system of rules.

As introduced above, the action consists of the single operation performed at the node. We might consider extending the action to include a sequence of operations:

$$\text{Condition} \longrightarrow \text{action}_1;\ \text{action}_2;\ \ldots\ \text{action}_k$$

Such productions would consist of short plans of action that would integrate the behavior over a series of nodes. They would correspond to the fact that our problem graph was somewhat disaggregated. How much conditionally we admit in the sequence is an open question, and need not be decided in advance. At one extreme the sequence is totally unconditional. Next, the sequence might be terminable if the actions did not produce appropriate outputs for the next action in the sequence. Thus the sequence could be of variable length, but of fixed composition. Next, it might be permitted to have several alternative routes, as in the standard flow diagram with conditional transfers:



In the limit, of course, one could admit an entire system of productions for the action part. This latter involves the notion of a subsystem of productions being evocable in isolated context. With each expansion of the conditionality of the action part of a rule, the rule becomes less and less a unit of behavior.

Since path information is part of the state of knowledge of the problem solver, there is no increase in the generality of the behavior producable as we increase the conditionality permitted. By suitably recording the occurrence of prior actions any of the more elaborate schemes mentioned above can be reduced

to a primitive production system in which only a single action is taken at each production. Indeed, the use of a place keeper in the action sequence, which is required in any of the schemes except the simplest, is just such an encoding of past behavior.

Production system for S3. The production system given in Figure 13 purports to describe the behavior of S3 in the problem graph of Figure 10. These rules were inducted by examination of the problem graph along with various notions of how problem solving might proceed in the crypt-arithmetic task. The notation is that used in describing problem spaces, except for a few additions, which will be noted in the discussion of individual rules.

The productions are divided into four classes on the basis of their function.* The first set, S1 to S5, involves the selection of one of the operators PC, GN and AV. The second set, G1 to G5, provides mechanisms for setting goals. The third set, T1 and T2, acts to terminate a line of search. The selection of the fourth operator, TD, is governed by these rules. Finally, the last set, R1 and R2, permits the repetition of previous paths.

Several new processes are introduced. These all produce outputs that do not change the knowledge state as we defined it -- they produce columns, letters, prior actions to be considered. Thus none of these should have been included in our set of operators. (They do indicate that it might have been worthwhile to operate in an expanded problem space that made attention control explicit.) We define these processes here at the same level used for the operators.

---

* There is no significance to this grouping as far as the operation of the production system is concerned; we have not yet provided the priority order in which the rules are to be considered.

Productions

Selection

S1 v=d*|v←d* → FC(v)=> c; PC(c) (not repeated)

S2 get v|get v=d → FC(v)=> c; PC(c,v)

S3 get l → FA(l)=> c[v]; AV(v); PC(c,l)

S4 get v[constrained]|l free → GN(v) [simple]=> d[first]; AV(v,d) (not repeated)
[-simple]=> ds; [small] → AV(v)

v constrained := v odd|v even|v>d|v=ds[small]

S5 check cs → GNC(cs)=> c; PC(c)

Goal setting

G1 ee? → get ee (immediately)

G2 ee[v]-p → get v (note: ee-p accepted) (immediately)

G3 check ee[new] → get ee

G4 get ls → FL(ls)=> l; get l

G5 ee! → check ee (not repeated)

Terminating

T1 l=d|GN(l)=> d → TD(l,d)=> { + ; l=d-p(ee!) } (not repeated)

T2 ee-p → FA(ee)=> ee'; ee'-p
(except ←)

Repeating

R1 Q=> e[v][unclear] → get v; repeat Q

R2 check ee[old] → FP(ee)=> P; get ee; repeat P

Figure 13: Production system.

FC(v) Find column containing v. The input to FC is the variable (l or t) for which it is desired to find the column. The output, if it exists, is a column that involves that variable. The column currently being attended to is also known.

FA(ee) Find the antecedent of ee. The input is an expression or a variable. The output is the column or relationship that was used in deriving the expression, or that can be used in determining a value for the variable.

GNC(cs) Generate columns of cs. The input is a set of columns; the output is the columns of the set from right to left.

FL(ls) Find letter in ls. The input is a set of letters, ls (not letter occurrences). The output is a letter, l, of the set. The display and current knowledge about letters is available to FL, so it can select, for instance, that l that is still undetermined and occurs a maximum number of times in the display.

FP(ee) Find production that produced ee. The input is an expression that was derived at some prior point in the analysis. The output is the production that gave ee; thus FP is essentially a recall process. (It differs from FA in providing a production, and not just the situation on the board.)

We now provide a brief discussion of each of the productions.

Selection S1. In words: if an expression determining a value (either v=d* or v←d*) has been produced, then find a column, c, that contains the variable involved in the expression, and process that column, PC(c). If no column is produced by FC, of course then PC is never evoked; that is, the sequential action is conditional on appropriate outputs being produced by prior actions. This production represents the ability to take new information and apply it elsewhere to get yet more new information. Its evocation depends only upon some new information occurring. Once such a production has been executed it will not be repeated. (This constraint does not apply to repetition forced by R2.)

Selection S2. In words: To get a variable, v, or a specific value for a variable, v=d, find a column, c, containing that variable and then process that column for the variable, PC(c,v). Again, if no column is produced by FC, PC is not evoked. It can be applied repeatedly to the same variable to gradually accumulate information about it. The goal can either be to get information about a variable, or to obtain some relation, as in, "get t5=1." Both S1 and S2 have fundamentally the same action sequence; yet they derive from quite different concerns. (The fact that one has PC(c), the other PC(c,v) has little operational significance, since only rarely is there ambiguity over what variable is to be used in PC.)

Selection S3. In words: To get l, find a column, c, from which l can be derived; then assign a value to the other variable, v, that exists in c, after which l may be determined using PC(c,l). Brackets are used to indicate a dependence, as in c[v], or a condition, as in l[constrained](S4). S3 provides an alternative means for obtaining a goal. It gives the appearance, whether justified or not is another question, of being less arbitrary than simply assigning a value to l.

Selection S4. In words: To get a variable, v, that is constrained in its set of possible values, or to get the value of a free letter, generate its values; if the constraint is simple, then generate only the first and assign it to v; if it is not simple, but there are only a few of them, assign a value, AV(v); otherwise do nothing. This production appears to be complicated because it consists of two action parts, both involving generation and assignment. The issue is whether to generate all the values or only the first. This decision is based on the process of generation itself -- whether simple, such as all digits

greater than five, or complex, such as all odd digits greater than five. Thus the test is carried out within the process of generation, and not at the point where the production is evoked. Additional conditionality occurs in the case of full generation; namely, that the action sequence only continue to assignment if the set is small. One rationale for this is that the large set makes the choice seem too arbitrary, but such an assessment occurs only when the full set is actually generated.

Selection S5. In words: To check a set of columns, generate the columns from right to left, executing PC(c) on each. This production is only evoked once during the course of problem solving; namely at the end. However, it is sufficiently clear that it is included anyway (and in fact governs a rather long sequence of behavior).

Goal setting G1. In words: If an expression is relevant but unknown, ee?, then set up the goal of getting it. This production must be applied immediately after ee? is produced, or not at all. That the expression ee is unknown only gets evoked because some other process attempts to use ee and finds that it is unknown. Thus, there does not exist in the knowledge state expressions for all possible things that are unknown. This illustrates that the knowledge states are not to be taken in the sense of "all things that the observer can infer the subject could know are true."

Goal setting G2. In words: If it is known about a variable, v, that a certain fact is not possible -- i.e., ee[v]-p -- then set up the goal of getting v. This production must be evoked immediately upon producing the negative information, or not at all. The additional note in Figure 13 affirms that this production is not an attempt to deny the new information (which would then

lead to check ee), but accepts the information and looks for what other value v might have.

Goal setting G3. In words: To check an expression, ee, that is new (that is, has not been derived before), set up the goal of getting it. Although it occurs only rarely, it can happen that a fact becomes assumed or known without there being any specific prior derivation of it. This production simply bridges the gap between the goals "check" and "get" in these situations.

Goal setting G4. In words: To get a set of letters, find one (according to FL) and set up the goal of getting it. This production simply moves from a set to its members. Its role is essentially that of finding something to work on when all else fails, since the initial problem is stated as getting 1s.

Goal setting G5. In words: If an expression has been critical in determining some process, as expressed by ee!, then set up the goal of checking ee. The production has been stated unconditionally, but clearly how certain the subject is about ee, expressed in some manner, will also condition its evocation. For instance the subject will not check D-5. Some, but not all of this is taken care of by not permitting repetitions of the production.

Terminating T1. In words: Anytime a new digit is derived as the value of a letter, l=d, evoke TD. The result is either +, indicating that everything is OK, or the expression that l=d is not possible along with the statement of the critical fact (ee!). T1 is also evoked by the generation of digits that occurs in the context of obtaining values for a variable. Note that there is no similar check on the values of a carry.

Terminating T2. In words: If it is determined that an expression, ee, is not possible, then find the expression, ee', that was used in deriving ee and declare it not possible also. Clearly this cannot be evoked on assignments and these are excluded. This production provides backtracking down a succession of implications when one is finally discovered that is contradicted.

Repeating R1. In words: If the result of a process, Q, is unclear, then repeat Q, setting up as a goal to get the variable that was involved in the unclear statement. The process in question is normally PC, but can be others on occasion. As with the terms "simple" and "small," the term "unclear" requires further delineation. Addition of two digits, as in 5 + 5 = T, must be "clear"; and complex determinations, such as R odd from 1 + L + L = R, must be "unclear," at least the first time encountered. Likewise, processes that lead to contradictions or from which no definite conclusion can be drawn should also be "unclear."

Repeating R2. In words: To check an expression that has been previously derived, find the production involved in that derivation and repeat it (after setting up the goal of getting the expression). R2 implies some memory of production occurrences.

The production system of Figure 13 is not complete. First, it is still necessary to specify the priority system to resolve conflict between productions. These conflicts will occur frequently, since some productions have identical conditions; e.g., S2 and S3. Furthermore, other productions, such as S1 and S2, although different can both be satisfied at the same node.

More important than the priority ordering is the fact that not all of the nodes of the problem graph are intended to be covered. The productions arise

from the regularities that are found; and nodes that are either idiosyncratic in their behavior of sufficiently unclear do not give rise to production. Thus, the production system is incomplete from a task point of view. It is not capable of solving the task of DONALD+GERALD, nor many other crypt-arithmetic tasks, without augmentation.

The productions of Figure 13 form an integrated system in at least one non-obvious way. The context in which the conditions are tested is provided by the products of the productions themselves, at least insofar as the system is complete. Thus the conditions can be adequate to make the right discriminations within the restricted context, even though they would be non-discriminative within a larger context.

We can summarize these rules in relation to the evaluative and selective functions that are required for any system that is capable of generating a problem graph.

Operator selection. The function of Si, S2, S3, S4, S5 and "Tl is to select which of the four operators, PC, GN, AV and TD is to be performed. Two of these, S3 and S4, specify a pair of operators to be applied in sequence, if all goes well. In addition, Rl also selects an operator, as does R2' indirectly.

Evaluation. Terminal nodes occur either on the occurrence of an impossibility, ee-p, or on the product of an operator being unclear. Termination on impossibility is implicit in that none of the productions that can select new operators do so with ee-p as a condition. Those that do respond to ee-p imply the selection of an old node from which to proceed, thus terminating the current one. Positive evaluations ("go on") are implicit in all the selection rules, and there is no special process to determine this.

Node selection. T2, R1 and R2 carry out the function of node selection. R1 and R2 determine the actual node; T2 simply backs down the tree eliminating nodes as candidates for starting over. Implicit in T2 is the retention of path information. Implicit also is the principle that if you are at a node and it isn't prohibited, then it is selected. This, in conjunction with path memory, is equivalent to a depth-first search strategy.

Goal setting. The productions that create goals have been pulled together already in G1 - G5. Three of these, G1, G2, G5, along with R1, form one component of means-ends analysis; namely, that of immediate reaction to difficulties by setting up goals of dealing with it. These produce the phrase structuring of behavior by setting up subgoals within subgoals.

Empirical Evaluation.

Given the production system of Figure 13 we want to evaluate to what extent it characterizes the problem graph. The first step is to write down for each node what production, if any, seems to have been evoked. This is done on the protocol itself, rather than on the problem graph, to permit comparison of the productions with the verbal behavior that is the main evidence for them. We require a few conventions.

First, it is often the case that we can not discern the information on which the production conditions are based, or can discern neither condition nor action. We use a question mark (?) to indicate these cases; this is to be distinguished from the question mark associated with outputs, as in production G1. Sometimes we make a comment in English, especially when the behavior is clearly outside the problem space and is coded by a footnote on the problem graph.

Examples:

B1 ?: (ask Exp. about rules) (outside space)

B137 ?: → get R (evidence for action only)

B39 ?: (no production in evidence)

Second, several of the productions cover more than one node. We use an up arrow to indicate at a node that it is covered by the production named at the preceding node.

B22.1 R1: PC unclear → get R; repeat PC

B23 ↑ : PC(c2,R)=> R odd

Third, on occasion we need to indicate that TD is applied to several members of a generator. We do this by using a variable (d) for the input to TD. Sometimes this appears to interrupt another production, since the TD is being applied to each output.

B59 S4: get R → GN(R)=> 7,9

B60 T1: R=d → TD(d,R)=> +

B61 ↑S4: AV(R)=> R←7

A question of more substance arises from the fact that once a production occurs at some point in the protocol, a sequence of production occurrences is automatically generated by the outputs of one becoming the inputs of the next. Sometimes several members of this implied sequence happen within a node. An example is at B8, where FC=> ϕ, so that two S1 productions occur within the node; another is at B7 (and several other places) where T1 produces an occurrence of TD for which no node occurs in the graph because TD=> +. The question is whether to consider the set of production occurrences fixed in advance by the number of nodes in the problem graph or to expand it by the additional implied occurrences.

We do the latter; consequently expanding the data set from 238 nodes to 275 production occurrences (about 15%). We can call each of these places a context; i.e., a place where a production was (or should have been) evoked.

Now we are in a position to make an accounting of the productions. Considering each rule separately, the protocol provides all the positive occurrences. But there could also be a number of other contexts in which the conditions of the production were satisfied, but either some other production was evoked, or no recognizable production occurred (?). These are the negative instances. This information can be obtained by asking for each context (i.e., each place where a production or ? occurs in the protocol) and each production whether its conditions were satisfied. The data appears at the end of the protocol in the State → Production Table. This table was constructed by first recording the expressions produced as output from the productions that actually occurred, and also the goal stack. The horizontal lines in the table simply indicate when an exploration terminated and the arrows at the left show at what point the new exploration starts out. With this state information recorded, each production was matched at each context point and one of five marks made:

+	The conditions are satisfied and the production is evoked;
↑	the production was evoked in a prior context and is still in effect;
-	the conditions are satisfied but the production is not evoked.
?	the conditions may have been satisfied, but the production is not evoked.
blank	the conditions are not satisfied.

Notice that it is not possible for a production to be evoked but its conditions

not satisfied; for we do not recognize a production only by its action part. For example, in B185 the action part is taken to be similar to that of S5, but a ? coded for the production. However, we accept as evidence for the condition part either explicit data from the protocol or implicit data output by prior productions that have been evoked.

Figure 14 summarizes these data in a matrix. The labels at the top, hence columns, are the productions that did occur in a context; the labels at the side, hence rows, are the productions that could have occurred in the context. Thus the entry (i,j) of the i^{th} row and j^{the} column gives the number of times production i could have occurred, but instead production j did occur. The total number of times the j^{th} production did occur is given by the diagonal entry (j,j). Each entry has two possible numbers. The top one is the main one; it counts the contexts in which the conditions of a production were definitely satisfied, hence marked with a plus (+) or minus (-). The lower number shows the additional contexts which were questionable, hence marked with a question mark (?).

For any pair of productions, the symmetric two off-diagonal entries (i,j) and (j,i) tell how they fared against each other. If there were no joint occurrences, then both entries would be zero; in this case they have been left blank. The zeros that do occur imply, then, that choices existed, but the column production was never chosen. Thus, we see that in the 10 cases in which either S1 and S2 could have occurred, S1 did in 8, and S2 and 2. Similarly, in the 14 cases in which either S3 and S4 could have occurred, S4 did 14 times and S3 did not occur at all (0).

These strong biases towards one production dominating another are consistent with the imposition of a priority ordering on the production system.

DID OCCUR

COULD OCCUR		S1	S2	S3	S4	S5	G1	G2	G3	G4	G5	T1	T2	R1	R2	own ↑	↑	?
	S1	35	2	0			3			0		24		3			18	6
	S2	8	29	9	15		6	2			4	13		2			11	34
	S3	4	16	9	14		3	1			4	9	1	7		16	19	19
	S4		2 2	0 2	24		1	0 3		0	1	3	0	1 2		4	4	2 3
	S5					1										6		
	G1	0	0	0			11			0		0						3
	G2		0	0	1			8		0	5		9	11				9
	G3					1			2									
	G4	26			1		5	4		4	2	18	8	6			3	9
	G5		0	0	0			0 1		0	7		4 4	3 2				3
	T1	1 2	0	0	0		2 1			0		31		2 2			1 3	2
	T2			0	1			5		0	4		10	11				9
	R1	0	0	0	0		1	0		0	0 1	0	0	18		18		2 1
	R2														4			
own	↑															44		
	↑																	
	?																	38

Figure 14 : Crypt-arithmetic: Matrix showing production conflicts.

The necessity of this was discussed earlier, although it was left somewhat open whether a more complicated principle of conflict resolution might be involved. We can attempt to impose a linear ordering on the productions, always placing production j above production i if the (i,j)th entry is greater than the (j,i)th entry. In general this can lead to difficulties if there are intransitivities in the data, such that i before j and j before k, but k before i. However, it turns out we do not have to face this problem, since no intransitivities show up in the data. Figure 15 shows the reordered matrix. No entry above the diagonal is greater than its symmetric mate below the diagonal (there is one tie between T2 and G5).* Note that the data are not everywhere equally strong, and that in several cases there is no data at all to specify the ordering. The three productions that are essentially isolates, R2, S5 and G3, are placed at the top, but they could equally well be anywhere. The one additional rule we impose is that once a production is evoked there is no opportunity for evoking new productions until its action part has run its course. Operationally, this means that ↑ has top priority of all.

The consistency of the ordering does not imply that a priority system is without error. Every non-zero entry above the diagonal in Figure 15 gives

---

* In constructing Figure 15 only the definite comparisons (the top entries in each cell) have been considered, the questionable ones (? in the State → Production Table) being ignored. If these were to be added, the picture remains about the same. The tie between T2 and G5 would be broken in the direction opposite from the way we have it; in addition the ordering between G2 and S4 would be reversed. Thus the four consecutive productions, G5, S4, T2, G2, would become reordered as T2, G5, G2, S4. This would introduce one intransitivity in that the single comparison between S4 and T2 showed S4 to be preferred.

DID OCCUR

COULD OCCUR		R2	S5	G3	G1	R1	T1	G5	S4	T2	G2	S1	S2	S3	G4	own ↑	↑	?	Ex
	R2	4																	4
	S5		1													6			7
	G3		1	2															2
	G1				11	0	0		0			0	0	0	0			3	11
	R1				1	18	0	0 1	0	0	0	0	0	0	0	18		2 1	36
	T1				2 1	2 2	31		0			1 2	0	0	0		1 3	2	31
	G5					3 2		7	0	4 4	0 1		0	0	0			3	7
	S4				1	1 2	3	1	24	0	0 3		2 2	0 2	0	4	4	2 3	28
	T2					11		4	1	10	5			0	0			9	10
	G2					11		5	1	9	8		0	0	0			9	8
	S1				3	3	24					35	2	0	0		18	6	35
	S2				6	2	13	4	15		2	8	29	9			11	34	29
	S3				3	7	9	4	14	1	1	4	16	9		16	19	19	25
	G4				5	6	18	2	1	8	4	26			4		3	9	4
own	↑															44			
	↑																		
	?																	38	

Figure 15: Crypt-arithmetic: Reordered matrix according to priority rule.

cases where the data shows that the production lower on the priority order was in fact selected. The further off the diagonal it is, the larger is the inversion.

