Complex Learning Processes[1]

John R. Anderson

Paul J. Kline

Charles M. Beasley Jr.

Department of Psychology

Yale University

New Haven, Connecticut 06520

This research was sponsered by the Personnel and Training Research Programs, Psychological Services Division, Office of Naval Research, under Contract No.: N00014-77-C-0242, Contract Authority Identification Number, NR No. 154-399.

unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: 78-1

2. GOVT ACCESSION NO.:

3. RECIPIENT'S CATALOG NUMBER:

4. TITLE *(and Subtitle)*: Complex Learning Processes

5. TYPE OF REPORT & PERIOD COVERED: Final Technical

6. PERFORMING ORG. REPORT NUMBER:

7. AUTHOR(s): John R. Anderson, Paul J. Kline, & Charles M. Beasley

8. CONTRACT OR GRANT NUMBER(s): N00014-77-C-0242

9. PERFORMING ORGANIZATION NAME AND ADDRESS: Department of Psychology-Yale University Box 11a Yale Station-New Haven, CT 06520

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 66153N RR042-04-01 NR 154-399

11. CONTROLLING OFFICE NAME AND ADDRESS: Personnel and Training Research Programs Office of Naval Research Arlington, VA 22217

12. REPORT DATE: July 18, 1978

13. NUMBER OF PAGES: 88

14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*:

15. SECURITY CLASS. *(of this report)*: unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT *(of this Report)*: Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*:

18. SUPPLEMENTARY NOTES: This research was also supported by NIE-G-77-0005 from the National Institute of Education

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*:

Artificial intelligence
Associative networks
Cognitive Psychology
Computer simulation
Learning
Memory
Production system
Designation
Strengthening
Generalization
Discrimination

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*:

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

This paper describes the ACT theory of the learning of procedures. ACT is a computer simulation program that uses a propositional network to represent knowledge of general facts and a set of productions (condition action rules) to represent knowledge of procedures. There are currently four different mechanisms by which ACT can make additions and modifications to its set of productions as required for procedural learning: designation, strengthening, generalization, and discrimination. Designation refers to the ability of productions to call for the creation of new productions. Strengthening a production may have important consequences for performance, since a production's strength determines the amount of system resources that will be allocated to its processing. Finally, generalization and discrimination refer to complementary processes that produce better performance by either extending or restricting the range of situations in which a production will apply. Each of these four mechanisms is discussed in detail and related to the available psychological data on procedural learning. The small-scale simulations of learning provided as examples are drawn from the domains of language processing and computer programming, since our ultimate goal is for ACT to learn the complex procedures required in such domains.

Abstract

This paper describes the ACT theory of the learning of procedures. ACT is a computer simulation program that uses a propositional network to represent knowledge of general facts and a set of productions (condition⇒action rules) to represent knowledge of procedures. There are currently four different mechanisms by which ACT can make additions and modifications to its set of productions as required for procedural learning: designation, strengthening, generalization, and discrimination. Designation refers to the ability of productions to call for the creation of new productions. Strengthening a production may have important consequences for performance, since a production's strength determines the amount of system resources that will be allocated to its processing. Finally, generalization and discrimination refer to complementary processes that produce better performance by either extending or restricting the range of situations in which a production will apply. Each of these four mechanisms is discussed in detail and related to the available psychological data on procedural learning. The small-scale simulations of learning provided as examples are drawn from the domains of language processing and computer programming, since our ultimate goal is for ACT to learn the complex procedures required in such domains.

Introduction

We are interested in understanding learning. For many years learning theory was practically synonymous with experimental psychology; however, its boundaries have shrunk to such an extent that they barely overlap at all with those of modern cognitive psychology. Cognitive psychologists, by and large, concern themselves with a detailed analysis of the mechanisms that underlie adult-human intelligence. This analysis has gone on too long without adequate attention to the question of how these complex mechanisms could be acquired. In an attempt to answer this question, we have adopted one of the methodological approaches of modern cognitive psychology: Results of detailed experimental analyses of cognitive behaviors are elaborated into a computer simulation of those behaviors. The simulation program provides new predictions for a further experimental testing whose outcome is then used to modify the simulation and the whole process then repeats itself.

Our computer simulation is called ACT; this paper will describe its learning processes as well as describing some initial contact between empirical data and predictions derived from these learning processes. The ACT system embodies the extremely powerful thesis that a single set of learning processes underlies the whole gamut of human learning--from children learning their first language by hearing examples of adult speech to adults learning to program a computer by reading textbook instructions. If we can show that ACT's learning processes can acquire some of the cognitive skills required to master these two very different

domains, we will have made a beginning toward establishing this bold thesis. The failure of traditional learning theory invites skepticism of the claim that a single set of processes underlies all learning. However, since the consequences of such a thesis, if true, are so important, and since it is now possible to construct more sophisticated theories of learning processes by the use of computer simulation, another attempt to establish this thesis seems appropriate.

Chomsky (1965) and others have advocated the opposing point of view that special mechanisms are required to learn language. In fact, an earlier simulation program, LAS, developed by the first author to model language acquisition (Anderson, 1974, 1975, 1977, 1978) used learning mechanisms that were not applicable to other cognitive skills. However, it now appears that LAS's learning mechanisms can be seen as manifestations of more general learning mechanisms.