Figure 16 provides a way of looking at the total performance of the production system. Some productions account for many items of the behavior; some for only a few. Thus, one can think of adding new productions, each of which increases the total amount of the protocol described, but with a diminishing marginal utility (especially, if we view the extra production in the total description as a "cost"). In Figure 16 the productions are recorded according to this marginal utility. R1 comes first with 38 instances; G3 comes last with 2.

As we increase the number of productions in the system, two other changes occur simultaneously. The total number of contexts increases. Originally 238, the number of nodes in the PBG, it gradually increases to 275. As described earlier, this is due to carrying along subsequent evocations of productions to implied products as long as these are not contradicted by the data. The top line in Figure 16 is drawn 8 below the total, running from 230 on the left to 267 on the right. This is called the relevant total, since eight contexts are clearly outside the problem space we are dealing with, involving conversations with the experimenter, discussion of the rules, and so on. Thus the amount of the protocol to which some production applied must be viewed against this varying total. It starts at 16% with only R1 and climbs to 89% at the end.

Simultaneously with the increase in the amount of coverage we begin to get positive errors; that is, contexts in which the wrong production is evoked. This is shown by the lower curve. Following the solid line, which corresponds to the definite errors, it starts out at zero for R1 alone (since there is no possibility of conflict) and clumbs to 23 errors for the total system. The upper,

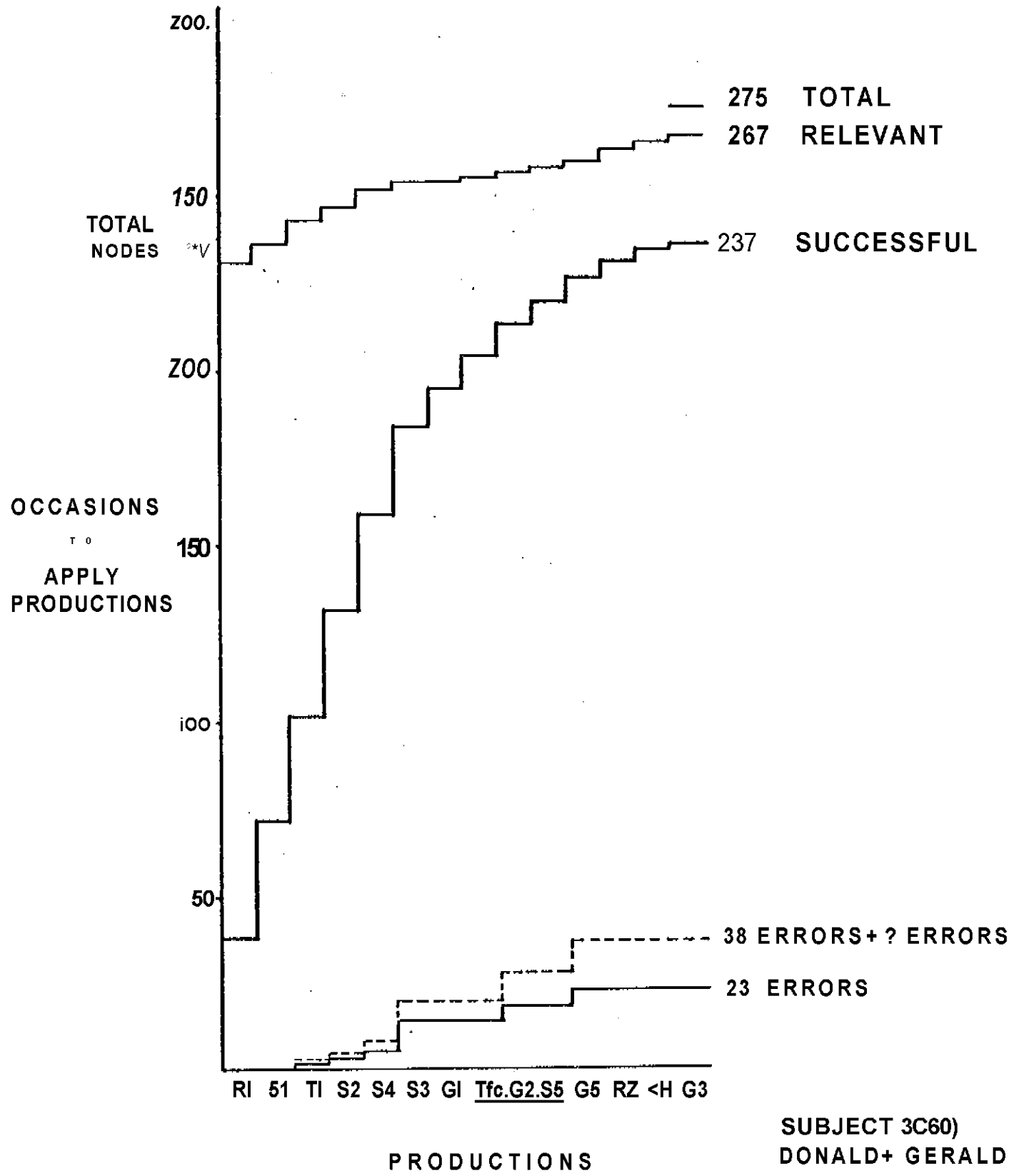


Figure 16: Crypt-arithmetic: Summary of performance of production system.

dotted, curve adds in the errors that have been labeled questionable; it rises to 38 for the total system. These error figures come from adding up the appropriate subset of entries above the diagonal in the matrix in Figure 15.

If we dock the system for its errors, then we might say that it had described (237-23)/267 = 80% of the protocol. In fact, this is not very informative, since it mixes errors of omission (30) and errors of commission (23)*. More important, we should be interested in understanding both of these types of errors to see why the system was unable to do better.

Errors of omission (?). The failure to find a production (or sequence of productions) that fit a segment of protocol has various causes, ranging from lack of data to lack of ideas about how to construct a mechanism that will do the task. Below we discuss the types of failures under a series of ad hoc headings that seem to be indicated by the omitted instances. Two instances illustrate more than one concern, so the sum of the listed labels is 40, rather than 38, the total number of ?-nodes.

Although we do not deal with it as a separate category, many of the failures are due implicitly to lack of information in the protocol. As a similar point, it will be noted that most of the failures occur at terminals (26) or beginnings (8). However, the causal arrow undoubtedly runs from failure to interpret the protocol to the existence of a termination, and not vice versa. Given that lines of exploration are not excessively long, once the thread is lost, for whatever reason, it is unlikely to be picked up again until the subject has stated a new line. In particular the apparently obvious inference that the

---

* If we also add in the questionable errors, we get (237-38)/267 = 75%.

model is in most trouble on "evaluation," because most ?-nodes are terminals is false, being due to this artifact.

Evaluation. (B39, B119.1, B143.1, B183, B222.1, B238.1.) There are several nodes where there is explicitly an evaluation -- i.e., we do not just loose the thread -- but we are unable to incorporate it. The decision to "forget it" at B39 is most explicit. B119.1 is an example where the subject clearly runs out of gas; but it is not clear how to stop the production system. In B143.1 he clearly enunciates his options (R↚7, R↚9); then decides to remain with R↚7. Part of the same dilemma is the decision to switch at B183, and then the decision to switch back at B238.1. In all of these the subject shows some persistence, then finally backs down. We have no representation of this kind of evaluation process. A related pair of evaluations concern t5>1. At B222.1 the subject opts not to investigate t5>1; whereas a little later at B230 he does. The system only founders on one of these options, but in fact we face a general problem formulating the effect of multiple tries at the same decision.

Selection of nodes in stock. (B97, B137, B175, B178.1, B217.) Besides the immediate path and the initial situation, the subject appears to keep two additional nodes available, the decision situation around R and the one around E. A look at the total PBG in Figure 11 reveals three substantial breaks in the continuity of the search (lines 16-18, 23, and 31). (The other candidates, lines 7, 17, and 47 are all intermediate nodes in the process of backing down.) These breaks are all oscillations between E and R, as is the one other that is hidden, when the subject decides to explore R↚9 at line 34. We do not have any formulation that predicts these jumps. It might be handled partly by appropriate evaluation (a difficulty already noted) in conjunction with a better

handling of the goal stack. That is, the return to E after a jump to R is because "get E" never leaves the goal stack, but is simply pushed down.

Memory mechanisms. (B158, B188.1, B275.) A substantial number of memory mechanisms are built into the problem space as we have used it: knowledge of the current states, path memory, nodes in the stock, and a goal stack (see the State←Production Table). However, in a few places there is clear evidence of additional memory. At B158, the subject recalls that E=0 leads to a contradiction (at B136). Although we require memory of values assigned for the operation of TD, we do not have this more elaborate recognition that exactly the same path will be followed. At B188.1 we have an immediate recovery from the error of setting L←9 to setting R←9. At B105 the error was the same, but the recovery was much more involved. Although plausible that he would not stumble so badly the second time, we have nothing in the production system to make this sort of change. At B274 we have, first a forgetting that is only partly explainable by the system, and then a recall for which we have no mechanism.

Although the examples are few, the impression -- strongly reinforced from much else that we know about human memory -- is that we need a more generalized memory that is not completely tied to the service of the problem space. On occasion it would deliver some information of useful value, although it is not clear it could be relied upon to do so.

Extensions of production system. (B90.1, B100, B110, B124, B148.1, B155.1, B201, B237, B238.1, B244, B303.) We include here examples of mechanisms that seem close to being incorporated in the production system, but which we were unable to formulate properly. An important example are operations that are digit oriented. That is, instead of selecting a letter and finding a digit,

select a digit and find a letter. The two examples having some of this flavor are close together. In B237 the subject, after asserting E=0-p immediately asserts A=0-p; it appears the subject is finding all the things not possible with 0, now that he knows that 0 is used. In B238.1 the subject seems to be deciding whether to assign 9 to R or 9 to E; that is, the letter to be assigned to 9 is the focus, not the digit to be assigned a fixed letter.

A second important type of mechanism is induction, where the subject sees the general case from an examination of several specific instances. The one clear example is at B155.1, where the production system does not have too much difficulty incorporating the generation of values of 0, but cannot make the inductive step to "Actually, that's almost the case no matter what the situation is --."

Most of the other mechanisms seem of lesser moment. There is a third return to getting E due to unclarity (B90.1), which does not quite fit the production system. At B100 the subject makes the connection between E even and E cannot be 9, at least enough to go back and check. There is no place yet in the system for such partial glimpses. At B110 the gap should be successfully bridged by setting up a goal to check, but it does not seem to work. B124 (and its repetition, B148.1) apparently involves seeing that, since one is already assigning a value to get E (S3), one would be better off to use 0 rather than A. The key seems already to be in the behavior of the production system in producing 0?, since if the prior scheme didn't lead to determining 0, then it would be preferable to assign 0 directly. However, the gap from 0? to S3 on c5 doesn't quite close. At B201 there is apparently a switch to exploiting R←9, rather than getting E, once R←9 becomes the focus of attention via R←9!. At B244 it is

quite clear that the subject initially thinks in terms of the full digit set (ds) and then transforms his analysis to use the digits remaining (fds). At B281 - B282 the same reasoning processes is repeated but more smoothly (and without causing an omission error, as it happened). The production system could probably be extended to handle this; the main requirement is to have the internal structure of GN and AV explicit. Finally, at B303 we have no means of evoking the final checking operation. We could have written a production that reacted to the final positive solution. Actually, this is only indicated in a negative way in the current system, by FL=> ϕ. Thus, two productions would be required probably, one to say "Eureka," the other to say, "if Eureka, then check."

Interaction with External Problem Space (B185). The main justification for not making explicit the writing operations is that they do not appear to affect what goes on in the internal problem space. B185 is one exception, not only in terms of the ability to have two external displays, but also in having a copy operation that takes information from one to build the other. A second example, not recorded as an omission, occurs at B12, where the response to finding D in c6 is to write 5 for D, rather than to process c6. We might have found more interaction, especially in attention control, if we had incorporated a set of writing productions and thus tried to determine the conditions under which writing would be evoked.

Extensions of the Problem Space. (B1, B39, B50, B169, B224.1, B225, B317, B318, B319, B320.) Eight of these situations (all but B39 and B169) were so far outside the problem space as to be irrelevant to evaluating the performance of the production system, and deleted from the total set of contexts. Several of these instances, B1, B50, B169, B224.1, B225, and B320, involve

interactions with the experimenter. To handle these involves a quite new problem space (assuming such were the appropriate construct), and one that should not be constructed on such skimpy data. However, given that one had developed the general characteristics of such a model elsewhere, one might import it here with good effect. Contentwise, the extensions cover the areas immediately bordering the task itself: concern with the rules (B1, B50); concern with new methods (B39); concern whether the top goal is a single solution or all solutions (B317, B318, B319); and the exit from the problem space to the larger world (B320). Some of these areas conceivably could be developed as problem spaces, but they would be more complex than the central one we have worked with. For example, development of a space of methods for solving crypt-arithmetic problems is decidedly non-trivial.

As to other types of behavior in this group, B169 is the one clear example of the injection of emotionally toned behavior. B224.1 and B225 are task oriented behavior by the experimenter.

Blank periods (B30.1, B85.1, B103.1, B114.1). The last group of omissions are those that show extended periods of silence, usually broken by the experimenter asking the subject to continue to talk. Clearly, the PBG of Figure 10 does not show the final extension of exploration. However, if critical information were obtained during this period, we would expect it to show up at some later time. There is no clear evidence of this. It seems more likely that these silences constitute periods of not knowing how to proceed.

Errors of commission. The 38 errors (23 definite, 15 questionable) in which the wrong production was evoked according to the established priority scheme are best discussed by considering the various pairings. Table 1 breaks

Error pair	Separation	Definite errors	Questionable errors	Nodes
S3/S2	1	9/16		B40,B104,B120,B125,B144,B149,B186,B189,B218
G2/T2	1	5/9		B158.1,B203.3/2,B227.1,B229.2,B243.1/2
T2/G5	2	4/4	4/0	B106.2,B203.2,B234.1,B243.1; ?B136,?B229.1, ?B238,?B269
S2/S4	4	2/15	2/0	B116,B139; ?B111,?B299
S2/S1	1	2/8		B85,B290.1
S1/T1	5	1/24	2/0	B207; ?B129,?B278
G2/S4	2	0/1	3/0	?B158.1 (see G2/T2), ?B265,?B269/2
S3/S4	5	0/14	2/0	?B104 (see S3/S2), ?B218 (see S3/S2)
G5/R1	2	0/3	1/2	?B94
G2/G5	3	0/5	1/0	?B227.1 (see G2/T2)
		23/	15/	

Table 1. Errors of commission

the errors out this way. Taking the top row, S3/S2 indicates that we are dealing with cases where S3 was chosen over S2, although the priority ordering (Figure 15) indicates that S2 should be chosen over S3. The next column gives the separation in the priority ordering; in this case they are adjacent. The third column gives the number of choices both ways. Thus in 25 co-occurrences, S2 was chosen 16 times and S3 9 times; thus the number of errors is 9. The next column gives similar numbers for questionable errors, although in this case there were none. Finally, the nodes at which these errors occur are listed. In the rows where questionable errors are recorded, we prefix the node with a question mark (?) to distinguish them from nodes with definite errors. More than one error can occur at a node. This happened four times, in each case a questionable error occurring along with a definite error. We have noted these cases in parenthesis; e.g., the bottom row shows that the single error of G2 over G5 occurred also with G2 chosen over T2.

S3/S2, S3/S4. The difficulty here is that we do not have the appropriate discriminators to tell when S3 gets evoked. The 9 errors constitute all occasions on which S3 was evoked. S3 is clearly a secondary method; it never gets evoked until after S2 has been tried at least twice.

The total number of errors (9) is deceptive, either as an "amount of error" or as a sample from which to diagnose what is wrong. There appears to be only three essentially independent evocations of S3: on R at B40, on E at B120, and on E at B218. The rest involve various degrees of repetition. B104 repeats B40, as does B186 when the subject starts over with R←9 instead of R←7; and B189 is an immediate repeat of B186 due to an error in executing the latter. B125 is the shift of B120 to try it on c5 with O rather than on c3 with A;

this whole pattern is repeated in B144 and B149. Consequently, how many errors have really occurred here is quite uncertain. They are not independent, but the acts of repetition themselves are part of what is to be explained.

G2/T2, G2/S4. G2 is the production that converts a failure into a goal of establishing the value of the variable just affected. Of the 8 occurrences, all but one are implicated in some error, so that it clearly is of marginal efficacy. Even more significant, when a moderately careful account is taken of goals (see State-sProduction Table) 5 of the above 8 already have the goal sitting at the top of the stack. Of the three remaining cases one (B243.1/2) perhaps should have the goal in the stack, in that the system has been attending to obtaining 0 for some time before running into the difficulty that sets the goal of obtaining 0. In the other two cases (B203.3/2 and B229.2) a genuine switch of attention is made by G2. Both of these involve, the carry and are among the most obscure passages in the protocol. In short, with only slight modifications of the system one might dispense with G2 altogether.

T2/G5, G2/G5, G5/R1. The entire issue here is under what conditions a critical feature will trigger off the attempt to check it. As we observed earlier, we did not add additional discriminating conditions to G5, even though it is clear that discrimination is necessary. Thus, all four of the questionable errors on T2/G5 as well as the single G2/G5 are due to the subject never checking D*-5! or T<-0 ! (except with the latter, after the experimenter brings it to his attention). These are not open to much uncertainty. One can find plausible reasons in the four cases of definite error why there is little sense to checking. However, no clear pattern emerges, especially when viewed

against the times when checking is evoked. More generally, there is a relatively thin line between getting a value and checking a value, and our explication of this is only marginally satisfactory.

The one case of G5 over R1 is with noting, since it is one of the few places where a critical feature is generated by a process other than TD. The protocol at B94 is quite clear on the attention directed at t3=0. However, it is possible that B95 reflects a much more general switch to a concern with R -- this happens elsewhere in the protocol. In fact, B95 provides a nice instance of ambiguity over the antecedent of "this."

S2/S4. All four errors in this case involve repetition in some way, and show that our system is not explicit enough about exactly what things are remembered from the past and how processing is handled when guided by the past. The two definite errors (B116 and B139) both involve repeating the final path of a previous branched exploration. In both cases this implies not repeating the generation of values for the constrained variable. In one questionable error (?B111) it is unclear where the starting point is in repeating R; even if it were started from R odd, the previous concern about whether to repeat the generation still applies. The issue in the last error (?B299) is a little different. In obtaining a value for B via PC(c4), the subject is sidetracked because of an error in determining the set of available digits (fds). Having cleared this matter up the subject returns to get B. He now has both a simple way to get to B via a repeat of PC (which is what it does) or he can generate values from the restricted set and assign one of them. This kind of choice only occurs at this one point in the protocol. In all other places where S4 occurs there is no such alternative.

S2/S1. One of these errors (B85) is simply ambiguous. Both S1 and S2 lead to exactly the same result; namely, the determination of t6. The other error (B290.1) occurs in the midst of a context of processing larger than the new information that 1 is carried out of column 4. This test sequence requires first deriving t5=1 and then backing off to test N. Our production system is not constructed to back off this way. Even if one tried to model the method by a direct recall of t5=1, the error would still crop up. The system has not captured the higher level of organization adequately.

S1/T1. In all three of these errors, the main question is whether the produced relationship is new or not. It is clear generally that the subject does not evoke TD on old material. As we will see in the final summary, the issue is dritical for the subject -- and not just for us -- since the failure to evoke TD on E=0 at B129 and again at B207 is one of his major errors in problem solving. The definite error here is at B207; however, this may well be an analogous repetition for the R←9 case of the prior processing for the R←7 case. The correspondent of B207 is B129. This seems like a questionable error (for us), since E=0 has already occurred several times. The last questionable error (?B278) occurs very late in the protocol when E=9 and A=4 are rederived; it is not clear whether they should be treated as new or not.