There were a number of inadequacies in the LAS program. (These are reviewed in detail in Anderson, 1978.) There was an inability to make discriminations, to correct errors, to deal with non-hierarchical aspects of language, to deal with inflections, to properly handle the non-declarative aspects of language, to properly model human limitations in language learning and performance, and to account for the gradualness of human learning. In one way or another, each of these problems could have been handled by additions to the LAS theory--but at great cost to the overall parsimony and elegance of that theory. It seemed that a more elegant resolution was possible only by stepping back to a more general

)

learning approach. We expect that ACT will reproduce many of LAS's learning feats; however, it will do so in a way that will naturally extend to the many problems LAS could not handle. Thus, LAS established what could be done by a set of learning mechanisms and ACT is an attempt to generalize what we have learned from LAS.

The organization of the rest of this paper is as follows: First there is a short description of the non-learning aspects of the ACT production system. Following this there are sections discussing each of the three ways the system has of forming new productions: designation, generalization, and discrimination. The next topic discussed is production strength which serves to integrate the new productions into the behavior of the system to produce better performance. The final sections contain speculations on the origin of designating productions and some directions for future work.

The ACT Production System[2]

The ACT production system can be seen as a considerable extension and modification of the production systems developed at Carnegie-Mellon (Newell, 1972, 1973; Rychener & Newell, 1977). ACT represents its knowledge of general facts in a propositional network. This propositional network uses nodes to represent ideas (roughly) and labelled links, which connect nodes, to represent various types of associations between ideas. Information is organized into propositional units where each proposition is a tree interassociating a number of nodes. While the network aspects of this representation are important

for such ACT processes as spreading activation (to be discussed shortly) for most purposes ACT's data base may be thought of as consisting simply of a set of propositions. For example, ACT might represent the addition problem 32 + 18 by the set of propositions:[3]

(ADD 32 18)

(BEGINS 32 2)

(AFTER 2 3)

(ENDS 32 3)

(BEGINS 18 8)

(AFTER 8 1)

(ENDS 18 1)

ACT represents its procedural knowledge as a set of productions, i.e., (condition => action) rules. The condition is an abstract description of a set of propositions. If propositions can be found in the data base which satisfy this abstract description, the production will perform its action. Actions can both add to the contents of the data base and cause the system to emit observable responses.

Propositions that are added to the data base are treated as sources of activation. The total amount of activation given to a source is divided up among all the terms contained in that proposition and then spread from them out over the links in the propositional network to activate other propositions containing these same terms. The activation of these propositions causes them to be treated as sources in turn (but with a reduced amount of activation) and the process continues until the

activation spread to a proposition is less than the amount the system requires to consider a node active at all. The amount of activation that will accumulate at any given node will depend on the number, strength, and directness of its connections to the original sources of activation.[4]

ACT productions can only have their conditions satisfied by active propositions--a requirement that insures that the system will be most responsive to changes in the contents of its data base. ACT's basic control structure is an iteration through successive cycles, where each cycle consists of a production selection phase followed by an execution phase. On each cycle an APPLYLIST is computed which is a probabilistically defined subset of all of the productions whose conditions are satisfied by active propositions. The probability that a production will be placed on the APPLYLIST depends on the strength (s) of that production relative to the sum (S) of the strengths of all the productions whose conditions mention active nodes; that is, this probability is proportional to s/S. Discussion of the process of assigning a strength to a production will be postponed until a later section; all that needs to be said here is that this strength reflects just how successful past applications of this production have been. Thus one component of the production-selection phase consists of choosing out of all the productions which could apply those which are most likely to apply successfully. Further discussion of the details of production selection and execution is best conducted in the context of an example.

Example Production System

Table 1 presents a set of productions for adding two numbers.[5] Since it is difficult to grasp the flow of control among the productions in Table 1, this information is presented diagrammatically in Figure 1 which may be useful in understanding the discussion of the addition productions that follows. There are a number of notational conventions in this figure: Productions are represented as arrows connecting states represented by circles. Each arrow is labelled by the production which it represents. The state circle at the head of an arrow shows the action of the production. The arrows for other productions which need these actions performed in order to apply are shown originating from this state circle. When two or more productions originate from a state circle, additional information from the data base must be examined in order to decide which production should apply. Such additional conditions are represented in diamonds adjacent to the production numbers. The state circle at the tail of a production arrow along with the adjacent diamond totally account for the condition of that production.

Insert Table 1 about here

Insert Figure 1 about here

Suppose the addition problem 32 + 18 is in ACT's data base in the format described above (p. xx). Then the condition of production P1 is satisfied by making the following correspondences between elements of the condition and propositions in the data base:

add LVnumber1 and LVnumber2 = (ADD 32 18)

LVnumber1 begins with LVdigit1 = (BEGINS 32 2)

LVnumber2 begins with LVdigit2 = (BEGINS 18 8)

In making these correspondences, the variables LVnumber1, LVnumber2, LVdigit1, and LVdigit2 are bound to the values 32, 18, 2, and 8 respectively. The LV prefix indicates that these are local variables and can be bound to anything. They only maintain their binding within the production. Other productions are not constrained to match these variables in the same way. In contrast, there are global variables (GV prefix) which, once bound, keep their values in subsequent productions unless explicitly rebound.

The action of P1, add LVdigit1 and LVdigit2 becomes, given the values of the variables, an instruction to place the proposition (ADD 2 8) into the data base. The action of P1 also sets global variables to the digits in the first column.

After the execution of P1 the first element of the condition of production P2 is satisfied:

If GVdigit1 and GVdigit2 are being added = (ADD 2 8).

The remaining condition of P2 matches a proposition in the data base about integer addition:

LVsum is the sum of GVdigit1 and GVdigit2 = (10 = 2 + 8)

The action of P2 simply sets GVsum to this sum.

Productions which require that GVsum have a value can now apply. In particular production P6 is matched as follows:

GVsum > 9 = (10 > 9)

GVsum is the sum of LVdigit3 and 10 = (10 = 0 + 10).

Since this is the first column in the problem, the final requirement of P6, that there be no proposition in the data base indicating a carry into this column, is obviously satisfied. The action of P6 writes out 0 as the first digit in the answer and places a proposition in the data base, (DO-NEXT 2 8 CARRY), to the effect that this column is finished and a carry should be made into the next column.

It may be worth considering why no other production besides P6 can apply. Production P4 fails because there is a proposition in the data base, (10 > 9), inconsistent with the requirement that GVsum is not > 9. Productions P5 and P7 do not apply because there is no carry into the first column. One might wonder why P1 or P2 do not apply again since their conditions were satisfied once by data base elements that have not been changed. The current version of the ACT production system does not allow production conditions to match twice to exactly the same data-base propositions. This constraint serves to avoid unwanted repetitions of the same productions and the danger of infinite loops.

Production P12 applies next, resetting GVdigit1 to 3 and GVdigit2 to 1 and entering (ADD 3 1) into the data base so that the next column can be added. Production P3 sets GVsum to 4 obtained from the data base proposition (4 = 3 + 1). P3 applies here rather than P2 although the condition of P2 is also satisfied. This is because the condition elements of P2 are a proper subset of those of P3. This principle is referred to as specificity ordering in what follows because it results in more specific productions applying in place of more general ones.

Production P5 adds the carry to GVsum and writes out the second digit of the answer, 5. P11 then applies, noting that the problem is finished.

This example illustrates a number of important features of the ACT production system.

(1) Individual productions act on the information in long-term memory. They communicate with one another by entering information into memory and setting global variables.

(2) Productions tend to apply in sequences where one production applies after another has entered some element into the data base. Thus the action of one production can help evoke other productions.

(3) The condition of a production describes an abstract pattern of propositions in the data base. The more propositions that a condition requires in its pattern, the more difficult it is to satisfy that condition. Similarly, the more a condition relies on constants instead of variables to describe its pattern, the more difficult it is to satisfy that condition.

Production Designation

ACT needs the ability to augment its set of productions with new productions. For this reason, productions can designate the construction of other productions in their actions in much the same way that they designate the construction of memory structure. Production designation is an important means by which ACT learns procedural skills.

Encoding of Procedural Instructions

As a first example of procedural learning, let us consider how production designation can be used to assimilate the lessons provided by instruction. Consider how ACT might assimilate the following rules defining various types of LISP expressions (adapted from the second chapter of Weissman, 1967):

1. If an expression is a number it is an atom.

2. If an expression is a literal (a string of characters) it is an atom.

3. If an expression is an atom it is an S-expression.

4. If an expression is a dotted pair, it is an S-expression.

5. If an expression begins with a left parenthesis, followed by an S-expression, followed by a dot, followed by an S-expression, followed by a right parenthesis, it is a dotted pair.

After receiving this instruction ACT will have the sentences expressing these rules represented in its data base. However, this representation, by itself, does not allow it to perform any of the cognitive operations that would normally be thought of as demonstrating an "understanding" of these rules. In order to obtain such an understanding, a means of integrating these rules into ACT's procedural

knowledge is required. Since these rules have the form of conditionals (antecedent implies consequent), they can be translated in a fairly straightforward manner into the condition-action format of productions. Table 2 illustrates four ACT productions for performing such a translation.[6] Production P18 handles the antecedents of the first four conditionals. For example, P18 matches the segment If an expression is a number... of rule (1) by binding LVword to the word number and LVconcept1 to the concept @NUMBER that ACT considers underlies the word. Its action is to save the proposition If there is a @NUMBER by attaching it to GVhold.

Insert Table 2 about here

Production P19 is responsible for actually building the productions encoding these rules. It obtains the conditions of these new productions from the global variable GVhold, which is given a value by other productions, and it obtains the actions from its own processing of the consequent parts of the rules. For example, in the case of rule (1), P19 applies after P18, matching the remainder of the sentence ...number it is an atom. GVword had been previously fixed to number by P18; the local variables LVword and LVconcept had no prior constraints (by the definition of a local variable) and received values of atom and @ATOM, respectively, in the process of matching. The action of P19 builds the production:

P22: IF there is a @NUMBER
THEN it is an @ATOM

Production P22 is the mechanism by which ACT can actually make the inferences authorized by rule (1).