Any discussion of errors is only partial, and is fundamentally biased since it takes place against a background of positive choices about how to fashion the production system. A number of these errors could have been transformed into non-errors by modifying the production system. Of course an equivalent number (actually somewhat more, in the cases investigated during the

course of analysis) of errors would have shown up elsewhere. Still, this enumerated discussion provides some feeling for the places where the system is weak, and for what some remedial actions might be.

Basic processes. In the analysis so far we have introduced a set of processes that have been defined only by rough input/output descriptions: the main operators, PC, GN, AV, TD, and the auxiliary processes, FC, FA, FL, FP, GNC. These descriptions were sufficient for the task required; namely, to permit identification that a process of the specified type had occurred in the data. One can determine that a column is being processed, and even know exactly the information output, without being able to specify by what means that output was determined.

The use we have made of these processes, both in the PBG and in the productions, implies that a single process is to be associated with each name, and not merely that these names stand for types of processes. That is, these processes are to be subroutines. We should be able to write down expressions for them in some process language, such that the entire variation of output is determined by the variation of input. Stated one other way: to predict the output of the processes we should not have to appeal to information about the context in which they occur other than the specified inputs.

Insofar as this fails to be the case, the total errors in our description of the subject are underestimated, since some production occurrences will be judged correct when in fact any algorithmic specification of the basic processes consistent with other occurrences leads to an error. Consequently, the errors in the basic processes require investigation. They should not necessarily be

combined with the errors in the production system, since it is quite possible for the productions to be correct, even if the basic processes are in error; e.g., errors in arithmetic are possible. In the present context there is unlikely to be sufficient evidence to predict why such errors are made or when. In such cases, the error is assigned properly to the processes underlying PC and sheds no doubt on the validity of the productions that use PC. At some point, of course, if such errors are too numerous, they implicate the entire system. But our independent knowledge that errors of perception, memory, and elementary processing occur makes division of errors appropriate.

To fully explicate the basic processes requires postulating yet another set of processes (call them BB for basic-basic processes) in terms of which the basic processes can be described. Consideration of the basic processes, shows the BB-processes to be at the level of elementary operations of perception, immediate memory and accessing of long term memory. FC, for example, involves a visual exploration of the board under the direction of already assimilated information about the structure of the task display; it might take less than a second. Even the most complex of our basic processes, PC, involves operations of the order of adding a pair of digits and recalling the known properties of a letter (e.g., R>5). Thus, a model for the BB-processes is in fact a detailed model of immediate memory and immediate processing.

This is too large an undertaking for the present analysis, requiring extended consideration of the available experimental material on immediate memory and processing. However, we can proceed part way, if we permit ourselves to be considerably more informal and incomplete. We can examine the input/output correspondences for evidence of inconsistency or complexity of processing that

seems incommensurate with the postulation of these processes as basic.

We give below, grouped into six subsections, a discussion of each of the nine basic processes. The numbers in parentheses in the section titles give the number of occurrences of each process. Each section also has a table that provides a listing of each occurrence with its input and output.

PC(88). The most complex of the basic processes by far, PC, also occurs often enough to give considerable evidence about its nature. If PC were simply a routine for adding pairs of numbers (as in 5 + 5 gives T=0 and t2=1), concern over its internal mechanism would reduce completely to concern for how a human does arithmetic. As noted above, this is indeed an issue involving the detailed structure of immediate memory, and the detailed handling of attention. But much more is involved, since PC generates a rather wide variety of final responses. In fact, the power of S3's problem solving hinges strongly on the sophistication of PC. (This is evident if S3's behavior is compared with that of subjects whose PC admits only of simple arithmetic.)

The diversity of processing apparent in Table 2 implies that even if PC is in some sense a single subroutine, it is a highly conditional one. Hence, the key question is whether some uniform scheme of processing can yield this diversity. To provide some evidence on this we give in Figure 17 a sketch of a production system that might perform PC. There is a general progression through five stages of processing. First, the variable about which information is to be obtained (the unknown) is determined. This is done by the subsystem called U, which consists of the four productions, U1 to U4. It produces a value for u. If PC is entered with a goal already set -- e.g., get R --

N	Item	Goal	c	t	r1+r2=r3			t'	Result	Microsequence	Notes
1	B5		c1	0	5	5	T		T=0,t2=1	U4 C1 I0 A2 A13	
2	B20	R	c2	1	L	L	R		R odd	C1 M4 I1 M7 M3 A13	
3	B23	R	c2	1	L	L	R		R odd	C1 M4 I1 M7 M3 A13	
4	B32	R	c6		5	G	R[odd]		G even	C1 A6 C2 M3 A13	
5	B36	G	c6		5	G	odd		t6?	C2 M3 I2 M3 A10	
6	B44	R	c2	1	1	1	R		R=3	C1 I1 A13	
7	B47		c6		5	G	3		t7=1?	U3 C2 A8	
8	B49	t7=1	c7	[1]					-p(z1!)	U2	no new goal
9	B58	R	c6		5	G	R	0	R>5	C1 M1 A13	
10	B62		c2	1	L	L	7		L=3	U3 C2 M5 I1 A3 A13	
11	B65	L	c2	1	L	L	7		L=3	C2 M5 I1 A3 A13	
12	B74		c6		5	G	7		G=1,2;t6?	U3 C2 I1 A13 A13	
13	B78	t6	c5		0	E	O		E=0	U1 C1 M6 A5 A6 C2 M4 A13	
14	B81	E	c5		0	E	O		E=0	C2 M4 A13	
15	B83	E	c5		0	E	O		E=9,t5=1	C2 M4 I2 A7 A13	
16	B85	t6	c5	1	0	9	O		t6=1	U1 C1 M1 I1 A2 A13	
17	B86	E	c5		0	E	O		[unclear]		unclear
18	B92	E	c3	0	A	A	E		E even t3=0!	C1 M4 I0 A4 A13	
19	B95	t3=0	c2	1	3	3	7	[0]	+,R=7!	U2 U4 C1 I1 A10 A12	
20	B98	E	c3	0	A	A	E		E even	C1 M4 I0 A4 A13	
21	B101	E	c5		O [even]	E	O		E=0,9	C2 M4 I2 A7 A13 A13	E even not used
22	B106	R	c2	1	9	9	R		R=9	C1 I1 A2 A13	

Table 2. PC occurrences

N	Item	Goal	£	t	r1+r2==r3				Result	Microsequence	Notes
23	B108	R	c2	1	9	9	R		R=9	CI 11 A2 A13	
24	Bill	R	c6		5	G	R	0	R>5	CI M1 A13	
25	B115	E	c3	0	A	A	E		E even	CI M4 10 A4 A13	
26	B116	E	c5		0 [even]	E	0		E=0;t5=0?	C2 M4 12 A10 A13	
27	B118	t5=0	c4		N	7	B	[0]	N<3	U2 U3 C2 M1 A13	
28	B123	E	c3	0	X	X	E		E=y	CI 10 A13	
29	B123.1		c5		0	y	0		0?	U3	
30	B128	E	c5		9	E	9		E=0	C2 12 A11 A13	A10 on E/9?
31	B129	.	c3	0	A	A	0		A=5	U3 C2 M5 10 A3 A11 A8 A3 A13	
32	B132	A	c3	0	A	A	0		.A=5	C2 M5 10 A3 A13	t't-4 now
33	B138	R	c2	1	L	L	R		R odd	CI M4 11 M7 M3 A13	
34	B139	R	c6		5	G	R	0	R>5	CI M1 A13	" ' '
35	B147.1	E	c3	0	X	X	E		E=y	CI 10 A13	
36	B148		c5		0	y	0		0?	U3 C2 M6 A5	
37	B152	E	c5		1	E	1		E=9;t5=:1	C2 11 A8 A13	why 11? 0=1?
38	B154.1	E	c5		2	E	2		E=9;t5==1	C2 11 A8 A13	
39	B156	E	c5		X	E	X		E=9; t5==1	C2 M4 12 A8 A13	
40	B157	E	c5		X	E	X		E=0	C2 M4 10 A13	
41	B160		c5		0	9	0		t5=1	U3 C2 M6 A5 A6 C2 M4 A7	but t5=1 known
42	B161.1		c5		0	9	0		t6=1	U4 CI M6 11 A2 A13	U4 because A5 on 0
43	B162		c6	1	5	G	7	0	G=1	U3 C2 11 A13	

Table 2 (continued)

N	Item	Goal	c	t	r1+r2=r3	t'	Result	Microsequence	Notes
44	B165		c3	0	A A 9		E=9-p	U3 C2 M5 I0 A3 (T2)	T2 because A-p => E=9-p
45	B167	A	c3	0	A A 9		E=9-p	C2 M5 I0 A3 (T2)	
46	B179	E=9	c3	0	A A [9]		-p	U2 U3 C2 M5 I0 A3 (T2)	
47	B182	A	c3	0	A A 9		t3=1?	C2 M5 A9 I2 A3	
48	B182.2		t3=1	1	3 3 7	[1]	-p(R←7!)	U2 U4 C1 I1 A10 A12	
49	B185		c1	0	5 5 0	1	+	U4 C1 I0 A2 A12	
50	B187	R	c2	1	9 9 R		R=9	C1 I1 A2 A13	
51	B190	L	c2	1	L L 9		L=4	C2 M5 I1 A3 A13	
52	B197		c6		5 G 9	0	G=3,4;t6?	U3 C2 I2 A13 A13	
53	B199.2	t6	c5		O E O		E=9;t5=1	U1 C1 M6 A5 A6 C2 M4 I2 A13	knows too much
54	B203	(t5)	c4		N 9 B		t5=1	U3 C2 M1 A6 C1 M1 A2 A13 U1 C1 M1 A2 A13	if goal= get t5
55	B203.1		c5	1	O E O		E=9	U3 C2 M4 I1 A7 A13	assume U3=>E
56	B206		c5	0	O E O		E=9	U3 C2 M4 I0 A13	
57	B207		c3	0	A A 0		A=5	U3 C2 M5 I0 A3 A11 A8 A13	
58	B210	A	c3		A A 0(?)		t3=1?	C2 M5 A9	
59	B211	t3=1	c2	1	L L R	[1]	L>5?	U2 U3 C2 M5 M1 I1 A3 A13	=> L>5, not L>5?
60	B212		c2	1	6 6 R	1	R=3	U4 C1 I1 A10 A13	
61	B220	E	c5		8 E 8		t5=2?	C2 A7	
62	B222	t5=2	c4		N 9 B	[2]	-p	U2 U3 C2 M1 A1	
63	B230	t5=2	c4		N 9 B	[2]	-p	U2 U3 C2 M1 A1	
64	B231	t5 (max)	c4		N 9 B		B=8;t5=1	U1 C1 M9 I2 M9 A2 A13	
65	B240		c5		O 9 O		t5=1	U2 C2 M6 A5 A6 M4 A7 A13	

Table 2 (continued)

N	Item	Goal	c	t	r1+r2=r3	t'	Result	Microsequence	Notes
66	B242		c5	1	0 9 0		0 free	U3 C2 M6 I1 A5 A13	
67	B255		c4		N 7 B	1	N>2,t4?	U3 C3 M1 I2 A13 A13	=>N>1,N>2;t4?
68	B257	t4	c3	0	A A 9		A-p	U1 C1 M4 I0 A4 A10	
69	B258	A	c3	0	A A 9		A-p	A6 C2 M5 I0 A3	continuation of 68
70	B260	A	c3		A A 9		A=4,t3=1?	C2 M5 A9 I2 A3 A13	
71	B261	t3=1	c2	1	L L R	[1]	L>5?	U2 U3 C2 M1 M4 I1 A3 A13	=>L>5, not L>5?
72	B262.2		c2	1	6 6 R	[1]	R=3	U4 C1 I1 A2 A10 A13	
73	B264	R	c6		5 6 R	0	R>5	C1 M1 A13	
74	B268.1		c2	1	7 7 R	[1]	R=5	U4 C1 I1 A2 A10 A13	
75	B271		c2	1	8 8 R	[1]	R=7	U4 C1 I1 A2 A10 A13	
76	B276		c3	1	A A E	[9]	A=4,E=9	U3 C2 M5 I1 A3 A10 A13	=>A=4,+
77	B278		c5	1	x 9 x [free] [free]		+	U4 C1 M4 I1 A10	
78	B279	N	c4	0(?)N	7 B		N free	C2 M1 A13	
79	B290		c4	0(?)x	7 B		B=y,t5=1	U4 C1 M8 A10	recollect t5=1
80	B290.1	N	c4	0(?)N	7 y	1	N>2	C2 M1 A13	
81	B295		c4	0	6 7 B	1	B=3	U4 C1 I0 A10 A13	
82	B299	B	c4	0	6 7 B	1	B=3	C1 I0 A10 A13	
83	B304		c1	0	5 5 0	1	+	U4 C1 I0 A10	
84	B306		c2	1	8 8 7	1	+	U4 C1 I1 A10	
85	B309		c3	1	4 4 9	0	+	U4 C1 I1 A10	
86	B311		c4	0	6 7 3	1	+	U4 C1 I0 A10	
87	B313		c5	1	2 9 2	1	+	U4 C1 I1 A10	
88	B315		c6	1	5 1 7	0	+	U4 C1 I1 A10	

Table 2 (continued)

PC: U → C → M → I → A

Determine unknown (U)

U1 get t' → set u=s

U2 get v=d → set v=d; get new goal

U3 u undefined and 1 above → set u=1

U4 u undefined and no 1 above → set u=s

Set to compute (C)

C1 u below → set to add; set a=r2

C2 u above → set to subtract; set a=r3

Modify operand (M) (comparison of operand and a)

M1 d vs 1 → convert 1 to inequality:

1 above → >0
1=r3 and t'=0 → <10
1=r3 and t'=1 → >9
1=r3 and t'=2 → >19
1=r3 and t'? → undetermined

M2 1 vs inequality → convert 1 to inequality

M3 d vs even/odd → convert d to even/odd

M4 1 vs same 1 → convert 1 to 1:1 (i.e., 1 unit of 1)

M5 operand = u and on same side → add 1 to n:u

M6 operand = u and on opposite side → subtract 1 from n:u

M7 d vs 2:1 → convert 2:1 to even

M8 d vs specified unknown digit → convert d to specified unknown digit

M9 v and max and add → GN(v)[top]
(others from (max,min)x(add,subtract) not used)

Figure 17: Micro production system for PC.

Carry into (I)

I0 t=0 → set operand = 0

I1 t=1 → set operand = 1

I2 t? → get with t=1

Analyze answer (A)

A1 a>9 and u above → -p

A2 a>9 and u below → decompose a into digit and carry

A3 n:u and n>0 → divide a by n

A4 a=2:1 → convert a to even

A5 0:u → u undetermined

A6 a undetermined and another v → change u to v

A7 a=0, subtract d → a= complement d

A8 a-p(too small) → t'=1?

A9 a-p → set t?

A10 u is also e → compare a to e

A11 a=d = operand → -p

A12 u is get v=d → compare a to d

A13 → u=a (including a undetermined)

Notation

u = unknown

a = the developing answer (accumulator)

s = sum = r3+10t'

t = carry into column

t' = carry out of column

Figure 17 (continued).

then u will already be determined. The second stage (C) sets up the system for how to calculate. Either operands are to be added into the accumulator (called a), or operands are to be subtracted. In the latter case r3 is set as the initial value of a. Then follows as the third stage a series of attempts to take the information in the column and arithmetically combine it with the developing answer. However, since the operands can be other kinds of information than digits a set of reactions is required, which depend on the nature of the operand and the nature of the developing answer. These reactions are contained in M. The fourth stage (I) consists of determining whether there is a carry into the column (t), and taking it into account. To do so may involve additional operand modifications. The last stage (A) analyses the answer. It may require some additional operations on the accumulated answer, a, depending on the nature of the unknown, u.

Table 2 gives not only the occurrences of PC along with the inputs and the result, but also the sequence of micro-actions that supposedly would generate the result if the production system of Figure 17 were in operation. A couple of examples will make clear how the system works.

The simplest case is shown by the first item B5. No unknown is specified, and all letters above the line have digits for values (both D=5); hence U4 is evoked, which makes u = T. Since u is below the line, C1 is evoked which sets to add. Addition of the two 5's proceeds without need for modification, and I0 obtains no carry, leaving a with the value of 10. A2 is evoked since a is greater than 9; thus the answer is decomposed into a digit of 0 and a carry of 1. Finally, A9 makes the assignment of a to the unknown; thus getting T=0, and t2=1.

A more complex case is shown in the third occurrence, B23. Since getting R is already the goal, no U-production is evoked, and C1 sets to add with a initially equal L. Since the next operand is L, M4 is evoked, which then allows a to become 2:L. Next, I1 fetches the carry of 1 and the attempt to add a digit to 2:L evokes M7. This says, in essence, that there is no way to add apples to oranges and so abstracts the 2:L to even. The attempt to add 1 to even evokes M3 which forces the conversion of 1 to odd. This leads to odd plus even is odd for the value of a, which A9 then assigns to R. (Thus the system is capable of both digit arithmetic and even/odd arithmetic.)

We asserted earlier that we would not produce a formal model of immediate memory and processing, but would proceed informally. The scheme of Figure 17 is in this spirit. There are several ways in which it falls short of a complete model, most notably in not being completely specific on what evokes each production, on how the various operands are actually picked up, and on the handling of the answer (a) when it consists of both a digit and the carry out (t'). Even so,it does indicate one kind of system that can yield the variety of responses recorded for PC. It also shows that there is no gross inconsistency in the behavior of PC from occasion to occasion.

One regularity is apparent. The carry is attended to only after the other operands. There are eleven instances in the data in which some evidence of language is available on the order of addition (B20, B23, B36, B65, B182, B271, B276, B306, B309, B313, B315) and in all of these the carry is dealt with afterward. Furthermore, no evidence of any kind exists for the carry being considered at any other position (such as picking up the first operand, adding

in the carry and then the dealing with the second operand). This issue bears on another one; namely, whether the carry is in fact taken into account at all on some occasions. There are three occasions (B20, B32-36, B78-81-83), all at the beginning, where the carry is not attended to at first. However, it is always discovered eventually, and throughout the remainder of the protocol the best fitting assumption seems to be that the carry will always be taken into account. Very often, of course, when the carry is 0 or when it is undefined, there is nothing explicit in the protocol to indicate that the carry has been noted.

The system of Figure 17 also points to a few rough spots. Among the more prominent difficulties is the handling of c5; in particular, whether E=9 is obtained by subtraction, by complementation, or by some form of recognition. As to the last, in a highly overlearned task, such as arithmetic operations on simple numbers, the possibility always exists that answers are obtained by recognition rather than by applying arithmetic operations. Sometimes the language evidence is compelling that operations occurred, but often it is not. Thus in the present case, transforming O+E=O into E=0, the subject may simply recognize that 9+1 provides a solution, and then afterwards connect the 1 with t5. Subtraction seems less likely, especially since no questions about whether $t'=1$ appear to get raised (as the evocation of A6 would imply). We chose to encode with complementation, both to avoid the difficulties of $\bar{t}'$ and because item 61 (B220) seems consistent with taking the complement of 8 to get 2, overlooking the fact that it is O that is 8 and not E.

In addition to the bias of the present scheme toward doing operations rather than recognizing the answers, it does not reflect any short term

learning, some of which must be going on. A good example of this failure is item 53 (B199.2) where the subject has been through c5 enough times (15) not to have to go through the extended sequence of micro-actions shown. On the other hand, it should be noted that there is little evidence from other sources (and none from the present behavior) that the subject does not go through the motions of the arithmetic operations even though the sequence has become quite familiar and the outcome is expected (in the sense of being recognized as familiar, not of being produced in immediate memory prior to finishing the sequence).

There are a few cases in which the outcome of the microsequence is somewhat at variance with the outcome used in the rest of the analysis. In item 59 (B211) and again in 71 (B261) the scheme produces L>5 rather than L>5?. This might have some effect on the main analysis, since the ? evokes G1 which sets the goal of obtaining L. (The apparently analogous case at B58 where R>5 is obtained, already has the goal of getting R established.) In item 67 (B255) the microsequence puts out both N>1 and N>2, whereas the main analysis stated only N>2; however, this has no effect on the analysis. Likewise, in item 76 (B276) the microsequence includes the comparison via A7, which is only implicitly included in the main analysis; again, no consequences flow from this difference.