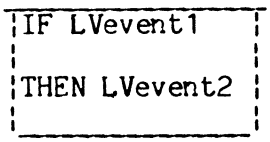
Productions P20 and P21 are responsible for processing complex conditionals like (5). P20 processes the first begins phrase and P21 each subsequent followed by phrase. GVhold has as its value all of the condition elements collected by P20 and P21. After the antecedent of the conditional has been entirely processed, production P19 will apply to process the consequent and then designate a production. In the case of rule (5) this production would be:

P23: IF an expression begins with a @LEFT-PARENTHESIS
and this @LEFT-PARENTHESIS is before an @S-EXPRESSION
and this @S-expression is before a @DOT
and this @DOT is before an @S-EXPRESSION
and this @S-EXPRESSION is before a @RIGHT-PARENTHESIS
THEN it is a @DOTTED-PAIR

Designation with Substitution

The power of the designation mechanisms can be greatly increased by simply allowing substitutions of one item for another throughout a designated production. For example, consider the following production which might be useful in learning by modelling:

P24: IF when LVmodel sees LVevent1
another event, LVevent2, occurs
consisting of LVmodel doing LVaction

THEN BUILD 

substituting ACT for all occurrences of LVmodel

Applied in a situation where Mommy says Hi to Alice after seeing her wave, P24 will designate:

P25: IF Alice waves to ACT

THEN ACT say "Hi"

The substitution mechanism also allows ACT to handle implicit variables in definitions. For example, when CONS(A B) = (A . B) is offered as a definition (rather than an example) of the LISP function CONS, A and B are implicitly variables. ACT knows this, in the sense that when it designates a production to encode this definition of CONS it substitutes variables for the constants appearing as arguments.

Generalization

It is the ability to perform successfully in novel situations that is the hallmark of human cognition. For example, productivity has often been identified as the most important feature of natural languages, where this refers to the speaker's ability to generate and comprehend utterances never before encountered. Traditional learning theories are generally considered inadequate to account for this productivity and ACT's generalization abilities must eventually be evaluated against this same standard.

While it is possible for ACT to designate new productions to

apply in situations where existing ones do not, this kind of generalization requires having designating productions that correctly anticipate future needs. It is plausible that ACT could have such designating productions to guide its generalizations in areas in which it possesses some expertise. For example, if ACT were learning a second language, its experience with its first language might reasonably lead it to expect that the syntactic rules of this new language would treat whole classes of morphemes as equivalent (e.g., the class of all nouns), rather than including different syntactic rules for each individual morpheme. ACT's ability to substitute variables for constants when designating new productions would allow it to capitalize on this expectation and immediately generalize its competence beyond those sentences in the second language that it had actually observed.

It would be much more controversial to attribute such sophisticated expectations to ACT when it learns a first language; and even if it turned out to be justified in this case, it is highly unlikely that sophisticated expectations are available in all cases in which people can make generalizations. For this reason, ACT has the ability to create new productions automatically that are generalizations of its existing productions. This ability, while less powerful than the ability to designate generalizations, is applicable even in cases where ACT has no reliable expectations about the characteristics of the material it must learn.

Examples used to illustrate ACT's automatic generalization

mechanism will draw on productions from Table 3. Production P26 is a designating production which builds comprehension productions. It takes a sentence spoken by a teacher and makes it the condition of a production whose action is ACT's representation of the event the teacher is thought to be describing. When ACT hears this sentence in the future this comprehension production will allow it to understand that another instance of the event the teacher described has occurred.

Insert Table 3 about here

Productions P27 and P28 were built by production P26 based on pairings of the sentences John gave the ball to Jane and Bill gave the dolly to Mary with the events they describe. ACT's automatic generalization mechanism forms a new production P29 which has variables in place of the constants that differ in these two designated productions. Production P29 will handle any sentence of the form LVagent gave the LVobject to LVrecipient and thus extends ACT's competence far beyond the specific examples encountered.

Formal Definitions

Further discussion of the properties of ACT's automatic generalization mechanism requires a formal definition (adapted from Vere, 1977): A production C_1 => A_1 is considered a generalization of C_2 => A_2, if C_1 => A_1 can apply in every circumstance that C_2 => A_2 can (and possibly others); and in these circumstances C_2 => A_2 would cause just the same changes to the data base as C_1 => A_1. We can specify the

conditions under which one production will be a generalization of another: Consider any consistent scheme for replacing local variables and constants in $\underline{C}_2$ by local variables in $\underline{C}_1$. We will refer to this as a substitution θ. Let $\theta\underline{C}_2$ denote $\underline{C}_2$ after these substitutions have been made. Similarly, let $\theta\underline{A}_2$ denote the action after the same substitutions. Then $\underline{C}_1 \Rightarrow \underline{A}_1$ is a generalization of $\underline{C}_2 \Rightarrow \underline{A}_2$ if and only if there is some θ such that $\underline{C}_1 \subseteq \theta\underline{C}_2$ and $\underline{A}_1 = \theta\underline{A}_2$.

Consider how this definition can be applied to show that production P29 (which corresponds to $\underline{C}_1 \Rightarrow \underline{A}_1$ in the definition) is a generalization of production P28 (which corresponds to $\underline{C}_2 \Rightarrow \underline{A}_2$): The substitution 0 will replace Bill in P28 by LVagent from P29. Similarly dolly will be replaced by LVobject, and Mary by LVrecipient. After the substitution 0, the two productions are identical; in the terms of the definition, $\underline{C}_1 \subseteq \theta\underline{C}_2$ and $\underline{A}_1 = \theta\underline{A}_2$. The fact that P29 is a generalization of P28 will be denoted by P29 < P28.

The result $\underline{C}_1 = \theta\underline{C}_2$ is stronger than what is required by the definition of generalization ($\underline{C}_1 \subseteq \theta\underline{C}_2$), which means that in forming the generalization P29 from P27 and P28, ACT could have deleted some condition clauses as well as substituting variables for constants. The reason no clauses were deleted is that ACT forms <u>maximal common generalizations</u> (this concept is also due to Vere, 1977). P29 is a maximal common generalization of P27 and P28 because P29 < P27 and P29 < P28 and there exists no production P such that P29 < P, P < P27, and P < P28. A maximal common generalization of P27 and P28 is one that deletes

the minimum number of their clauses and replaces the minimum number of their constants by variables.

Productions P30 and P31 are the immediate results of a sequence of training trials whose eventual outcome is the generalization P32. P32 will comprehend all statements of the form LVagent gave to Lvrecipient the LVobject. These training trials were performed to demonstrate that ACT would properly distinguish the two different sentence structures for the verb gave and would not form a generalization of them that would handle all sentences containing this verb. There is no way to substitute corresponding variables from the condition of P29 into P32 to produce the identity of actions required by the definition of generalization.

There are occasions on which the maximal common generalization of two perfectly reasonable productions is a production that we would not want ACT to have. For example, consider the following pair of productions:

P33: IF there is a LVlocation in Asia
which is wet and hot and flat
THEN rice can be grown in this LVlocation

P34: IF there is a LVlocation in Vietnam
which has roads,
which is near the river,
but which is not in the mountains
THEN rice can be grown in this LVlocation

Their maximal common generalization is:

P35: IF there is a LVlocation in a LVplace

THEN rice can be grown in this LVlocation

Here the perceived lack of commonality among these two sets of requirements for rice growing has led to the spurious generalization that rice can be grown anywhere. To avoid such obviously spurious generalizations, a restriction is placed on the number of clauses that can be deleted in producing a generalization. If k is the number of clauses in the smaller of the two conditions, then no generalization will be formed if more than $.25k$ clauses must be deleted.

The Problem of Efficiency

A number of other researchers (e.g., Hayes-Roth & McDermott, 1977; and Vere, 1975, 1977, in press) have also worked on generalization routines for production systems. Their routines use different computational techniques to produce generalizations of pairs of productions. ACT's generalization routine uses a rather brute-force technique that tries to put clauses from the two productions into correspondence by substitution of variables. Clauses which have no corresponding member in the other production are not included in the generalization. If there are n clauses in the condition of one production and m clauses in the other ($n > m$) there are potentially $n!/(n - m)!$ ways to assign correspondences. ACT's generalization routine manages to achieve some efficiency by the use of heuristics to guide the search for corresponding clauses. However, there is a sense in which research directed toward discovering efficient algorithms for generalizing two productions is hopeless. Hayes-Roth (1977) has observed

that the generalization problem in its most general form is an NP-complete problem. Since it is widely believed that the time to solve NP-complete problems must be an exponential function of the complexity of the problem, there is probably no entirely satisfactory algorithm for generalization.

Several features of ACT's generalization routine were motivated by this inevitable computational inefficiency. The first of these is that a limit is placed on the amount of computing time that will be spent trying to generalize any pair of productions. The second is that an attempt is made to generalize as few pairs as possible. A realistic simulation of an adult-human's entire procedural knowledge would require hundreds of thousands of ACT productions. Under these circumstances it would be disastrous to attempt to generalize all possible pairs of productions. Not only would this be astronomically costly, but it would produce many spurious generalizations. ACT only attempts to form generalizations when a new production has been designated. Although no potential generalizations would be missed if a generalization was attempted for each possible pairing of this newly-designated production with an existing production, an enormous computational cost is required even under this scheme. For this reason generalizations are attempted only for pairings of newly-designated productions with the productions on the APPLYLIST. Since a production is on the APPLYLIST only if the constants it references are active and it has met a strength criterion (see p.xx), this implies that attempts to generalize will be restricted to productions that are relevant to the current context and which have had a fair history of past success.

Overgeneralization

Since ACT's automatic generalization mechanism extrapolates beyond observed situations, it is bound to make errors. However, given the goal of a realistic psychological simulation, such overgeneralizations on ACT's part would actually be desirable if it could be shown that people also overgeneralize in similar ways. For example, children learning language (and, it appears, adults learning a second language--see Bailey, Madden, & Krashen, 1974) overgeneralize morphemic rules. Thus a child will generate mans, gived, etc. ACT will do the same.