GN(27), AV(27). These two processes (given in Tables 3 and 4) are discussed together, since AV(v) can be viewed as:

AV(v): GN(v)=> d; AV(v,d)=> v←d

AV(v,d) is essentially the operation that occurs in a standard paired associates learning task. It may differ somewhat, since recall can be via either v or d. and this may modify the way the subject stores the information. Also, one of

N	Item	Prod.	v	Set	Result	Notes
1	B26	S4	R	odd	1,3,5,7,9	
2	B59	S4	R	odd,>5	7,9	
3	B96	S4	R	odd,>5	7,9	repeat B59
4	B98.1	S4	E	even	2,4,6,8	
5	B140	S4	R	odd,>5	7,9	repeat B59
6	B150	tS3	0	ds		)
7	B154	tS3	0		2	} one generation
8	B155	tS3	0		interrupted	
9	B159	S4	E	0,9;-0 .	9	
10	B178	S4	R	7,9;-7	9	
11	B182.4	S4	R	odd,>5	7,9	
12	B204	S4	t5	0,1;-1	0	
13	B211.2	S4	L	>5	6	
14	B215	S4	R	7,9	7,9	possible R odd,
15	B223	S4	E	0,9	0,9	
16	B228	S4	E	0,9	0,9	repeat B223
17	B235	S4	E	0	0	
18	B239	S4	E	0,9;-0	0	
19	B242.1	S4	0	free(ds)	1	
20	B248	S4	0	free(fds)	j	
21	B262.1	S4	L	>5	6	)
22	B268	S4	L		7	S one generation
23	B270	S4	L		8	

Table 3. GN occurrences

N	Item	Prod.	v	Set	Result	Notes
24	B285	S4	N	fds	3,6	
25	B291	S4		fds	3,6	repeat B285
26	B298	S4	N	fds	2	
27	B301/3	S4	O	fds	2	

Table 3 (continued)

N	Item	Prod.	v	Set	Result	Notes
1	B42	S3	L	ds	L←1	
2	B61	↑S4	R	7,9	R←7	
3	B105	S3	L	7,9;-7	L←9	
4	B120	S3	A	ds	A←x	
5	B125	S3	O	ds	O←9	
6	B143	↑S4	R	7,9	R←7,R←9	
7	B147	S3	A	ds	A←x	repeat B120
8	B150	↑S3	O	1	O←1	
9	B154	↑S3	O	2	O←2	
10	B155/1	?	O	?	O←x	
11	B159	S4	E	9	E←9	
12	B178	S4	R	9	R←9	
13	B182.4	S4	R	7,9;-7	R←9	
14	B186	S3	L	9	L←9	repeat B105
15	B189	S3	R	9	R←9	
16	B204	S4	t5	0	t5←0	only t
17	B211.2	S4	L	6	L←6	
18	B219	S3	O	-0,-9	O←8	
19	B239	S4	E	9	E←9	
20	B248	S4	O	ø	O←x	
21	B262.1	S4	L	6	L←6	
22	B268	S4	L	7	L←7	
23	B270	S4	L	8	L←8	

Table 4. AV occurrences

N	Item	Prod.	v	Set	Result	Notes
24	B288	S4	N	3,6	unclear	
25	B289	R1	N	3,6	N←x	
26	B264	S4	N	3,6	N←6	
27	B301/3	S4	O	2	O←2	

Table 4 (continued)

the components (v) remains in view at all times, even though embedded in a display. Actually, 13 of the 27 occurrences of AV are essentially AV(v,d), where a single value has been delivered externally, either by GN (in S4) or because S3 goes slowly enough for us to see the generation going on (B150-B154).

There are 5 cases where AV assigns a symbolic value; e.g., O←x at B248. These symbols simply stand for "a value" and are not more complex than digits. There is certainly no difficulty assuming that AV(v,x) can associate such symbols. However, it does imply additional structure to that shown above. In particular, some processes must evoke by-passing of GN and going into the abstract mode. There is not enough data in the present situation to pin the mechanism down; indeed, it is clear that more than one mechanism is involved. B120 and its repetition, B147, involve a kind of planning; whereas B248 is a way to indicate that O is taken care of, even though its value is unknown until the end.

If we add in the remaining 9 cases of AV, which contain a buried GN, we get a total of 36 cases of GN. The main form of this process, accounting for 24 cases, is to start at the low member of the set and generate values in ascending order. Included in this total are three cases in which the set has one or no elements. Also included is B98.1, which generates E even, starting from 2 rather than from 0. This is undoubtedly correct (only programmers start counting from 0), even though for the problem at hand 0 is the correct starting place. (The consequences of this will be apparent later.)

A somewhat different mechanism seems to be involved in 6 other cases, in which there is a two element set of which one member has been determined to be not possible just prior to the evocation of GN. The proximity to

this new information implies that the basic set of values has not yet been updated; indeed, one can view the CN as the performance of this updating. This could be accomplished, of course, by a generate and test, using GN and TD. However, with only a two element set there is the strong possibility of a mechanism that says "pick the other one." The coexistence of the two mechanisms is possible, unique clues for evoking this one being clearly present. In general, an organism with a small immediate memory may be expected to handle small sets quite differently than large ones, which must necessarily involve serial generation.

This leaves us with 6 cases. Three of these are unimportant, involving either incomplete GN's for which no information is available (B155, B288), or an already discussed unsatisfactory situation, which is probably digit centered rather than letter centered (B235). In the final three there is evidence both that a set of values exist and that the top one in the range is selected, rather than the bottom one (they involve generation of only a single value). In the two cases where O←9 (B125) and O←8 (B219) there are complexities going on that we can sense, but have not captured. For instance, by assigning O to be 9, the dilemma for E=0,9 is resolved. Likewise, O←8 is probably confused with E=8, since it evokes t5=2 (or is determined by t5=2, a shift of interpretation we explored, but discarded, at an earlier stage of the analysis). The final case of generating from the top occurs at B294 and leads to the selection of N←6 instead of N←3. This follows upon an extensive comparison of the consequences of each value, which apparently ended indecisively. The failure here is our inability to discover the additional considerations that went into the decision.

TD(31). The central part of TD(l,d) consists of two attempts to associate:

Is d associated to anything?

If so, is it l?

Is l associated to anything?

If so, does it contain d?

For the present subject, who appears to work always from l to d, presumably if any letter is associated with d, it need not be tested to see if it is l. That is, we find no instances in Table 5 in which TD was evoked with a variable that already had a value. In the other direction, however, it is possible for l to be constrained, hence for a given d to lie outside the admissible set for l. This happens in three cases, two for R>5 and one for E even. Whether these two accesses are done in the order shown, the inverse order, or as a single access on a compound stimulus (l,d) is unclear. It may vary with the circumstances, being l first and then d, when a new assignment is being proposed, but d first during a generation of digits. Such variation, of course, requires either that TD be two different processes or that there be enough executive structure in TD to permit adaptation to circumstances. Additional executive structure is indicated, at least for B229, where two values of E are discarded, one because T=0, the other because R←9.

The only other noteworthy occurrence of TD is at B295.1 where the subject becomes aware that if he permits the conclusion that B=3, there will be no other digits available (fds=∅) even though a letter (O) is unassigned. Clearly this inference does not come from the two associations listed above, but

N	Item	Prod.	l	d	Result	Notes
1	B7	T1	T	0	+	
2	B28	T1	R	1,3,5,7,9	R=5-p(D←5!)	
3	B45	T1	R	3	+	
4	B60	T1	R	7,9	+	
5	B63	T1	R	3	L=3-p(R=3-p!)	
6	B66	T1	L	3	+	
7	B84	T1	E	9	+	
8	B99	T1	E	2,4,6,8	+	
9	B103	T1	E	9	E=9-p(E even!)	
10	B106.1	T1	R	9	R=9-p(L←9!)	
11	B131	T1	A	5	A=5-p(D←5!)	
12	B135	T1	A	5	A=5-p(D←5!)	repeat
13	B163	T1	G	1	+	
14	B188	T1	R	9	R=9-p(L←9!)	repeat
15	B190/2	T1	L	4	+	
16	B200	T1	E	9	E=9-p(R←9!)	
17	B203.2	T1	E	9	E=9-p(R←9!)	
18	B209	T1	A	5	A=5-p(D←5!)	
19	B213	T1	R	3	R=3-p(R>5!)	
20	B224	T1	E	9	E=9-p(R←9!)	
21	B229	T1	E	0,9	E=0,9-p(T=0!,R←9!)	
22	B233	T1	B	8	B=8-p(O←8!)	
23	B236	T1	E	0	E=0-p(T=0!)	

Table 5. TD occurrences

N	Item	Prod.	l	d	Result	Notes
24	B243	T1	O	1	O=1-p(G=1!)	
25	B262.3	T1	R	3	R=3-p(R>5!)	
26	B268.2	T1	R	5	R=5-p(D←5!)	
27	B273	T1	R	7	+	
28	B293	T1	N	3,6	+	
29	B295.1	T1	B	3	B=3-p(fds=ø!)	
30	B300	T1	B	3	+	
31	B301	T1	O	2	+	

Table 5 (continued)

requires other processing. If our model had some way of handling "noticing," this event could be handled differently (and also B202-203, which raises some of the same issues).

FC(64). The defined input to FC is the variable whose column is being sought. Thus, we require additional specification of mechanism when there is more than one column that involves the given variable. Table 6 shows the alternative outputs in the column labeled 'Others." For a carry both the column that determines it and the adjacent column that uses it are listed. Among the 64 cases there are 10 that are repeats of other occurrences of FC (noted in the last column) and another 8 that have uniquely determined columns: for these there is no further concern. For the others, the key feature seems to be whether the processing of that variable on the current column has occurred or not. This is indicated on the table by a yes or no in the column labeled c done?, which is left blank if the subject is not located on any particular column. Suppose we assume the rules:

1. If current column is unprocessed for the variable, always select current column;

2. If current column is already processed for variable, do not select current column.

The first rule accounts for 6 cases. The second rule accounts for 30 cases, in that it reduces the set from which selection must occur to either one or no elements. These cases are labeled rule 1 and rule 2, respectively.

We are left with 10 cases in which one column was selected from a set of two or three eligible columns. Almost all (8) involve the selection of a

N	Item	Prod.	V	£	c done?	Total Result	Set Others	Notes
1	B5	SI	D			cl	c6	cl most constrained
2	BIO	SI	T	cl	yes	4	cl	cl out by rule 2
3	B12	SI	D	cl	yes	c6	cl	cl out by rule 2
4	B20	S2	R			c2	c4,c6	c2 most constrained
5	B32	S2	R	c2	yes	c6	c2,c4	c2 out by rule 2, c6 most constrained,
6	B47	SI	R	c2	yes	c6	c2,c4	c2 out by rule 2, c6 most constrained
7	B49	S2	t7	c6	yes	c7	c6	c6 out by rule 2
8	B58	S2	R	c6	no	c6	c2,c4	c6 by rule 1
9	B62	SI	R	c6	yes	c2	c4,c6	c6 out by rule 2, c2 most constrained
10	B72	SI	L	c2	yes	4	c2	c2 out by rule 2
11	B74	SI	R	c2	yes	c6	c2,c4	c2 out by rule 2, c6 most constrained
12	B78	S2	t6	c6	yes .	c5	c6	c6 out by rule 2
13	B85	S2	t6	c5	no	c5	c6	c5 by rule 1
14	B86	S2	E	c5	no	c5	c3	repeat
15	B92	S2	E	c5	yes(?)	c3	c5	c5 out by rule 2(?)
16	B95	S2	t3	c3	yes	c2	c3	c3 out by rule 2
17	B98	S2	E			c3	c5	repeat
18	B101	S2	E	c3	yes	c5	c3	c3 out by rule 2
19	Bill	S2	R			c6	c2,c4	repeat
20	B115	S2	E			c3	c5	repeat
21	B116	S2	E	c3	yes	c5	c3	repeat

Table 6. FC occurrences

N	Item	Prod.	v	c	c done?	Total Set Result	Others	Notes
22	B118	S2	c5	c5	yes	c4	c5	c5 out by rule 2
23	B123.1	S1	E	c3	yes	c5	c3	c3 out by rule 2
24	B129	S1	E	c5	yes	c3	c5	c5 out by rule 2
25	B138	S2	R			c2	c4,c5	repeat
26	B139	S2	R	c2	yes	c6	c2,c4	repeat
27	B148	S1	E	c3	yes	c5	c3	repeat
28	B160	S1	E	c5	no	c5	c3	c5 by rule 1
29	B161.1	S1	t5	c5	no	c5	c4	c5 by rule 1
30	B162	S1	t6	c5	yes	c6	c5	c5 out by rule 2
31	B164/1	S1	G	c6	yes	ϕ	c6	c6 out by rule 2
32	B165	S1	E	c5	yes	c3	c5	c5 out by rule 2
33	B179	S2	E			c3	c5	repeat
34	B182.2	S2	t3	c3	yes	c2	c3	c3 out by rule 2
35	B191	S1	L	c2	yes	ϕ	c2	c2 out by rule 2
36	B197	S1	R	c2	yes	c6	c2,c4	c2 out by rule 2, c6 most constrained
37	B199.2	S2	t6	c6	yes	c5	c6	c6 out by rule 2
38	B203	S1	R			c4	c2,c6	Note c2, not c4, most constrained
39	B203.1	S1	t5	c4	yes	c5	c4	c4 out by rule 2
40	B206	S1	t5	c5	no	c5	c4	c5 by rule 1
41	B207	S1	E	c5	yes	c3	c5	c5 out by rule 2
42	B211	S2	t3	c3	yes	c2	c3	c3 out by rule 2
43	B212	S1	L	c2	no	c2	ϕ	unique
44	B222	S2	t5	c5	yes	c4	c5	c5 out by rule 2

Table 6 (continued)

N	Item	Prod.	v	c	c done	Total Set Result	Total Set Others	Notes
45	B230	S2	t5	c5	yes	c4	c5	c5 out by rule 2
46	B240	S1	E			c5	c3	Neither most constrained
47	B242	S1	t5	c5	no	c5	c4	c5 by rule 1
48	B255	S1	t5	c5	yes(?)	c4	c5	c5 out by rule 2 (?)
49	B257	S2	t4	c4	yes	c3	c4	c4 out by rule 2
50	B261	S2	t3	c3	yes	c2	c3	c3 out by rule 2
51	B262.2	S1	L	c2	no	c2	ϕ	unique
52	B264	S2	R	c2		c6	c2,c4	c2 out by rule 2 c6 most constrained
53	B268.1	S1	L	c2	no	c2	ϕ	unique
54	B271	S1	L	c2	no	c2	ϕ	unique
55	B276	S1	t3	c2	yes	c3	c2	c2 out by rule 2
56	B278	S1	A	c3	yes	ϕ	c3	c3 out by rule 2
57	B278/2	S1	E	c3	yes	c5	c3	c3 out by rule 2
58	B279	S2	N			c4	ϕ	unique
59	B290	S1	N	c4	no	c4	ϕ	unique
60	B290.1	S2	N	c4	no	c4	ϕ	unique
61	B295	S1	N	c4	no	c4	ϕ	unique
62	B299	S2	B	c4	no	c4	ϕ	repeat
63	B301	S1	B	c4	yes	ϕ	c4	c4 out by rule 2
64	B302	S1	O	c5	yes	ϕ	c5	c5 out by rule 2

Table 6 (continued)

column for R; one involves the initial selection of a column for D, and the remaining one the selection of a column for E after the final decision has been made to assign E the value 9. Any variation on a concept of "select the most constrained" will account for all these cases but B203 and B240. The subject would select either c2 or c6 for R in preference to column 4, and c1 for D in preference to c6. One must be careful in evoking such a mechanism, however, since it can easily imply considerable computation and comparison of all columns before a selection is made. This clearly does not occur. For example, B12 makes it highly probable that the subject did not select the initial column for D (at B5) by a deliberate comparison of c1 and c6. Of the two cases not explained by maximum constraint, one (B203) appears to involve a genuine anomaly (already discussed) in which the concern for R leads the subject to evoke S1 on R←9, rather than take off on E←9. In the other case (B240) we have no clues why c5 should have been selected over c4 after E←9 (with the subject not located at any column).

FA(19), FP(4). These two processes(presented in Tables 7 and 8) are grouped together because the essential component in both is the recall of past behavior. In both FP and the FA in production T2 the call is for some past information. But even in the use of FA in S3, which on the surface simply calls for a relationship that determines the input variable, the result is never a new relationship, but one which has been used already. Thus, in B42, which concerns the assignment of L to get R, it seems implausible to think of FA as having the choice between three columns for R (as in FC). Rather, return is to c2 which was used to derive that R is odd.

N	Item	Prod.	v	Result	Notes
1	B42	S3	R	c2,L	no alternatives
2	B56	T2	t7=1	R=3	no alternatives
3	B105	S3	R	c2,L	repeat B42
4	B106.2	T2	R=9	L←9	no alternatives
5	B120	S3	E	c3,A	c5
6	B125	S3	E	c5,O	parallel B120, c3 (but already used)
7	B136	T2	A=5	E=0	O←9 but hypothetical as the line disappears
8	B147	S3	E	c3,A	repeat B120
9	B149	S3	E	c5,O	repeat B125
10	B186	S3	R	c2,L	repeat B125
11	B189	S3	R	c2,L	repeat B186
12	B203.3	T2	E=9	t5=1	no alternative (not R←9-p)
13	B219	S3	E	c5,O	no alternatives (once c5 is constrained at B217)
14	B229.1	T2	E=0,1	t5=0,1	no alternatives (not O←8)
15	B234.1	T2	B=8	t5>1	no alternatives
16	B238	T2	E	R←9	O←8 possibly?
17	B243.1	T2	O=1	O free	no alternatives
18	B264.1	T2	R=3	L←6	no alternatives
19	B269	T2	R=5	L←7	no alternatives

Table 7. FA occurrences

N	Item	Prod.	Action
1	B95.1/2	R2	repeat S4 on R←7
2	B182.4	R2	repeat S4 on R←7
3	B214	R2	repeat S4 on R>5
4	B263	R2	repeat S2 on R>5

Table 8. FP occurrences

With this view there is very little to say about the mechanisms of FA and FP without a more detailed model of memory. Table 7 for FA shows that for 9 cases there are no alternatives to the output provided, and that 5 others are repetitions of prior sequences. This leaves 5 cases which are worth some discussion. Three of these concern whether O←d could have been evoked by FA. These assignments do stand in the PBG prior to the evocation of the production in question (always T2). In fact, no consequences follow from the assignment of values to O, but incorporation of this would seem to require a memory that keeps track of additional connections other than just the tree ordering. The other two cases (B120, B125) belong together as one, since B125 is a parallel version of B120, with c5 and O substituted for c3 and A. This in itself reveals that at B120 there was a choice between c3 and c5 as a way of determining E. However, we have no proposed mechanism for making this initial selection.

We do deal with two different forms of memory in these processes: with path memory for FA, and production occurrence memory for FP. These are probably not distinct, but are all interwoven in the memory of past behavior. However, as already noted, there is little additional light we can shed on this.

GNC(2), FL(4). Both of these processes are represented by so few instances that essentially nothing can be said about their internal mechanisms. They both exist in response to needs for sufficiency -- e.g., it is not possible to add up a sequence of columns serially (B304-B316) without sequencing through the columns.

The only instance of FL that offers food for thought is the extended attempt at B14-B18 to select a letter for processing. The subject

N	Item	Prod.	Input	Results
1	B185	?		c1
2	B303	S5	c3	c1 to c6

Figure 9. GNC occurrences

N	Item	Prod.	Input	Results
1	B14	G4	1s: all - 1,D	R
2	B278	G4	1s: N,B,O	N
3	B301	G4	1s: O	O
4	B302	G4	1s: ϕ	ϕ

Figure 10. FL occurrences

clearly is considering letters and their multiple occurrences. Thus a mechanism that chooses the unprocessed letter with the maximum number of occurrences will be as good an approximation as one can get (even though it may differ considerable from the actual set of considerations that the subject goes through).