Insert Table 4 about here

The following example illustrates some of the ways in which ACT will overgeneralize. Suppose ACT has the set of productions shown in Table 4 for learning the syntactic structure of simple agent-action-object sentences. ACT brings to this effort the knowledge that certain morphemes refer to certain semantic categories. For instance, it knows that dog refers to the category @DOG. When it encounters a known morpheme it will assume that the semantic category is being referred to and build this information into the production. However, when it encounters an unknown morpheme it skips over it. In this example ACT starts out not knowing how morphemes signal tense and number.

To learn the syntactic structure of a simple sentence, the productions in Table 4 require that the sentence can be paired with the

event it describes. Productions P36-P39 step through the sentence, collecting all the semantic relations provided by the morphemes whose meanings are known to ACT. Once they are finished, production P40 designates a production which will say this sentence in response to any other event that has occurred during the same time and that is given the same semantic categorization by these known morphemes. For example, when an adult model says The dog chases the cat as a description of some event occurring at TIME1, these productions will cause the designation of:

P41: IF @DOGS are @CHASING @CATS at TIME1
and the morpheme "dog" refers to the category @DOGS
and the morpheme "chase" refers to the category @CHASING
and the morpheme "cat" refers to the category @CATS
THEN say "The dog chase+s the cat"

Once ACT has P41 it will say The dog chases the cat in response to events which should actually be described by The dogs chase the cat. It will also use this sentence to describe events that should be described, The dogs chased the cat (when TIME1 is no longer present). This shows that while the productions in Table 4 will designate only correct sentences if all the relevant morpheme-to-semantic-category correspondences are known, they will designate overgeneral productions in the absence of complete knowledge (i.e. what "+s" and "+ed" signal). Thus, directly designated productions can be overly general even before automatic generalization comes into the picture.

In addition, the automatic generalization mechanism will be shown

to act in such a way as to compound this overgeneralization. The distinction between morphemes and the semantic categories they refer to will be ignored in what follows to simplify the exposition. Thus, for example, the production designated when an adult says The cat kills the rat to describe an event will be abbreviated by:

P42: IF cat happens to kill rat at time2
THEN say "The cat kill+s the rat"

Insert Table 5 about here

In response to the pair of productions P41 and P42, the automatic generalization mechanism will produce production P43 in Table 5. P43 generates what is really a present-tense, singular-subject sentence regardless of the actual tense or plurality requirements of the event that sentence is supposed to describe. The other productions in Table 5 are similar overgeneralizations produced in response to other examples of grammatical speech. For example, production P46 is a generalization over the productions designated in response to the adult sentences The dogs chased the cat and The cats killed the rat. Since all four productions in Table 5 have identical conditions, as far as these productions are concerned the choice of inflection for subject and verb is entirely arbitrary. Another overgeneral feature of the productions in Table 5 is that they would apply to irregular words generating items like mans and gived.

Thus with the acquisition of the productions in Table 4 and 5,

ACT has passed from a state of never using the morphemes that express tense and number to a state in which they are used more or less haphazardly. While there is evidence for similar transitions in the empirical literature on language acquisition, it is also the case that people eventually learn to correct their overgeneralizations. The correction of overgeneralizations is primarily the responsibility of ACT's automatic discrimination mechanism.

Discrimination

One response to the problem of overgeneral productions is to designate new productions that apply in a more limited range of circumstances. However, just as in the case of designated generalization, the existence of the required designating productions is plausible only for domains in which ACT already possesses some expertise. In such domains, ACT could possess the knowledge required to debug its own errors intelligently, but in the majority of cases it will rely on its automatic discrimination mechanism. For example, ACT's automatic discrimination mechanism can form the new, correctly-restricted, productions of Table 6 from their overgeneral counterparts in Table 5 without recourse to any specific hypotheses about the nature of the material that must be learned.

Insert Table 6 about here

An Earlier Discrimination Algorithm

A comparison of any pair of corresponding productions from these

tables shows that the correctly discriminated member of the pair contains additional propositions in its condition involving the variables that occur in the condition of the overgeneral member of the pair. These additional propositions function to restrict the variable-bindings that will satisfy the condition of the discriminate production to some subset of those variable-bindings that will satisfy the condition of the overgeneral production. If the automatic discrimination mechanism can find additional propositions which restrict the set of variable bindings in just the right way, then the overgeneralization will be corrected.

Every time a production applies, there is an opportunity to obtain a new set of bindings for its variables. A proposition can then be chosen out of all of those in the data base that mention any of these new bindings. This proposition (appropriately variabilized) can then be added to those in the condition of the production that has just applied to form a new discriminate production with the same action. (It should be emphasized that the discriminate production does not replace the one it was formed from; productions used as the basis for discrimination or generalization continue to exist in the system alongside their "offspring.")

For example, the overgeneral production P43 might apply to generate the sentence The girl hits the boy to describe an event that occurred at TIME3. If there is a proposition in the data base stating that TIME3 is the present time, this proposition could be chosen to produce the discriminate production:

P51: IF LVagent happens to LVact on LVobject at LVtime

and LVtime is present

THEN say "The LVagent LVact+s the LVobject"

A subsequent discrimination of P51 that chooses a proposition stating that the agent is singular would be required to produce the correct production P47 in Table 6.

As long as appropriate propositions are somewhere in the data base, a random choice out of all the propositions that mention new variable bindings is all that is required to guarantee that correct discriminations will eventually be found without any recourse to specific hypotheses about the nature of the material that must be learned. However, the power of random choice is always bought at some cost in efficiency. An earlier version of the automatic discrimination mechanism did randomly choose a proposition to form a new discrimination after every production application. However, very large numbers of discriminations were generated before the correct one was formed.

The Current Discrimination Algorithm

A new discrimination algorithm was developed that greatly increases efficiency. This algorithm makes a distinction between correct and incorrect actions. Productions place new propositions into the data base and emit observable responses; either of these actions can be declared incorrect by a human observer or by ACT itself. In the absence of such a declaration an action is considered correct. That is, the only distinction made by the discrimination mechanism is between negative

feedback and its absence (a later section will take up a possible role for positive feedback). Since the way in which ACT declares that the action of a production is incorrect is to apply another production that makes such a declaration as part of its own action, arbitrarily complex ACT computations can be performed to decide the correctness of any particular action.

The current automatic discrimination mechanism will only attempt to discriminate a production when it has both a correct and an incorrect application of that production to compare. Consider two applications of P43, one of which correctly generates The boy hits the girl to describe a present tense situation and the other which incorrectly generates this same sentence to describe a past tense situation. Suppose the only difference between the variable bindings in these two applications was that LVtime was bound to TIME4 when the present-tense sentence The boy hits the girl was correctly generated, and bound to TIME5 when it was incorrectly generated, i.e., when the action took place in the past. Thus, assuming that ACT has received the appropriate feedback, it can correct its behavior if it can discover the relevant difference between TIME4 and TIME5.

A search is made for propositions mentioning the binding which occurred in the later of the two applications. In the case where the correct binding, TIME4, occurred in the later application this search might find the proposition TIME4 is present. However, before using this proposition to form the discrimination P51, a check is made that the

analogous proposition TIME5 is present was not also in the data base at the time of the first, unsuccessful, application. Finding such a proposition would show that the contemplated discrimination P51 would not have avoided the error made by the overgeneral P43. An attempt would then be made to find another proposition mentioning TIME4 which might better discriminate between successful and unsuccessful applications. If all propositions examined in this way fail. ACT forms no new production--it is possible that the feedback it received was unreliable.

In the case where the later of the two actions was the unsuccessful one, the proposition TIME5 is past might be found which mentions the binding of interest. Since the analogous proposition TIME4 is past was not in the data base at the time of the earlier, successful, application a discriminate production with an absence condition is formed:

P52: IF LVagent happens to LVact on LVobject at LVtime
and LVtime is not past
THEN say "The LVagent LVact+s the LVobject"

The current automatic discrimination mechanism also attempts to speed up the process of finding useful discriminations by its method of selecting propositions from the data base. While still using a random process so as to maintain the guarantee that if the appropriate propositions are in the data base they will eventually be found, this random choice is biased in certain ways that reflect general hypotheses about what sorts of propositions are likely to be incorporated by correct discriminations. Since the greater the amount of activation that has spread to a proposition the more relevant this proposition is likely to be to the current situation, the discrimination mechanism chooses propositions with probabilities that vary with their activation levels. Since the strength of a proposition's interconnections to associated propositions is an overall indicator of its past usefulness, the discrimination mechanism also chooses propositions with probabilities that vary with their average strengths of association.

Discrimination by Specificity Ordering

The use of all these efficiency-promoting devices allows the automatic discrimination mechanism to correct rather quickly the overgeneral productions in Table 5 when provided with feedback about the sentences these productions generate. However, our experience with the simulations performed to date is that while correct behavior on ACT's

part is obtained rather quickly, it is produced by a somewhat different set of productions than the completely-discriminated ones shown in Table 6. Although discriminations that add one additional proposition (e.g., P51) are obtained in all four cases, once completely discriminated productions are formed in two of the cases, they block the erroneous applications required to complete discrimination in the remaining two cases.

Insert Table 7 about here

To be more specific, suppose we have formed the discriminations shown in Table 7. Two of these productions, P48 and P49, are from Table 6. Each of these is included in just one cell in Table 7 showing that they are applicable to only one combination of tense and number, i.e., they are completely discriminated.

On the other hand, Table 7 also contains the incomplete discriminations P53 and P54:

P53: IF LVagent happens to LVact on Lvobject at LVtime

and ~~LVtime is present~~ LVagent is singular

THEN say "The LVagent LVact+s the LVobject"

P54: IF LVagent happens to LVact on LVobject at LVtime

and LVagent is plural

THEN say "The LVagent+s LVact+ed the LVobject"

Each of these are included in two cells reflecting their overgeneral

status. Cells in which they are the sole occupants indicate the combinations of tense and number for which they generate correct sentences, while membership in other cells indicates circumstances in which they will apply and produce errors. However, the left-to-right ordering of productions in these latter cells corresponds to their specificity ordering (p. xx); so, for example, if P49 is selected it will apply instead of P53 thereby preventing an error. In effect, the specificity ordering provides the needed additional discriminations. The control structure we have in Table 7 can be indicated:

If singular Then | If past Then apply P49
| Else apply P53
Else If plural Then If present Then apply P48
Else apply P54

While an <u>if-then</u> control structure is easily implemented in a production system, this <u>if-then-else</u> structure is possible in ACT only because of the specificity-ordering principle. The participation of P53 in this <u>if-then-else</u> control structure restricts its application in exactly the same way that the addition of the new condition clause <u>and LVtime is not past</u> would in an <u>if-then</u> control structure. This is the sense in which the specificity-ordering principle can produce discrimination.