Discussion

We have finished the detailed analysis. It remains only to pull a few threads together, and view the effort somewhat more broadly.

Overview of S3's behavior. Figure 11 (page 44) provides a way of glimpsing the total problem solving effort in which S3 engaged. This can almost be told as the play of a game. The opening development is straightforward. The subject takes each piece of new information and feeds it back into the problem to harvest further information (essentially production S1). This straightforward approach founders in the attempt to infer that G is even. Then follows the first attempts at assignments. These are done systematically. The first one, L←1, fails almost immediately, but leads to the discovery that R is 7 or 9. The second assignment, R←7, leads to a moderately advanced position, and ends the "opening game."

Then S3 enters a long phase (lines 11 to 29) in which he worries whether E is 0 or 9. This, of course, is the key to the problem for any approach that relies on "reasoning" through to the solution. It appears that the crucial feature is the subject's failure to recall that T=0 is already used. However, three features of the subject's behavior contribute to his difficulties.

1. The failure to note that T=0 so that E=0-p
2. The inability to produce a solid analysis of c5 so that E=0 or E=9 is a clear dichotomy; he continually iterates on the analysis.
3. The failure to see that t3=1 is possible and so to lay to rest the contradictions that keep arising from E=9.

As to the first, the production system asserts that TD shall be applied whenever a new value is obtained. Why did it fail? We can pinpoint the places where E=0 is produced, and where, therefore, TD perhaps should have been evoked:

B78, B81, B98.1, B103, B116, B128, B157, B206, B223. Right after B223 (at B225) the experimenter could contain himself no longer and asserted that 'You've used the zero, too.''. From that point on the fact was established.

Of these nine instances, five (B103, B116, B128, B157, B223) can be eliminated from concern, since TD is not evoked on a familiar relationship.' This obvious bit of efficiency clearly unstabilizes the problem solving processes. Slips made at critical junctures may never be corrected. This is reminiscent of the "Einstellung" effect, [10], in which subjects, once a method has proved successful, find difficulty in calling it into question when the situation changes. Here it is.a test for digit admissibility that drops out, whereas in Luchin's task the method selection process drops out. Nevertheless, both produce a cul-de-sac through the effects of familiarity.

Of the remaining four instances, two (B78, B81) occur where PC, being unclear, is repeated. According to the priority ordering, production R1 (repeat) is evoked in preference to T1 (TD) (see Fig. 15). Unfortunately, this evidence is hardly compelling, since R1 and T1 only compete 2 times solidly (namely, these two at B78 and B81) and two times where there is doubt as to whether T1 should have been evoked or not (B156.1 and B288.1). Furthermore, R1 and T1 are adjacent in the priority order, so that there is no indirect support for the contention of the model that TD would not be evoked if PC was unclear. In any event, by the time PC produced a result that did not evoke a repeat the result was not E=0 but E=9.

This leaves B98.1 and B206. The first, B98.1, involves the generation of E even. This has been taken to start at 2 rather than 0 and hence the TD at B99

does not get applied to 0. Furthermore, although TD is used a moment later to reject E=9, E even is used, rather than E=0. Hence, again the opportunity for getting TD asked of 0 slips by. This reconstruction of the subject's behavior is consistent with the total system and plausible (if a person is asked to give the even numbers he starts with 2, not with 0). However, there is no explicit evidence at B98.1 that the generation occurred or that it started at 0.

B206 is the first encounter with E=0 in the second phase of the problem solving where the subject is exploring the consequences of R=9. It might seem that the subject would treat E as a new item and evoke TD. However, the text makes clear that the information is not so treated, since the subject immediately re-evokes the impossibility of A=5, which is implied by E=0. This availability of the inference on c3 is also reflected in B158, where the subject in doing PC on c5 asserts that "E can't be zero." This might seem to indicate awareness of T=0, but the analysis makes clear that A=5-p is the source of this conclusion.

We have discussed in detail the failure of S3 to note E=0-p(T=0!). It seems that "if it had been otherwise" the subject would have solved the problem much more quickly. However, the other features of his behavior should not be overlooked. The second feature, his general inability to make a clean analysis of c5, shows clearly in the total PBG, but not so clearly from the production system. It is basically a property of PC, and beyond the level at which we have modeled it. One might think, perhaps, that additional information -- e.g., from c3 -- should help. But such information, although it can give clues for discovery and add confirmation, may be unable to produce surety, which is what the subject needs. His PC is already good enough to derive the results in question; namely, E=9, t5=1.

The third feature concerns the subject's inability until the sixth encounter with c3 finally to break free from the assumption that t3=0 (B94, B131, B168, B182, B208 and finally B257). This is one of the main traps offered by crypt-arithmetic puzzles -- ignoring the possibility of carries by identifying the result at a column with the named letter at that column. The trap is deeper than the E=0 issue, which depends only on whether the subject will ask the right question, TD(E,0). At B94 the subject states his conviction on t3, "because I know I'm not carrying 1." At B168 he directly faces the issue of A+A=9 without calling into question t3=0. At B182 he finally considers the need for t3=1 ("If that were 4 + 4 plus .."). However, this leads only to his rejecting R←7 and setting up to explore R←9. For the subject these are still the two alternatives for c2 and he does not become clear that R←9 will not solve his problems (i.e., make t3=1). Again, at B208 he considers getting t3=1 (B211: "If I could get L to be more than 5 ..") and rejects it after an analysis that apparently consists of looking only at L←6. Finally, starting at B257 he goes again to L>5, repeats the L←6 investigation, and only then breaks himself free to consider other values for L and R: "Now suppose these were real big numbers, not just little -- not 10, but way more than 10" (B265-267).

The three features combine to keep the subject oscillating around a point of moderate advancement, achieved early in the session. This is almost the total story of the "middle game." It can be seen clearly in the gross structure of the PBG: The long plateau from lines 9-33, punctuated with reviews of R←9, leading to starting over with R←9; and a second plateau (lines 35-46) much like the first in character although shorter. Finally, thanks to the experimenter, the issue

is finally resolved, E=9 is posited, and the "end game" proceeds very rapidly to a successful conclusion (lines 48-55).

The ability to record attempts externally plays some role in the subject's behavior. It permits him to start over without losing the other solution. When he drops back to the initial attempt (line 48) it is still available. In particular, G=1 is still recorded and does not have to be re-derived (it is never mentioned in the protocol during the final advance).

Comparison with earlier work. The present analysis grows out of a continuing effort to use information processing systems to model human problem solving. The two most relevant studies, mentioned already at the beginning of the paper, are the attempts to simulate human behavior in simple tasks in symbolic logic using GPS [17, 18, 19] and the analysis of a chess protocol [21], which is similar in spirit to the present analysis.

Taking the chess study first, the subject conducts a forward search from the existing position, and we were able to construct a PBG much as in the present analysis. The problem space was not handled as formally as in the present effort, but clearly the elements were positions and the operators were by and large legal moves. Those that were not were classes of moves defined functionally -- e.g., Q-move, defend-B. The PBG showed a striking regularity. The subject always returned to the base position before engaging in another long deep and little branching foray. Thus, the subject could be described as following an overall search strategy -- it was called progressive deepening -- which could be contrasted with the depth-first and breadth-first strategies widely used in problem solving programs. The PBG of the present subject shows no such global regularity,

and we have characterized his behavior simply as a production system without any overall search strategy. Besides the always present fact of individual differences, two features of the tasks may be involved in this difference. First, chess has a permanent memory of the base position always available, but no ability to record new positions externally. Thus, dropping back to the base position is returning to certainty. In our crypt-arithmetic task, advanced positions were recorded, and hence there was no need to return always to the base position. The current external position could operate as a base. For example, R←7, L=3, T=0, D←5 was used for very large periods. However, even here some additional non-written information was available -- e.g., R=7,9 in the state just mentioned.

The second difference between chess and our task that might account for the lack of global search strategy is familiarity with the task. Chess was a familiar game to our chess subject, whereas this was the first crypt-arithmetic task S3 has done. Thus S3 had only his unadapted reasoning procedures available for the task; the chess subject had had plenty of time to develop a style of analysis.

In the chess analysis we were able to develop what amounted to a set of productions to be applied at each node which produced the moves considered at that node. These productions were of the form:

features of position → function to be performed => moves

That is, the features of the board do not directly yield legal moves to be considered, but rather yield the function (defend, attack, etc.) to be performed. This function, in turn, is used to generate the legal moves that perform that

function. The situation is entirely analogous to the productions that yield PC, which, when applied to the particular column at hand, yields concrete assignments (either of form 1←d or 1=d). This view implies a problem space for chess whose operators are the function terms rather than the legal moves.

The production scheme developed for the chess analysis does not handle selection of the base move, nor modulation of the exploration. Some rules were adduced in the chess study for the latter, involving repeated re-examination of a path if successful, and working through the productions according to a priority list of *a priori* relevance if unsuccessful. Nothing in the present analysis corresponds to this. In part the chess problem solving takes place within the framework of a search strategy that gives meaning to factoring the problem into separate types of rules, whereas no corresponding factoring makes sense in the crypt-arithmetic task.

In summary, the chess analysis is generally supportive of the present analysis. However, there exists no theoretical frame large enough to explain adequately the differences in the two resulting schemes of problem solving. This is just what we should expect, of course. At least a modest number of analysed examples are needed before such a frame can become clear.

GPS presents a different problem of comparison. Its program organization is quite different, not being composed of a production system, but of a problem solving interpreter with a set of problem solving methods.* Initially, the

---

* GPS has had several basic organizations over the years, each of which is quite different from a production system [5, 13, 14].

techniques of the present paper -- the PBG and the production system -- were developed as preliminary schemes for data analysis -- as ways of organizing the data so a program, of type unspecified, could be inducted that would incorporate the regularities so exposed. However, it is now clear that the production system offers an alternative organization, which might be extended to a fully functioning program.

We indicated earlier how the behavior of GPS might be viewed in terms of the problem space: a system which follows the goal stack assumption and which, whenever an operator fails to be applicable, sets up the subgoal of modifying its input so that it can be applied. This, of course, covers only the central core of GPS and not all of the methods -- e.g., how to select a new goal to work on after one has been abandoned. Given this much correspondence, a set of productions could be developed that would carry this out. One of the main differences between such a system and existing versions of GPS would be the subgoal tree. GPS keeps a record of all the intermediate goals along a path from A to B: transform A to B; transform A' to B; transform A" to B, ... In the problem space version, although there must be some path memory, which we have never specified for S3, it would undoubtedly not be the goal stack. The system might simply keep an anchor point (e.g., the original position) and the current position; that is, the goal would remain "get B," always with the current position understood.

Without carrying out the reformulation of GPS and the analysis of the behavior in logic in terms of the new production system it is not possible to do more than note the general correspondence that is implied by the possibility

of this change of representation. The problem space will consist of logic expressions as elements and the legal logic rules as operators. The subjects introduce a large number of function terms (e.g., eliminate, decrease, reverse), which GPS currently treats as differences and uses to select relevant operators. In the light of the use of similar function terms in chess, these might show up as an expansion of the set of operators.* However, unlike chess and like S3 on crypt-arithmetic, the subject is new at the task, and cannot be expected to have the function terms so internalized that they seem to be the real operators.

As noted, Bartlett (2, pp. 49-63) considered crypt-arithmetic, seeing it as an example of evidence in disguise. Apparently, Bartlett gave the problem to several people, asking them to write down the steps they followed.** The eight written protocols that Bartlett reproduces are almost completely consonant with the present study. They show the same form of reasoning as the present case. Several are better, going directly from c1 to c5 and deducing that E=9; a few were worse, floundering in trial and error. The one exception generated correspondences of letters with digits according to global rules -- e.g., A=1,B=2, C=3,... .

Bartlett's discussion is also largely consonant with our account, although formulated only in generalities. He sees the subject taking a series of steps, each drawing out some information that has been disguised, thus filling the gap between the initial state of information and the final one that solves the problem. Bartlett's analysis and ours diverge in his division of processes into

---

* In [15] a PBG for logic is given along these lines.

** No procedural details are given; this is within the spirit of the work, which is exploratory and discursive.

three types: analytical, guessing and insightful. The first corresponds reasonably well to PC and the second to an AV sequence followed by PC. The third, insight, he describes as seeing the answer without going through the steps necessary to develop it. Generally his notion of insight is supported by examples of perceptual problems, where the person suddenly "sees" the answer with no ability to report intermediate stages. In crypt-arithmetic Bartlett focusses on the lack of evidence in his written protocols of intermediate steps; e.g., the subjects simply write down the inference on E from c5, with no indication why they were led to consider this. From our point of view, his data corresponds more to the PBG from the written record (Figure 12) than to the PBG from the verbal protocol (Figure 11). It is clearly incomplete, and cannot support the kind of argument he constructs upon it. Thus, we do not end up with our subject showing insight. Whether such a process is needed in addition to analysis and guessing remains inconclusive.

There is little profit going further afield for comparison with the picture of human problem solving presented by the present analysis. The general features of search, of the ability to do symbolic processing, of goals and subgoals, and of means-ends analysis are all here. Beyond that, other studies do not provide information at a detailed enough level to illuminate whether the production system is a good model for the organization of human problem solving system; and to what extent S3's behavior is either typical or noteworthy.

Observations on production systems. We have already remarked that the type of system used to represent the subject's behavior is a quite general form of process organization. A certain comfort can be drawn from this. Whatever model is eventually adopted for describing human problem solving must have the power of a Turing machine -- that is, be able to perform arbitrary calculations.

Some points about this are worth noting. The requirements for a Turing machine — for universal calculational ability — are really quite simple: the ability to write symbols in a memory, to find and read them out again, and to react differentially to them. For production systems this memory is the work space — that which the condition part of a production is contingent upon. Thus, the production system writes by putting information into the work space, and reads by evoking productions conditionally on the contents of the work space. There must be available a finite memory of such differential reactions, and these correspond to the finite number of different productions. In a Turing machine the system reads a symbol and reacts to it, not only writing a symbol but by going into a new state of its (finite) memory, since the entire set of productions is exposed each time.* The affixing of a ? in our system is a typical example of how an internally introduced symbol is used to effect a selection of a specific production on the next step. Similarly, the use of sequences of actions, rather than selecting afresh, each elementary action from the total production system is another way of achieving context dependence. If we were restricted to a single elementary action per production, then each would put in a tinique cue symbol in addition to the desired output, so that the appropriate next production could be uniquely selected.

The final requirement for general calculational ability is that the system have an unbounded memory capable of being addressed repeatedly. In a Turing machine this is accomplished by an infinite tape with operations for moving the tape left and right under the reading head (which defines the immediate access capability of the machine). In the present system the domain of the conditions

* In some productions systems used for practical purposes — e.g., syntax analyses an explicit link to the next production is provided, so that this property does not hold.

is the immediate memory, which clearly we do not wish to be infinitely extendable. Thus, our system as it stands does not fully satisfy the conditions for "general calculability."

As already noted, some sort of conflict resolution doctrine is required in a production system. We adopted a priority scheme, which is both simple and frequently used. The priority order became a free parameter of the system and its value was determined to best fit the data. The resulting order seems fragile, at least in part. Determination of a production's position in the order is sometimes based on only a few cases. An example occurred in trying to understand whether failure to evoke TD on E=0 was explainable by the production system or was a failure of the production system to describe the behavior adequately. The argument hung partly on whether R1 had higher priority than T1. Although the matrix yielded a positive answer to the question, it was with slight margin of confidence. Yet, much of the psychological import of the system is buried within the priority rules. It remains an open question whether the priority scheme is an appropriate decision structure. After all, no direct psychological case has been made for it. However, it could also be that the shortcoming (small N per cell) is inherent in any attempt to describe a system which has so much mechanism to be determined.

As we come to consider the production system as a possible theoretical form for describing human problem solving, the psychological significance of its various features becomes important. Perhaps the most fascinating is the way the system shreds the generators of behavior into a set of independent parts -- that bears a resemblance to a collection of S-R connections. The productions are,

in some sense, independent and "additive." That is, each may be added to the set of productions without concern for the others that are there. The behavior that arises from this is certainly not a sequence of independent actions. But the interaction comes entirely from the symbols the evoked productions put in the immediate memory. This situation may be contrasted with that of a flow diagram, where the structure of the system as well as the component processes determine the total behavior. Of course, this means only that production systems, as opposed to flow diagrams, provide a homogeneous way to encode the total information necessary for a process. We have already discussed how one can often pass from one to the other by introducing additional symbols which serve to link productions uniquely. Such a change in representation is none the less useful for being understandable. However, the independence of the component productions fails to the extent that the conflicts must be resolved by a rule, such as a priority ordering, that depends on a global property of the set of productions.

Each production, considered separately, has the form of a stimulus (the condition part) evoking a response (the action part). Thus, the entire system has the flavor of a network of stimulus-response bonds. What distinguishes this from a garden variety S-R system as considered in experimental psychology? First of all, it is highly mediational. That is, there are many internal cycles of stimulus to response to stimulus before any external response is made. The immediate memory plays a central role in this. Second, the action parts are not simple responses -- that is, not some invariable pattern in immediate memory. Rather, the response is a sequence of processes each of which produces a new

pattern in immediate memory. It is this secondary pattern which defines the next stimulus and selects the next response. Thus the immediate memory plays a second role, as well.

There are two concerns with these actions. First, each is a sequence of actions (or responses) rather than a single one. However, as already discussed, we may view this as a convenience in writing productions and we could modify the system so each production had a single action (which, however, must then produce at least two "symbols"). More important is the nature of these actions. If they are not representable as similar S-R systems, then there is a non-S-R link in the total chain. Given the schematic micro-production system developed for PC a further reduction might seem plausible, but the question is clearly still open.

A third difference between the production system and the standard S-R system lies in how the next action is selected. The current system has a match routine which responds to structured symbolic expressions. The usual concept of strengths of connection is absent. Likewise, there is strict selective attention to part of the total stimulus (the total immediate memory). Most important, the match routine uses variables. That is, it permits the passing of parameters from the working memory to the actions. Hence, the latter are functions, rather than simply self contained processes. The system mechanisms required to do this more complex matching and parameter passing must itself be explained in simple S-R terms. It is not enough to show it can be performed by some production system. This latter is like showing that an Algol compiler can be written in Algol; it does not settle the issue of whether a given device can

compile Algol. One might show, however, how a system of productions that was parameter free could perform the parameter passing. Even this would not settle the whole issue, since it would imply that the S-R mechanisms exhibited in our production system were being performed interpretively by another simpler S-R mechanism. In short, there may be requirements on the total system of productions which go beyond what a simple S-R system can produce.

We mention the possibility of viewing the production system as an S-R system, not because the interpretation is clear, but because the resemblance is striking enough to be worth exploring. When one looks at information processing theories represented as flow diagrams -- e.g., the concept formation system of Johnson [9] -- there appears to be no connection at all between an information processing model and an S-R model. When one looks at a production system, a relationship seems possible.

Turning from the general properties of production systems to the particular one developed here raises some additional issues. For one, the system is not complete. If turned loose on DONALD+GERALD, or almost any other cryptarithmetic task, it would simply fail to evoke an action at some point. From a data analysis point of view the ability to be incomplete and still useful for describing the data as a source of great strength. However, it prevents us from really verifying whether the system performs as we claim. In fact, knowing the vagaries of hand simulation, there are surely additional difficulties, which have been glossed over unawares in the present analysis. As already remarked, one major source of incompleteness is the lack of a sufficiently detailed model of immediate memory so that actions, such as PC, can be fully specified.

Another issue concerns the psychological significance and generality of our productions. If this model represents structurally what is going on, then the individual productions must be learned, transferred, etc. Are the same productions used in other tasks? Do the same productions show up in different people? Here it seems to me, we are in better shape than we have any right to expect. A number of productions are clearly of general utility: G1, G2, G3, T2, R1, R2. Most of the others could be so reframed. For example, S2 needs to have FC abstracted to "find a thing that involves v" and PC abstracted to "process that thing for information about v." The processes corresponding to FC and PC could now not be unit actions; they might have to be set up as goals, which would evoke yet other productions. Still, they would carry the kernel of organization of some behavior.

Whether these same productions occur in other subject's behavior is not within the confines of this study. Cursory work with a few other subjects does show a few major productions, such as S1 and S2. This is true only for subjects who attempt the problem in the same general way. For example, some subjects attempt to use sets of simultaneous equations, others to use global rules for generating correspondence between digits and letters. One would not expect the same productions in such cases (nor even production systems as the useful form of description).

These comments do not offer substantial demonstration of the generality of these productions over people and over tasks. Yet the situation is not without hope.

Summary. Despite discussion of the prospects of production systems developing into a better way to write information processing theories of human problem solving, this paper should still be viewed primarily as a detailed case study of a single problem solving episode, adding thereby one more bit of data to the study of human problem solving. We have been concerned with how difficult it is to carry out such analyses, and have developed a number of data analytical tools to help with the task. These operate in the context of an information processing theory of problem solving based on the concept of heuristic search in a problem space. These tools are:

1) The formalization of the problem space, including both the knowledge states and operators, so that it becomes easier to determine what changes of knowledge are going on.

2) The Problem Behavior Graph (PBG), which is a way of plotting the subject's search through the problem space.

3) The production system, which permits one to extract the regularities of behavior at a node of the PBG.

4) The determination of the priority ordering of the production system by minimizing the number of wrong selections of the production system.

5) The display of the performance of the production system in a graph that shows both the coverage and the errors against the addition of new productions.

Most of these steps are "simply" data analysis. No theory underlies them, in the sense that we expect statistical theory to underlie proposals for tests of significance or techniques of parameter estimation. They are not less useful for that. In an area -- protocol analysis -- which has few tools we need all we can get.

The generality of these tools remains to be seen. In the shorter paper

devoted to protocol analysis [15], which is largely based on the present work, we did present PBGs from several different tasks: logic, chess, crypt-arithmetic, and the missionaries and cannibals puzzle. However, further steps involving production systems have not been carried through on any body of data as substantial as the one represented here.

Finally, although of necessity concerned with methodology, the main contribution of this paper is the information it provides about a problem solving attempt. How well this particular analysis will hold up remains to be seen. Like the chess analysis it is bedeviled by being only a single instance. Also, like it, the analysis has not proceeded to working programs. Consequently, one of the main security devices of information processing theories, simulation, is absent. Still, a substantial amount of information has been extracted from the data and brought to bear on inducting the processing structure of the subject. The web of cross dependency is sufficiently great that a critic tampers with an isolated bit of the protocol at his peril. Any attempt to "patch up" the explanation at one point runs the risk of introducing new errors at other points.

There are very few examples of analyses of problem solving behavior in the literature that provides enough structure to support conjectures of how the behavior is organized. A comparison of this analysis with the material on crypt-arithmetic in Bartlett's book on Thinking [2] will emphasize the point. Even in our analysis the evidence often has turned out to be much thinner than we would like. The field needs even larger bodies of data, processed even more finely than the present one.

References

1. Backus, J. W., "The syntax and semantics of the proposed international language of the Zurich ACM-GAMM Conference." Information Processing, UNESCO, 1959.

2. Bartlett, F., Thinking, Basic Books, 1958.

3. Bobrow, D. G., "A question-answering System for high school word algebra problems," Proc. Fall Joint Computer Conference, 25, 1964.

4. Brooke, M., 150 Puzzles in Crypt-arithmetic, Dover, 1963.

5. Ernst, G. W. and Newell, A., GPS and Generality. Carnegie Institute of Technology, 1966.

6. Evans, A., "An Algol 60 Compiler," Annual Review in Automatic Programming, Pergamon Press, 1964.

7. Feigenbaum, E. and Feldman, J., Computers and Thought, McGraw-Hill, 1963.

8. Gagne, R., and Paradise, N. E., "Abilities and learning sets in knowledge acquisition," Psychol. Monogr., Whole No. 518, 1961.

9. Johnson, E. S., "An information processing model of one kind of problem solving," Psychol. Moriogr., Whole No. 581, 1964.

10. Luchins, A. S., "Mechanization in problem-solving," Psychol. Monogr., Whole No. 248, 1942.

11. Markov, A. A., Theory of Algorithms, Academy of Sciences, USSR, 1954.

12. Miller, G., Galanter, G., and Pribram, K., Plans and the Structure of Behavior, Holt, 1960.

13. Newell, A., "Some problems of basic organization in problem-solving programs," in M. C Yovits, G. T. Jacobi and G. D. Goldstein (eds.) Self Organizing Systems, Spartan, 1962.

14. Newell, A., A Guide to the General Problem Solver Program GPS-2-2. The RAND Corporation, 1963.

15. Newell, A., "On the analysis of human problem solving protocols," Proc. International Symposium on Mathematical and Computational Methods in the Social Sciences, 1966, (in press).

16. Newell, A. and Ernst, G. W., "The search for generality," E. W. Kalenich (ed.) Proc. IFIP Congress 65, pp 17-24, Spartan, 1965.

References

17. Newell, A., Shaw, J. C., and Simon, H. A., "The processes of creative thinking," in H. E. Gruber, G. Terrell and M. Wertheimer (eds.) Contemporary Approaches to Creative Thinking, Atherton, 1962.

18. Newell, A. and Simon, H. A., "Computer simulation of human thinking," Science, vol. 134, no. 3495, pp 2011-2017, 1961.

19. Newell, A. and Simon, H. A., "GPS, a program that simulates human thought," in H. Billing (ed.) Lernende Automaten, Oldenbourg, Munich, 1961.

20. Newell, A. and Simon, H. A., "Computers in psychology," in R. D. Luce, R. R. Bush, and E. Galanter (eds.) Handbook of Mathematical Psychology, vol. 1, Wiley, pp 361-428, 1962.

21. Newell, A. and Simon, H. A., "An example of human chess play in the light of chess playing programs," in N. Wiener and J. P. Schade (eds.) Progress in Cybernetics, vol. 2, pp 19-75, Elsevier Publishing Co. Amsterdam, 1965.

22. Paige, J. and Simon, H. A., "Cognitive processes in solving algebra word problems," in B. Kleinmuntz (ed.) Problem Solving: Research Method and Theory, Wiley, 1966.

23. Pittenger, R. F., Hockett, C. F. and Danehy, J. J., The First Five Minutes, Martineau, 1960.

24. Reitman, W., Cognition and Thought, Wiley, 1965.

Crypt-arithmetic
Subject 3 Problem

```
  DONALD
+GERALD
ROBERT
```

D=5

B1	Each letter has one and only one numerical value --	? :		(ask Exp. about rules)
B2	Exp: One numerical value.			
B3	There are ten different letters			
B4	and each of them has one numerical value.			
B5	Therefore, I can, looking at the two D's --	S1:	D←5	→ FC(D)=> c1; PC(c1)=> T=0
B6	each D is 5;			
B7	therefore, T is zero.	T1:	T=0	→ TD(T,0)=> +
B8	So I think I'll start by writing that problem here.			
B9	I'll write 5, 5 is zero.			
B10	Now, do I have any other T's?	S1:	T=0	→ FC(T)=> ∅
B11	No.			
B12	But I have another D.	S1:	D←5	→ FC(D)=> c6 (no PC(c6))
B13	That means I have a 5 over the other side.			
B14	Now I have 2 A's	G4:	get 1s	→ FL(1s)=> R; get R
B15	and 2 L's			
B16	that are each --			
B17	somewhere --			
B18	and this R --			
B19	3 R's --			
B20	2 L's equal an R --	S2:	get R	→ FC(R)=> c2; PC(c2,R)=> R odd
B21	Of course I'm carrying a 1.			
B22	Which will mean that R has to be an odd number.			

B22.1		R1: PC unclear	→ get R; repeat PC
B23	Because the 2 L's --	↑ :	PC(c2,R)=> R odd
B24	any two numbers added together has to be an even number		
B25	and 1 will be an odd number.		
B26	So R can be 1,	S4: get R	→ GN(R)=> 1,3,5,7,9
B27	3,		
B28	not 5,	T1: R=d	→ TD(R,d)=> R=5-p(D≠5!)
B29	7,		
B30	or 9.		
B30.1		? :	
B31	Exp: What are you thinking now?		
B32	Now G --	S2: get R	→ FC(R)=> c6; PC(c6,R)=> G even
B33	Since R is going to be an odd number		
B34	and D is 5,		
B35	G has to be an even number.		
B35.1		R1: PC unclear	→ get G; repeat PC
B36	I'm looking at the left side of this problem here where it says D + G.	↑ :	PC(c6,G)=> t6?
B37	Oh, plus possibly another number,		
B38	if I have to carry 1 from the E + O.		
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.		

B42	Well, if we assume --	S3:	get R	→ FA(R)=> c2; AV(L)=> L←1,
B43	if we assume that L is, say, 1,			
B44	we'll have 1 + 1 that's 3 or R --	↑:		PC(c2,R)=> R=3
B45	We'll put in a 3 here,	T1:	R=3	→ TD(R,3)=> +
B46	and one here.			
B47	Well, 5 plus something has to equal 3 in that case --	S1:	R=3	→ FC(R)=> c6; PC(c6)=> t7=1?
B48	I suppose it's --	G1:	t7=1?	→ get t7=1
B49	Well, not,	S2:	get t7=1	→ FC(t7)=> c7; PC(c7)=> -p(z1!) z1: 1 at r3 of c7 not exist
		G5:	z1!	→ check z1
B50	it's not possible that there could be another letter in front of this R is it?	? :	check z1	→ z2=> ∅ z2: Ask Exp.
B50.1		R1:	z2 unclear	→ get z2; repeat z2
B51	Is it or not?	↑:		z2 => z1 (from Exp.)
B52	Exp: No.			
B53	It's not --			
B54	all right --			
B55	so if --			
B56	if that couldn't be a 13 on the left side,	T2:	t7=1-p	→ FA(t7=1)=> R=3; => R=3-p
B57	then R cannot be 3.			
B57.1		G2:	R=3-p	→ get R
B58	R has to be a number greater than 5,	S2:	get R	→ FC(R)=> c6; PC(c6,R)=> R>5
B59	which means that it can be either 7	S4:	get R	→ GN(R)=> 7,9
B60	or 9.	T1:	R=d	→ TD(R,d)=> +

B61	So we'll start back here and make it a 7.	↑S4:		AV(R)⇒ R←7
B62	Now if the --	S1:	R←7	→ FC(R)⇒ c2; PC(c2)⇒ L=3
B63	Oh, I'm sorry, I said something incorrect here.	T1:	L=3	→ TD(R[sic],3)⇒ L=3-p(R=3-p!)
B64	I'm making --			
B64.1		R1:	PC unclear → get L; repeat PC	
B65	No, no, I didn't either.	↑:		PC(c2,L)⇒ L=3
B66	R is going to be a 7,	T1:	L=3	→ TD(L,3)⇒ +
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.			
B72	Now, it doesn't matter anywhere what the L's are equal to --	S1:	L=3	→ FC(L)⇒ ∅
B73	so I'm independent of L when I get past the second column here.			
B74	But now I know that G has to be either 1	S1:	R←7	→ FC(R)⇒ c6; PC(c6)⇒ G=1,2; t6?
B75	or 2,			
B76	depending on whether or not E + 0 is greater than 10	G1:	t6?	→ get t6
B77	or greater than 9.			
B78	Now I have this 0 repeating here in the second column from the left;	S2:	get t6	→ FC(t6)⇒ c5; PC(c5,t6)⇒ E=0
B79	that is, itself plus another number equal to itself.			
B80	This might indicate that E was zero,			
B80.1		R1:	PC unclear → get E; repeat PC	

B81	In fact, it might have to necessarily indicate that.	↑:		PC(c5,E)=> E=0
B82	I'm not sure.	R1:	PC unclear → get E; repeat PC	
B83	Or, E could be 9	↑:		PC(c5,E)=> E=9,t5=1
B84	and I would be carrying 1,	T1:	E=9	→ TD(E,9)=> +
B85	which would mean that I was then carrying 1 into the left hand column.	S2:	get t6	→ FC(t6)=> c5; PC(c5,t6)=> t6=1
B85.1		? :		
B86	Exp: What are you thinking now?	S2:	get E	→ FC(E)=> c5; PC(c5,E)=> [unclear]
B87	I was just trying to think over what I was just --			
B88	about the possibility --			
B89	the implications of an 0 + another number equaling an 0,			
B90	and what that necessarily implies.			
B90.1		? :		
B91	Let's see --			
B92	I have two A's equaling an E.	S2:	get E	→ FC(E)=> c3; PC(c3,E)=> E even, t3=0!
B93	Therefore, E has to be an even number,			
B94	because I know I'm not carrying 1.	G5:	t3=0!	→ check t3=0
		G3:	check t3=0	→ get t3=0
B95	Of course, this all going on the assumption that R is 7 --	S2:	get t3=0	→ FC(t3)=> c2; PC(c2,t3=0)=> +(R←7!)
B95.1		G5:	R→7!	→ check R←7
		R2:	check R←7	→ repeat S4
B96	R could be 9 also.	S4:	get R	→ GN(R)=> 7,9
B97	Well, maybe I'll just continue to try to work this through again.	? :		
B98	If E has got to be an --	S2:	get E	→ FC(E)=> c3; PC(c3,E)=> E even
B98.1		S4:	get E	→ GN(E)=> 2,4,6,8

B99	Now, wait a second.	T1;	E=d	→ TD(E,d)=> +
B100	I got something out of this.	? :		(9 not in)
B101	E has to be an even number	S2:	get E	→ FC(E)=> c5; PC(c5,E)=> E=0,9
B102	and E + 0 = 0 --			
B103	E cannot be 9.	T1:	E=d	→ TD(E,d)=> E=9-p(E even!)
B103.1		? :		→ (change R, use 9)
B104	Exp: What are you thinking now?			
B105	I'm going back over these L's here and try to think what would happen if they are ni --	S3:	get R	→ FA(R)=> c2; AV(L)=> L←9
B106	rather --	↑:		PC(c2,R)=> R=9
B106.1		T1:	R=9	→ TD(R,9)=> R=9-p(L←9!)
B106.2		T2:	R=9-p	→ FA(R=9)=> L←9; => L←9-p
B107	Let's see, how did I arrive at the point of that?	R1:	PC unclear	→ get R; repeat PC
B108	This is going to be a little confusing to start trying to trace back here.	↑:		PC(c2,R)=> unclear (L←9-p)
B109	What's the reasoning here?			
B110	I'm thinking in the back of my mind what this R was.	? :		(vacuous?)
B111	I decided that R had to be greater than 5,	S2:	get R	→ FC(R)=> c6; PC(c6,R)=> R>5
B112	because that was given			
B113	and R + G,			
B114	or rather, D + G = R.			
B114.1		? :		
B115	I know you're wondering what I'm thinking.	S2:	get E	→ FC(E)=> c3; PC(c3,E)=> E even
B116	I'm still trying to look at this second column here, where E + 0 = 0,	S2:	get E	→ FC(E)=> c5; PC(c5,E)=> E=0,t5=0?

B117	and A + A = E.			
B117.1		G1:	t5=0?	→ get t5=0
B118	Then again, that's assuming that N is less than 3,	S2:	get t5=0	→ FC(t5)=> c4; PC(c4,t5=0)=> N<3
B119	because I don't want to be carrying 1 into that E + 0 column.			
B119.1		? :		
B120	I think I'll try once more here -	S3:	get E	→ FA(E)=> c3; AV(A)=> A←x,
B121	just trying to sort of bluff my way through this.			
B122	That is, just assume some value for A,			
B123	so I can get that E.	↑:		PC(c3,E)=> E=y
B123.1		S1:	E=y	→ FC(E)=> c5; PC(c5)=> O?
B124	I can do better than that.	? :		
B125	I --	S3:	get E	→ FA(E)=> c5; AV(O)=> O←9
B126	I know that E + O has to equal O,			
B127	and, at most, O is going to be 9;			
B128	in which case E would be zero.	↑:		PC(c5,E)=> E=0
B129	If E is zero.	S1:	E=0	→ FC(E)=> c3; PC(c3)=> A=5
B130	A + A --			
B131	But A can't equal 5 --	T1:	A=5	→ TD(A,5)=> A=[illegible](D←5!)
B131.1		R1:	PC unclear	→ get A; repeat PC
B132	That is, A + A would equal E	↑:		PC(c3,A)=> A=5
B133	and if E were zero,			
B134	A would have to equal 5;			
B135	but A can't equal 5.	T1:	A=5	→ TD(A,5)=> A=5-p(D←5!)
B136	And --	T2:	A=5-p	→ FA(A=5)=> E=0; => E=0-p
B137	See --	? :		→ get R

B138	I decided .that R had to be an odd number,	S2: get R	-»FC(R)=> c2; PC(c2,E)=> R odd
B139	and has to be greater than 5	S2: get R	-» FC(R)^> e6; PC(c6,R)=> R>5
B140	which leaves only 7	S4: get R	-» GN(R)=> 7,9
B141	and 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.	f:	AV(R)=> R<-7, R*4
B143.1		? :	
B144	Let's see what do I want that E to be ?		
B145	I think that you're absolutely right.		
B146	It might take a full 30 minutes.		
B147	A + A = E —	S3: get E	-^FA(E)=> c3; AV(A)=> A<HK,
B147.1		t:	PC(c3,E)=> E=y
B148	E + 0 = 0.	SI: E=y	-» FC(E)=> c5; PC(c5)=> 0?
B148.1		? :	
B149	I'd better start back at this 0 here.	S3: get E	-> FA(E)=> c5,
B150	What values could 0 have?	?	GN(0)=> 1; AV(0,1)=O 0U
B151	Suppose 0 were 1		
B152	and E would have to be 9,	t :	PC(c5,E)=> E=9,t5=1
B153	and I'd have to be carrying a 1.		
B153.1		t::	(return to GN)
B154	Suppose —	t :	GN(0)=> 2; AV(0,2)=> 0<-2,
B154.1		t	PC(c5,E)=> E=9,t5=1
B154.2		t	(return to GN)
B155	s'pose —	t	GN(0)
		? :	-> AV(0)=> CH-x,

B156	Actually, that's almost the case no matter what the situation is --	↑S3:		PC(c5,E)-> E=9,t5=1
B156.1		R1:	PC unclear → get E; repeat PC	
B157	Unless E is zero --	↑:		PC(c5,E)=> E=0
B158	But E can't be zero --	? :	E=0	→ (recall c3)=> E=0-p
B158.1		G2:	E=0-p	→ => get E
B159	Therefore, E might have to be 9	S4:	get E	→ GN(E)=> 9; AV(E)=> E=9
B160	and I have to carry	S1:	E=9	→ FC(E)=> c5; PC(c5)=> t5=1
B161	in order to have the 0 = the 0.			
B161.1		S1:	t5=1	→ FC(t5)=> c5; PC(c5)=> t6=1
B162	In that case, it looks like G is going to be 1,	S1:	t6=1	→ FC(t6)=> c6; PC(c6)=> G=1
B163	because I am going to be carrying 1.	T1:	G=1	→ TD(G,1)=> +
B164	I think I'll tentatively put that in there.			
		S1:	G=1	→ FC(G)=> ϕ
B165	And I'll call E --	S1:	E=9	→ FC(E)=> c3; PC(c3)=> E=9-p
B166	Let's see, E can't be 9 though.	R1:	PC unclear	→ get E; repeat PC
B167	It doesn't look like E can be 9,	↑:		PC(c3,A)=> E=9-p
B168	because A + A has to equal E.			
B169	Am I irritating you being so far off the course?	? :		
B170	Exp: No.			
B171	I still feel as though I'm baring my soul to my mind here.			
B172	Exp: What are you thinking now?			
B173	Well, I see you here pacing around the room.			
B174	You have me all worried.			
B175	Now I'm going back to see if I've made some obvious fallacy.	? :		→ get R

B176	Exp: You haven't made any obvious fallacies.			
B177	I sort of thought --			
B178	Of course, I did have the choice of making this R a 9.	S4:	get R	→ GN(R)=> 9; AV(R)=> R←~~9~~
B178.1		? :		
B179	I seem to be running into trouble the way I am here.	S2:	get E=9	→ FC(E=9)=> c3; PC(c3,E=9)=> -p
B180	Having trouble getting this E --			
B181	I can't make E a 9.	R1:	PC unclear	→ get E; repeat PC.
B182	If that were 4 + 4 plus --	↑ :		PC(c3,A)=> A=4,t3=1?
B182.1		G1:	t3=1?	→ get t3=1
B182.2		S2:	get t3=1	→ FC(t3)=> c2; PC(c2,t3=1) => -p(R←~~7~~!)
B182.3		G5:	R←~~7~~!	→ check R←~~7~~
B182.4		R2:	check R←~~7~~	→ repeat S4
		S4:	get R	→ GN(R)=> 7,~~9~~; AV(R)=> R←~~9~~
B183	I think I'll make myself another model on the other side,	? :		
B184	and compare these two.			
B185	Now I had a column 5 + 5 = 10.	? :		→ GNC=> c1; PC(c1)=> +
B186	Now the change I'm going to make is that I'm going to make L a 9.	S3:	get R	→ FA(R)=> c2; AV(L)=> L←~~9~~
B187	Which means --	↑ :		PC(c2,R)=> R=9
B188	I'm sorry --	T1:	R←~~9~~	→ TD(R,~~9~~)=> R=9-p(L←~~9~~!)
B188.1		? :		
B189	I'm going to make R a 9,	S3:	get R	→ FA(R)=> c2; AV(R)=> R←~~9~~
B190	which means that L has to be 4.	↑ :		PC(c2)=> L=4
		T1:	L=4	→ TD(L,4)=> +
B191	I wish that L appeared somewhere else,	S1:	L=4	→ FC(L)=> ∅
B192	since I can get it within at least 1			

P193	or 2 of right.			
B194	Now, since R equals 9,			
B195	and in the far left hand total is going to be a 9,			
B196	and the R in GERALD is going to be 9.			
B197	Now instead of G plus 5 being 9,	S1:	R←9	→ FC(R)=> c6; PC(c6)=> G=3,4; t6?
B198	G can be either 3			
B199	or 4.			
B199.1		G1:	t6?	→ get t6
B199.2		S2:	get t6	→ FC(t6)=> c5; PC(c5,t6)=> E=9,t5=1
B200	Of course now my E can't be a 9,	T1:	E=9	→ TD(E,9)=> E=9-p(R←9!)
B201	since I've used the 9 for R.	? :		
B202	And also am using R as 9. instead of a 7			
B203	makes me think it more likely that I'm going to be carrying 1 into that E + 0 column.	S1:	R←9	→ FC(R)=> c4; PC(c4)=> t5=1
B203.1		S1:	t5=1	→ FC(E)=> c5; PC(c5,E)=> E=9
B203.2		T1:	E=9	→ TD(E,9)=> E=9-p(R←9!)
B203.3		T2:	E=9-p	→ FA(E=9)=> t5=1; => t5=1-p
		G2:	t5=1-p	→ get t5
B204	Oh, or zero --	S4:	get t5	→ GN(t5)=> 0; AV(t5)=> t5←0
B205	Suppose I were to --			
B206	This implies that E is zero perhaps.	S1:	t5←0	→ FC(E)=> c5; PC(c5)=> E=0
B207	But as soon as E is zero	S1:	E=0	→ FC(E)=> c3; PC(c3)=> A=5
B208	that means that A + A has to equal 10,			
B209	and I don't find any way to do that.	T1:	A=5	→ TD(A,5)=> A=5-p(D←5!)

B209.1		R1:	PC unclear	→ get A; repeat PC
B210	Let's see --	↑:		PC(c3,A)=> t3=1?
B210.1		G1:	t3=1?	→ get t3=1
B211	If I could get L to be more than 5 --	S2:	get t3=1	→ FC(t3)=> c2; PC(c2,t3=1)=> L>5?
B211.1		G1:	L>5?	→ get L>5
B211.2		S4:	get L>5	→ GN(L)=> 6; AV(L)=> L←6
B212	On the other hand L + L has to equal R	S1:	L←6	→ FC(L)=> c2; PC(c2)=> R=3
B213	and R has to be greater than 5.	T1:	R=3	→ TD(R,3)=> R=3-p(R>5!)
B214	So I'm in sort of a dilemma in that case.	G5:	R>5!	→ check R>5
		R2:	check R>5	→ repeat S4
B215	Now I really think that R is either 7	S4:	get R	→ GN(R)=> 7,9
B216	or 9			
B217	And let's get back to E + O.	? :		→ get E
B218	Something we're missing here.			
B219	Suppose O were something like 8.	S3:	get E	→ FA(E)=> c5; AV(O)=> O←8
B220	8 plus something has to equal 8.	↑:		PC(c5,E)=> t5=2?
B221	Suppose I would carry 2 from the column.	G1:	t5=2?	→ get t5=2
B222	That's sort of difficult in this particular problem.	S2:	get t5=2	→ FC(t5)=> c4; PC(c4,t5=2)=> -p
B222.1		? :		(vacuous?)
B223	I sort of keep coming up with idea that E should equal zero, or 9.	S4:	get E	→ GN(E)=> 0,9
B224	Of course I've used the 9 --	T1:	E=9	→ TD(E,9)=> E=9-p(R←9!)

B224.1		? :		(interrupted)
B225	Exp: You've used the zero, too.	? :		→ E=0-p(0 used!) (from Exp.)
B225.1		R1:	TD unclear	→ get E=0, repeat TD
B226	Yeah, that's certainly true.	↑ :		TD(E,0)=> E=0-p(T=0!)
B227	I used the zero.			
B227.1		G2:	E=0-p	→ get E
B228	Well, I'm getting into problems here if I can't make E either zero	S4:	get E	→ GN(E)=> 0,9
B229	or 9,	T1:	E=d	→ TD(E,d)=> E=0,9-p(T=0!,R←9!)
B229.1		T2:	E=0,9-p	→ FA(E)=> t5=0,1; => t5=0,1-p
B229.2		G2:	t5=0,1-p	→ get t5[≠0,1] (get t5=2)
B230	and it doesn't seem as though I'm going to be able to carry more than 1 in any case.	S2:	get t5=2	→ FC(t5)=> c4; PC(c4,t5=2)=> -p
B230.1		R1:	PC unclear	→ get t5; repeat PC
B231	That is true.	↑ :		PC(c4,t5)=> t5=1,B=8
B232	The most I could have any column total to would be 18.			
B233	And even that can't occur	T1:	B=8	→ TD(B,8)=> B=8-p(O←8!)
B234	unless the column immediately preceding equals 0.			
B234.1		T2:	B=8-p	→ FA(B=8)=> t5>1; => t5>1-p
B235	I've used the zero,	S4:	get E	→ GN(E)=> 0
B236	so E can't be a zero.	T1:	E=0	→ TD(E,0)=> E=0-p(T=0!)
B237	and A can't be zero.	? :		=> A=0-p(T=0!)
B238	I'm finding difficulty in making it a 9.	T2:	E-p	→ FA(E)=> R←9; => R←9-p

B238.1		? :		
B239	We'd better make E a 9	S4:	get E	→ GN(E)=> 9; AV(E,9)=> E←9
B240	I'm going to have to carry 1	S1:	E←9	→ FC(E)=> c5; PC(c5)=> t5=1
B241	in order to satisfy the other conditions.			
B242	It looks then as though my solution for O is going to be independent of the rest of the column.	S1:	t5=1	→ FC(t5)=> c5; PC(c5)=> O free
B242.1		S4:	O free	→ GN(O)=> 1
B243	But --	T1:	O=1	→ TD(O,1)=> O=1-p(G=1!)
B243.1		T2:	O=1-p	→ FA(O)=> O free; => O free-p
		G2:	O free-p	→ get O
B244	No, it's not either,	? :		(shift to fds instead of ds)
B245	because I'm only going to have.--			
B246	I only have 10 letters to use --			
B247	10 numbers to use for 10 letters.			
B248	So it's probably going to be the last one I ever find.	S4:	O free	→ GN(O)=> ϕ; AV(O)=> O←x
B249	I'll put an x in here for the O.			
B250	Make E a 9.			
B251	which leaves my right hand --			
B252	(noise) --			
B253	which leaves this left hand solution.			
B254	Now, R is 7.			
B255	I have to have it so it carries 1 into the E + O column.	S1:	t5=1	→ FC(t5)=c4; PC(c4)=> N>3,t4?
B256	Yeah, this is looking pretty good right now.	G1:	t4?	→ get t4
B257	I guess I still have a problem.	S2:	get t4	→ FC(t4)=> c3; PC(c3,t4)=>A-p
B257.1		R1:	PC unclear	-> get A; repeat PC

	B258	Yes, I have an awful problem here.	↑:	PC(c3,A)=> A-p
	B259	I can't make A + A = 9 very well.	R1: PC unclear	→ get A, repeat PC
	B260	A + A --	↑:	PC(c3,A)=> A=4,t3=1?
*	B260.1		G1: t3=1?	→ get t3=1
*	B261	There's no place where I can get L + L to equal more than 10,	S2: get t3=1	→ FC(t3)=> c2; PC(c2,t3=1)=> L>5?
	B262	so I could make --	G1: L>5?	→ get L>5
	B262.1		S4: get L>5	→ GN(L)=> 6; AV(L)=> L←6
	B262.2		S1: L←6	→ FC(L)=> c2; PC(c2)=> R=3
	B262.3		T1: R=3	→ TD(R,3)=> R=3-p(R>5!)
	B263	Well, then the problem is over here on this R --	G5: R>5!	→ check R>5
			R2: check R>5	→ repeat S2
	B264	This R, it seems to me, has to be greater than 5.	S2: get R	→ FC(R)=> c6; PC(c6,R)=> R>5
	B264.1		T2: R=3-p	→ FA(R=3)=> L←6; => L←6-p
	B265	Now, suppose these were real big numbers,	G2: L←6-p	→ get L>5
	B266	not just little --		
	B267	not 10, but way more than 10.		
	B268	Suppose that was something like 7	S4: get L>5	→ GN(L)=> 7; AV(L)=> L←7
*	B268.1		S1: L←7	→ FC(L)=> c2; PC(c2)=> R=5
*	B268.2		T1: R=5	→ TD(R,5)=> R=5-p(D←5!)
	B269	No, make it even bigger --	T2: R=5-p	→ FA(R=5)=> L←7; => L←7-p
			G2: L←7-p	→ get L>5
	B270	Make it 8.	S4: get L>5	→ GN(L)=> 8; AV(L)=> L←8
	B271	If we let L be 8 and 8	S1: L←8	→ FC(L)=> c2; PC(c2)=> R=7
	B272	and the 1 carried would be 17,		

B273	so my R is still 7.	T1:	R=7	→ TD(R,7)=> +
B274	Now, I sort of forgot why I was going through all this trouble.			
B275	Oh, yes, I wanted to carry 1 into the A + A column,	? :		(recall get t3=1)
B276	which will make A now equal to 4 + 4,	S1:	t3=1	→ FC(t3)=> c3; PC(c3)=> A=4, E=9
B277	and the 1 I am carrying gives me a 9 for the E,			
B278	which satisfies the requirements in the second column from the left.	S1:	A=4	→ FC(A)=> ϕ
		S1:	E=9	→ FC(E)=> c5; PC(c5)=> +
		G4:	get 1s	→ FL(1s)=> N; get N
B279	Now I have an R + N,	S2:	get N	→ FC(N)=> c4; PC(c4,N)=> N free
B280	7 + something = B.			
B281	This looks like I'm getting into the independent part of the solution right now.			
B282	Of course, it all has to satisfy the fact that I have 10 letters for 10 numbers.			
B283	I rather feel I'm pretty close to it right now.			
B284	Let's see what I am doing here.			
B285	I just need --	S4:	get N	→ GN(N)=> 3,6
B286	I'm only two numbers short, aren't I?			
B287	3 and a 6.			
B288	So I'll make the --	↑ :		AV(N)=> (output unclear)
B288.1		R1:	AV unclear	→ get N; repeat AV
B289	suppose I make the N a --	↑ :		AV(N)=> N←x

B290	Well, I have to carry one in that column.	SI:	N<HK	->»FC(N)=> c4; PC(c4)=> B=y,t5=1
B290.1 •		S2:	get N.	->FC(N)=> c4; PC(c4,N)=> N>2
B291	Well, 3	S4:	get N	GN(N)=> 3,6
B292	or 6,			
B293	either one would do it.	T1:	N=d	-> TD(N,d)=0 +
B294	Suppose I make this a 6.	ts4:		AV(N)=> N<-6
B295	Now, this R plus N is 7 + 6 —	SI:	N<-6	-> FC(N)=> c4; PC(c4)=> B=3
B295.1		T1:	B=3	-> TD(B,3)=> B=3-p(fds^'!)
B295.2		R1:	TD unclear	-> get B; repeat TD
B296	Have to make B a —	t :		->TD(B,3)=> B=3-p(fds=9<'!) .
B297	I must be missing another number here.	G5:	fds=9<'.	-¥check *fds=4*
		G3:	check fds=9(	-» get fds
B298	Yeah, I'm missing 2 also.	S4:	get fds	-> GN(fds)=> 2
B299	7 and 6 is 13,	S2:	get B	-> FC(B)=> c4; PC(c4,B)=> B=3
B300	so B becomes 3,	T1:	B=3	-> TD(B,3)=> +
B301	which leaves me a 2 for 0.	SI:	B=3	-> FC(B)=> *4*
		G4:	get Is	-> FL(1s)=> 0; get 0
		S4:	get 0	-»GN(0)=> 2; AV(0)=> 0<-2
		T1:	0=2	->TD(0,2)=> +
* B302	Now I think I may be satisfied .	SI:	0<-2	-> FC(0)=> *4*
: -		G4:	get Is	-> FL(1s)=> *4*
B303	Probably better check the addition.	?:		-» check cs
		S5:	check cs	. -> GNC (cs)
B304	5 and 5 is 10,	t :		=0 c1; PC(c1)=> +
B305	carry 1;			

B306	8 and 8 is 16	↑:	=> c2; PC(c2)=> +
B307	and 1 is 17		
B308	carry 1;		
B309	4 and 4 is 8	↑:	=> c3; PC(c3)=> +
B310	and 1 is 9;		
B311	7 and 6 are 13,	↑:	=> c4; PC(c4)=> +
B312	carry the 1;		
B313	9 and 2 are 11	↑:	=> c5; PC(c5)=> +
B314	and the 1 is 12;		
B315	5 and 1 is 6	↑:	=> c6; PC(c6)=> +
B316	and 1 is 7.		
B317	Just for the sake of really giving a complete answer,	?:	(get another solution)
B318	I imagine you could shift these numbers around here a little bit to make this --	?:	(method; shift assignments => ϕ,
B319	Well, I really don't know how to check.	?:	
B320	I think I've completed the problem.	?:	(end)
B321	Exp: That's right.		

Notes on protocol

B1 The exchange deals with the definition of the problems, hence is outside the problem space.

B5 The subject has been told that D←5 prior to the start of the tape.

B8 We do not encode writing operations.

B16 After identifying A's and L's, searching for more occurrences. The pattern shows for R in B18-19.

B28 "not 5" shows S3 is generating and testing at same time.

B30.1 Don't know what S3 does after GN.

B35 Shows S3 has ignored carry.

B39 Don't know what the decision is based upon; however, there is no place to go as long as assignments are not made (see B40).

B40 One of the few indications of development (or change) of methods.

B44 -B46 S3 is writing 3's at C2, C4, C6.

B48 "I suppose it's [not possible]." Determined by repeat in B50.

B55 Precursor to B56.

B61 "back here" indicates C2.

B62 "Now if the [R is 7, L must be 3]."

B63 The difficulty is R=3-p coupled with a general confusion between L and R. The continuation through B71 adds support: "its the L's that will have to be 3's." B105 and B186 where S3 assigns L←9 and not R←9, confirms this.

B72 Evidence for FC being evoked after new information derived (L=3).

B74 Note that S3 says G=1,2 not G=2,1. This latter would be expected if he worked without t5 and then remembered it later.

B77 Probably "or less than 9"; but could be a restatement with slight correction of "greater than 10." The ambiguity is created by 1) "whether or not," which would normally be followed by only the single condition and 2) "greater" which is ambiguously $>$ or $\geq$ in casual conversation.

B84 Taken as t5=1, since B85 states t6=1 as a consequence.

B85.1 Don't know how much further S3 goes; e.g., to c6 and G=1; thence to S1, which gives FC(G)=>ϕ.

B87 Clearly reworking C5, but unclear whether any information derived.

B90 Note the "necessarily" and the parallelism of phrasing to B81.

B94 Is the emphasis a precursor to his checking t3 rather than following up on E even?

B95 An alternative interpretation is that S3 simply reflects on the contingency of the current line of attack; however the concern with t3 in B94 makes the chosen interpretation more plausible.

B98 "If E has got to be an [even number]." Note the "an"; also compare B93 and B101. The assumption is that S3 starts counting from 2 and not from zero; otherwise might have seen E=0-p(T=0!).

B99 -B100 It is unclear what clue evokes the possibility that E=9-p. but does not yet settle it. That E=9 has not occurred in the generation is a possibility. In B101-B103 S3 goes through the argument as if for the first time.

B105 "they are ni[ne]" makes clear the assignment is misplaced from R←9 to L←9. This (and B186) might be due to the use of production S3, which to get x assigns a value to a different variable, y.

B106 The confusion, starting here and running to B109, stems from the assignment error. But why so confused, rather than simply recognizing the misassignment? The peculiarities of PC(L,9) make it plausible:

1. 9+9+1 = 19; thus get R=9, which is the true assignment.

2. Thus, to assign R←9 would seem to lead back to L←9, as given above. (That this is not necessary, since 4+4+1=9 as well, would not be apparent.)

3. TD(R,9) leads to rejecting L←9; but once L←9 is rejected then TD(R,9)=>+!

B113 -B114 Can make nothing substantive of the slip.

B114.1 Might have to go on to R=7,9 as he did in B137-B143.

B116 The phrasing of B116, B117 is c5, c3. However, the subsequent behavior concerning c4, which refers to carries into c5, indicates that a repetition of the reasoning from E even (c3) to E=0 (c5) to t4=0 (c4) is going on. Hence the order is c3,c5.

B122 An explicit statement of production S3, implying the ability to go over a method in a particular context without carrying through the calculation in detail.

B124 Do better than to assign A to get E; namely, assign O to get E. Is this better because it is closer to the difficulty; namely E in c5?

B125 Precursor to B126.

B127 Why O←9 rather than O←1? Perhaps because S3 excludes E=9; perhaps because. by maximizing O he maximizes chance of getting t6=1.

B128 If O←9 and E=0,9 then E=0; however, probably PC.

B136 "and [that means E can't equal zero]."

B137 "[Let's] see." Precursor to B138.

B145 Outside the problem space.

B146 Outside the problem space.

B147 The designation of c3, c5 followed by the assignment of values to O, make it plausible that a repeat of B120 to B124 is occurring. An alternative, less structured and less attractive, is the he simply "considers" each in turn.

B154 "Suppose [O were 2]." The grounds for inferring a generation comes from B150, which announces it explicitly, and the parallelism between B154 and B151.

B155 "S'pose [O were --]." Unclear whether he actually sets up another value (O←3) or senses the fact that the reasoning would give the same answer (as indicated B156).

B156 Subject has inducted the general form from a sequence of cases. We have coded this as the assignment of a general variable (AV(O)=>O←x) and the carrying through of a symbolic calculation in PC. PC certainly has these capabilities (B122-B123, B249-B250). An alternative is a mechanism for inducting directly from the invariance of the internal process in PC for the different specific values of O. There need not be any checking with O←x; i.e., no performance of PC after O←x.

B158 Recalls B128-B136. Not E=0-p(T=0!).

B159 "might have to be" indicates the force of "E=0,9 and E=0-p therefore E=9" rather than PC(c5). Also supporting is the "have to carry" (B160), which indicated E=9 imposed from outside c5.

B162 -B165 The only reference to G=1 until it shows in checking the answer (B315). Apparently G=1 was recorded (B164) but E=9 was not (B165-166). Thus, when going back to the first display, the E must be written in (B250) but the G=1 is already there.

B164 There is little evidence for FC(G); the production system demands it and there is no evidence against it.

B169 -B176 The only major interaction on non-task matters with the experimenter.

B177 Unclear what he thought (that R←7 was necessary?).

B182 Indicates either 1) awareness that t3 might be 1; or 2) a consideration of whether it might be so. The decision to try the alternative route (R←9) is probably influenced by the fact the 9>7, but clearly does not represent any detailed consideration of whether R←9 implies t3=1 (it is independent of it, of course).

B185 Simply copying over the first column, not rederiving it. However, still does a PC.

B186 Note the error: means R←9. Compare B63 and B105.

B187 "which means [that R has to be 9]."

B188 The fact that he catches himself more readily then at B105 may indicate some learning. This might simply be recall of recovery at B105.

B191 Good indication of S1 evoked when there is no column to be found. See also B10, B72.

B194 -B196 Writing in R←9.

B202 -B203 This appears to be a place where the noting of R←9 for TD leads to attending to the R in c4, rather than get E in c5. Clearly, sees that R←9 in c4 leads to t5=1, rather than working back from c5 (where in fact E=9-p leads to t5=0).

B204 Now checks c5 and sees that t=0 is implied.

B205 "Suppose I were to [make t5=0]."

B208 Does not consider A+A=0.

B212 Clearly does not see L←8 or L←9, since he thinks L>5 implies R<5. This might be done by general reasoning; trying L←6 seems more plausible.

B219 -B221 Why O←8? Clearly means O from B220. Two alternatives:

1. Since E=0,9 not select O=0,9. If generating from top (see B127) then O←8 is next. However, why generate from top?
2. Confuse O with E so that E=9-p implies try E(=O)←8. Getting t5=2? implies this, since O←8 does not imply anything about t5.

We chose the latter interpretation.

B222 t>1 is not possible with only two addends. However, S3 is not completely sure.

B225 Apparently, the experimenter can contain himself no longer. Too bad.

B226 This makes it clear that B158 did not mean E=0-p(T=0!)

B230 Confirms B222.

B232 Either 1. max = 9+9 and ignore carry.

or 2. max = 9+8+(t=1).

We don't have to choose.

B233 "that" = sum = 18.

B234 Shows still O←8.

B235 -B237 Probably digit oriented action: x=0-p(T=0!) with x=E and x=A. However current production system doesn't accommodate this.

B238 "it" = R (not E), as evidenced by B239.

B240 -B241 "carry" = t5 and the "conditions" are c5 (not c3, see B257).

B242 -B247 "independent" means can be chosen arbitrarily; i.e., by GN. Whether GN=>1 and O=1-p(G=1!) in order to see that this is not possible, is only a conjecture (although the production system generates it).

B248 -B249 Clearly GN(O)=>ϕ, but there is no mechanism to realize it will be the last one and to put O←x.

B251 -B253 Starts to correct current version (R←9), then switches to earlier one (R←7).

B254 Reading off R←7 in c4.

B265 -B268 Making L>6. Confirms L←6 at B211-B214.

B269 Sees 7+7 = 15 -p (D←5!).

B273 This is a check of R←7 but not one that requires "check," since all right at c2.

B274 -B275 Trouble with goal stack. Not implausible because of duration since get t3=1. However, exact mechanism of forgetting and of recall obscure.

B281 Coded simply as analogous to c5 and O (B242-B249), since B and N are mutually undetermined. However, could be more to it. "From now on all letters are undetermined; or "the independent part is localized here in c4 and c5."

B282 Note that there is little hesitancy in asserting here what took substantial effort in B244-B249.

B283 Unclear whether evaluation is more than a way of summarizing that all terms left are "independent." Might be evoked because can't go further, but needs to indicate (to himself) that the failure doesn't mean it can't be done.

B285 "I just need [N and B]."

B286 -B287 Error: short 2,3,6. Unclear why error is made. Possibly, O←x leads to need two, leads to generating the first two d. But then why 3,6 and not 2,3?

B288 "So I'll make the [N]--" Cannot decide on whether N is 3 or 6.

B289 Repeat of B288.

B290 "that column" = c5.

B295 -B296 "Now this R plus N is 7 + 6, [which is 13]."
"Have to make B a [3]."
Detects difficulty from checking with TD, since there are no more digits, and aware (peripherally) that O still to go. However, not a clear inference, so repeats. The break in sentence between B295 and B296 is the clue that something is going on. Alternatively could get B=3 and start to process c5 before realizing fds-∅; however, seems like too much processing.

B304 -B316 Note in all the additions that the carry comes after adding digits of column.

B318 Unclear exactly what is being tried in attempting to get another solution.

B319 "Well, I really don't know how to check [that there aren't other solutions.]"
Subject was trying to be "complete" in B317-318 -- i.e., get all solutions.

	item	result	goal stack	S1 S2 S3 S4	G1 G2 G3 G4 G5	T1 T2	R1 R2	?	error
		D←5	1s						
	B1			-	-			+	
	B5			+	-				
		T=0		-	-	+			
	B8	+		+	-				
		∅		+	-				
					+				
	B20		R,1s	+ -					
	B22.1	R odd		- - -			+		
	B23		R,1s	- -			↑		
	B26	R odd		- - +					
	B28	R=1,3,5,7,9		- -		+			
	B30.1	R=1,3,7,9		- -				+	
	B31		(R,1s)	+ -					
	B35.1	G even		- -			+		
	B36		G,R,1s	- -			↑		
	B39	t6?		- -	-			+	
	B40		R,1s	- +					S2
	B44	L←1		- - ↑					
	B45	R=3	1s	-	-	+			
		+		+	-				
	B48	t7=1?			+ -				
	B49		t7=1,1s	+					
		-p(z1!)	1s		- - +				
			z1!,t7=1,1s					+	
	B50.1	∅					+		
	B51		z2,z1!,t7=1,1s				↑		
	B53	t7=1-p	1s		- -	+			
	B57.1	R=3-p	(1s)		+ -				
	B58		R,1s	+ -					

	item	result	goal stack	S1	S2	S3	S4	G1	G2	G3	G4	G5	T1	T2	R1	R2	?	error
	B59	R>5	(R,ls)			-	+											
		R=7,9				-	-						+					
	B61 ←—	+				-	↑											
	B62	R←7	ls	+							-							
	B63	L=3		-							-		+					
	B64.1	-p(R=3-p!)						-			-	-		-	+			
	B65		L,ls			-	-								↑			
	B66	L=3	ls	-							-		+					
	B72	+		+							-							
		∅		+							-							
	B76	G=1,2;t6?						+			-				?			
	B78		t6,ls		+													
	B80.1	E=0		-	-								-		+			
	B81		E,t6,ls			-	-								↑			
	B82	E=0	t6,ls	-	-								-		+			
	B83		E,t6,ls			-	-								↑			
		E=9,t5=1	t6,ls	-	-								+					
	B85	+		-	+													S1
	B85.1	t6=1	ls	-							-						+	
	B86		(E,ls)		+	-												
	B90.1	(unclear)			-	-									-		+	
	B91		(E,ls)		+	-												
	B94	E even,t3=0!			-	-	-					+			?			?R1
			t3=0!,E,ls						+									
	B95		t3=0,t3=0!,E,ls		+													
	B95.1	+(R←7!)	t3=0!,E,ls									+						
			R←7!,t3=0!,E,ls												+			
	B96						+											
	B97	R=7,9	t3=0!,E,ls												-	+		
	B98		(E,ls)	+	-													

	item	result	goal stack	S1	S2	S3	S4	G1	G2	G3	G4	G5	T1	T2	R1	R2	?	error
	B98.1	E even	(E,ls)		-	-	+											
	B99	E=2,4,6,8			-	-							+					
	B100	+			-	-											+	
	B101	(9 not in)	(E,ls)		+	-												
	B103	E=0,9			-	-	-						+					
	B103.1	E=9-p			-	-	?		-					-			+	
	B104 (B61)		(R,ls)		-	+	?											S2,?S4
	B106	L←9		-	-	↑	?											
	B106.1	R=9	ls	-							-		+					
	B106.2	-p(L←9!)							-		-	-		+				G5
	B107	L=9-p	(R,ls)		-	-	?		-					-	↑			
			R,ls		-	-	?								↑			
	B110 (B58)	(unclear)	(R,ls)		-	-	?								-		+	
	B111				+	-	?											?S4
	B114.1	R>5			-	-	-										+	
	B115 (B98)		(E,ls)		+	-												
	B116	E even			+	-	-											S4
	B117.1	E=0,t5=0?	ls	-				+			-		?					
	B118		t5=0,ls		+													
	B119.1	N<3	ls								-						+	
	B120		(E,ls)		-	+												S2
	B123	A←x		-	-	↑												
	B123.1	E=y		+	-	-												
	B124	O?			-	-		-							?		+	
	B125		(E,ls)		-	+												S2
	B128	O←9		-	-	↑												
	B129	E=0	ls	+							-		?					?T1
	B131	A=5		-							-		+					
	B131.1	-p(D←5!)						-		-	?			-	+			
	B132		A,ls		-	-									↑			

	item	result	goal stack (Si S2 S3 S4	Gl G2 G3 G4 G5	Tl T2	R1 R2	*i*	error
	B135	A=5	Is		–			
	B136	-p(D<-5!)		- ?	+			?G5
	B137 (B23H	E=0-p	(R,1s)				+	
			+ -					
	B139	R odd	+ - -					S4 . 1
	B140	R>5	- - +					
	B142	R=7,9	- " .t		?			
	B143.1	R<-7,R<-9					–	
	B144(B125)		(E,1s)	- +				S2
	B147.1			- - t				
	B148	E-y		+ - -				
	B148.1	0?					! +	
	B149		(E,1s)	- +				S2
	B152	Ck-4						
	B153.1	E=9,t5=1	Is		?			
	B154		(E,1s)					
	B154.1	0*-2						
	E154,.. 2	E=9,t5=1	Is	: ~ !	?			
	B155		(E,1s)					
	B156	0<-x		- - t				
	B156.1 ..	E=9,t5-1	Is		?	+		
	B157		(E,1s)			t		
	B158	E=0	Is		?			
	B158.1	E=0-p		? –				T2,?S4 *
	B159		E,1s	- - –				
	B160	E<-9	Is	–				
	B161.1	t5=1		–				
	B162	t6=1		– -				

	item	result	goal stack	S1	S2	S3	S4	G1	G2	G3	G4	G5	T1	T2	R1	R2	?	error
		G=1	(1s)	-							-		+					
	B164	+		+							-							
		ϕ		+							-							
	B166	E=9-p							-		-			-	+			
	B167		E,1s		-	-									↑			
	B169	E=9-p			-	-	?		-					-			+	
	B175 (B104)		(R,1s)		-	-	-										+	
			R,1s		-	-	+											
	B178.1	R←9	1s	-							-						+	
	B179		(E,1s)		+	-												
	B181	E=9-p			-	-	?		-					-	+			
	B182		E,1s		-	-									↑			
	B182.1	A=4,t3=1?		-	-	-		+					-					
	B182.2		t3=1,E,1s		+													
	B182.3	-p(R←7!)	E,1s		-	-			-			+		-				
	B182.4		R←7!,t3=1,E,1s												+			
							+											
	B183	R←9	t3=1,E,1s	-	-												+	
	B185		R,1s		-	-											+	
	B186	+			-	+												S2
	B187	L←9		-	-	↑												
	B188	R=9	1s	-							-		+					
	B188.1	-p(L←9!)							-		-	-		-			+	
	B189		(R,1s)		-	+												S2
	B190	R←9	1s	-		↑					-							
		L=4		-							-		+					
	B191	+		+							-							
		ϕ		+							-							
	B199.1	G=3,4;t6?						+			-							
	B199.2		t6,1s		+													

	item	result	goal stack	S1	S2	S3	S4	G1	G2	G3	G4	G5	T1	T2	R1	R2	?	error
	B200	E=9,t5=1	(t6,1s)	-	-								+					
	B201	-p(R←9!)			-				-			-		-			+	
	B202 (B199.2)→	(R←9)	(t6,1s)	+	-													
	B203.1	t5=1		+	-													
	B203.2	E=9		-	-								+					
	B203.3	-p(R←9!)			-				-			-		+				G5
		t5=1-p			-				+					-				T2
	B204		t5,t6,1s		-		+											
	B205	t5←0	t6,1s	+	-													
	B207	E=0		+	-								-					T1
	B209	A=5		-	-								+					
	B209.1	-p(D←5!)			-				-		?			-	+			
	B210		A,t6,1s		-	-									↑			
	B210.1	t3=1?			-	-		+										
	B211		t3=1,A,t6,1s		+													
	B211.1	L>5?			-			+										
	B211.2		L>5,A,t6,1s				+											
	B212	L←6	A,t6,1s	+	-	-												
	B213	R=3		-	-	-							+					
	B214	-p(R>5!)			-	-			-		+			-				
			R>5!,R=3,A,t6,1s												+			
	B215						+											
	B217	R=7,9	A,t6,1s		-	-							?				+	
	B218		E,1s		-	+	?											S2,?S4
	B220	O←8		-	-	↑	?											
	B221	t5=2?			-	-	?	+										
	B222		t5=2,E,1s		+													
	B222.1	t5=2-p			-				-					-			+	
	B223		(E,1s)		-	-	+											
	B224	E=0,9			-	-							+					

	item	result	goal stack	S1	S2	S3	S4	G1	G2	G3	G4	G5	T1	T2	R1	R2	?	error
	B224.1	-p(R<9!)	(E,1s)		-	-			-			-		-			+	
	B225 (B224)		(E,1s)		-	-											+	
	B225.1	E=0-p			-	-			-			-		-	+			
	B226		E=0,1s		-										↑			
	B227.1	-p(T=0!)	1s						+		-	?		-				T2,?G5
	B228 (B223)		E,1s		-	-	+											
	B229	E=0,9			-	-							+					
	B229.1	-p(T=0,R<9)	1s						-		-	?		+				?G5
	B229.2	t5=0,1-p	(E,1s)		-	-			+					-				T2
	B230		t5=2,E,1s		+													
	B230.1	t5=2-p	E,1s		-	-			-					-	+			
	B231		t5,E,1s		-										↑			
	B233	t5=1,B=8	E,1s		-	-	-						+					
	B234.1	-p(O<8!)				-	-		-			-		+				G5
	B235	t5>1-p	(E,1s)		-	-	+		-					-				
	B236	E=0	1s	-							-		+					
	B237	-p(T=0!)							-		-	?		-			+	
	B238	A=0-p(!)							-		-	?		+				?G5
	B238.1(B190)	R<9-p	(1s)						-		-			-			+	
	B239 (B189)		E,1s		-	-	+											
	B240	E<9	1s	+							-							
	B242	t5=1		+							-							
	B242.1	O free		-			+				-							
	B243	O=1		-							-		+					
	B243.1	-p(G=1!)							-		-	-		+				G5
		O free-p							+		-			-				T2
	B244		0,1s		-	-											+	
		(O free)			-	-	+											
	B249	O<-x	1s	+							-							
	B256	N>2,t4?						+			-							

	item	result	goal stack	S1	S2	S3	S4	G1	G2	G3	G4	G5	T1	T2	R1	R2	?	error
	B257		t4,1s		+													
	B257.1	A-p			-				-					-		+		
	B258		A,t4,1s		-	-										↑		
	B259	A-p	1s						-		-			-		+		
	B260		A,1s		-	-										↑		
	B260.1	A=4,t3=1?		-	-			+					-					
	B261		t3=1,1s		+													
	B262	L>5?			-			+										
	B262.1		L>5,1s				+											
	B262.2	L←6		+														
	B262.3	R=3	1s	-							-		+					
	B263	-p(R>5!)							-		-	+		-				
			R>5!,R=3,1s												+			
	B264				+													
	B264.1	R>5	1s								-			+				
	B265	L←6-p	(L>5,1s)				?		+									?S4
			L>5,1s				+											
	B268.1	L←7	1s	+							-							
	B268.2	R=5		-							-		+					
	B269	-p(D←5!)							-		-	?		+				?G5
		L←7-p	(L>5,1s)				?		+									?S4
	B270		L>5,1s				+											
	B271	L←8	1s	+							-							
	B273	R=7		-							-		+					
	B274	+									-						+	
	B276	t3=1		+							-							
	B278	A=4,E=9		+							-		?					?T1
		ϕ		+							-							
		+									+							
	B279		N,1s		+	-												

item	result	goal stack	S1	S2	S3	S4	G1	G2	G3	G4	G5	T1	T2	R1	R2	↑	er
B285	N free	(N,1s)		-	-	+											
B288	N=3,6			-	-	↑						-					
B288.1	(unclear)			-	-							?		+			
B289		N,1s		-	-									↑			
B290	N←x		+	-	-												
B290.1	B=y,t3=1		-	+	-												S1
B291	N>2			-	-	+											
B293	N=3,6			-	-	-						+					
B294	+			-	-	↑											
B295	N←6	1s	+							-							
B295.1	B=3		-							-		+					
B295.2	¬p(fds=ϕ!)							-		-	-		-	+			
B296		B,1s		-	-	?								↑			
B297	¬p(fds=ϕ!)			-	-			-			+		-				
		fds!,B,1s							+								
B298		fds,fds!,B,1s				+											
B299	fds=2	B,1s		+	-	?											?S
	B=3	1s	-							-		+					
B301	+		+							-							
	ϕ									+							
		O,1s		-	-	+											
	O←2	1s	-							-		+					
B302	+		+							-							
	ϕ									+							
B303	ϕ	ϕ														+	
		cs!				S5		?									
B304						↑											
B306	T=0,t1=1		-			↑											
B309	R=7,t2=1		-			↑											
B311	E=9,t3=0		-			↑											

	item	result	goal stack	S1	S2	S3	S5 S4	G1	G2	G3	G4	G5	T1	T2	R1	R2	?	error
	B313	B=3, t4=1		-			↑											
	B315	O=2, t5=1		-			↑											
	B317	R=7, t6=0	ϕ	-													+	
	B318																+	
	B319																+	
	B320																+	

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)*
Carnegie-Mellon University
Department of Computer Science
Pittsburgh, Pennsylvania 15213

2a. REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

2b. GROUP

3. REPORT TITLE
STUDIES IN PROBLEM SOLVING: SUBJECT 3 ON THE CRYPT-ARITHMETIC TASK DONALD + GERALD = ROBERT

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

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

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
July 1967	169	24

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
SD-146	
b. PROJECT NO. 9178	
c. 6154501R	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d. 681304	

10. DISTRIBUTION STATEMENT
Distribution of this document is unlimited,

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
TECH,OTHER	Air Force Office of Scientific Research(SRI) 1400 Wilson Boulevard Arlington, Virginia 22209

13. ABSTRACT

The behavior of a single subject on a symbolic task (crypt-arithmetic) is examined in some detail. The study is part of a continuing effort to understand the information processes involved in problem solving. Following a brief review of the advantages and difficulties in using protocols to aid in protocol analysis. These are essentially descriptive in nature, consisting in a specification of the problems space in which the subject is working, and in a display of his behavior, of the total behavior in a series of decision points, and permits an analysis of the adequacy of the productions (essentially, a system of conditional statements). A rather extensive analysis of the adequacy of the production system is included. Finally there is some discussion of the implications of the production system, not only as a descriptive tool, but as a theoretical scheme.

This work is highly detailed in its approach and narrow in its focus. Another paper, A. Newell, "On the Analysis of Human Problem Solving Protocols", is shorter, takes a somewhat broader view and includes the main results of this paper. It is recommended for anyone who is not interested in examining the behavior and the analysis in full detail.

DD FORM 1 NOV 65 1473

Security Classification

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT

Security Classification