Since P48 will prevent P54 from making errors in a similar fashion, these four productions in Table 7 produce errorless performance as long as the completely discriminated ones get selected (a probabilistic process) whenever they can apply. However, erroneous

applications of P53 and P54 are just what are required to produce the completely discriminated productions that would occupy the major diagonal cells of Table 7 (i.e., P47 and P50).

Irregular Verbs

There are cases in which productions produced on the way to obtaining those in Table 6 are more than mere stepping stones. Notice that the productions in Table 6 generate grammatical sentences only for regular verbs--they would generate The girl hitted the boy to describe an event occurring in the past. Now based on experiences where the sentences The girl hit the boy and The boy hit the girl were paired with the events they described, the following generalization would have been formed:

P55: IF LVagent happens to hit LVobject at LVtime
THEN say "The LVagent hit the LVobject"

This production would sometimes apply incorrectly, perhaps to describe an event that would be correctly described by The girl hits the boy. Punishment of such errors would eventually result in a discrimination which correctly handles the irregular verb hit:

P56: IF LVagent happens to hit LVobject at LVtime
and LVtime is past
THEN say "The LVagent hit the LVobject"

Production Strength

The usual situation is for a number of ACT's productions to all have their conditions satisfied at the same time. On the one hand, this gives ACT a capability for parallel processing which, we have argued elsewhere (Anderson, Kline, & Lewis, 1977), is crucial for an accurate simulation of complex cognitive skills like language processing. On the other hand, the assumption of the ACT model of procedural learning is that the acquisition of most complex cognitive skills requires trying out competing sets of productions for performing the same task. These competing productions would all tend to have their conditions satisfied at the same time and to differ only in the appropriateness of their actions. The strength of an ACT production is a number which is interpreted as a predictor of this appropriateness. Decisions about which productions will actually apply, out of all those satisfied in any given situation, are made largely on the basis of their strengths. Consequently, ACT's ability to adjust the strengths of productions is an important component of its learning.

Adjustments to Strength

Since a production will not apply if it is not strong enough to be placed on the APPLYLIST (see p. xx), the impact of a production on ACT's performance depends crucially on ~~the~~ that production's strength. ACT has a number of ways of adjusting the strength of a production in order to improve performance. Productions have a strength of .1 when first created. Each time it applies, a production has its strength increased

by .025. However, when a production applies and receives negative feedback, its strength is reduced by a factor of 1/4. Since a multiplicative adjustment produces a greater change in strength than an additive adjustment, this "punishment" is much more effective than a reinforcement.

While these two mechanisms are sufficient to adjust the behavior of any fixed set of productions, additional strengthening mechanisms are required to integrate new productions into the behavior of the system. Because these new productions are introduced with low strength, they would seem to be victims of a vicious circle: They cannot apply unless they are strong, and they are not strong unless they have applied. What is required to break out of this circle is a means of strengthening productions that does not rely on their actually applying. This is achieved by taking all of the strength adjustments that are made to a production that applies and making these adjustments to all of its generalizations that are in the system as well. Since a general production will be strengthened every time any one of its (possibly) numerous specializations applies, new generalizations can quickly amass enough strength to extend the range of situations in which ACT performs successfully.

For purposes of strengthening, re-creation of a production that is already in the system, whether by designation, generalization, or discrimination, is treated as equivalent to a successful application in the sense that the re-created production receives a .025 strength

increment and so do all of its generalizations. One implication of this principle is that repetition of instructions has cumulative benefits for performance.

Interaction Between Strength and Specificity

While selection rules based on strength can make some of the required choices among competing productions, it is clear that strength cannot be the sole criterion. For example, people reliably generate irregular plurals (e.g., oxen) under circumstances in which the "add s" rules for regular plurals are presumably also applicable. This reliable performance is obtained despite the fact that the productions responsible for generating regular plurals are applied much more frequently than those for irregulars and therefore should be much stronger. ACT's solution to the problem of exceptions to strong general rules relies on the specificity-ordering principle, ~~discussed earlier~~ to decide which productions on the APPLYLIST should actually execute. This principle accounts for the execution of a production generating an irregular plural since its condition presumably contains all of the requirements for generating ~~a~~ the regular plural and must, in addition, make reference to the specific noun to be pluralized.

The precedence of exceptions over much stronger general rules does not imply that exceptions are impervious to feedback, however. In order to benefit from the specificity ordering principle exceptions must first have achieved the amount of strength necessary to be placed on the APPLYLIST. Furthermore, because this amount depends on the strengths of

the other productions that could apply, the stronger a general rule is, the more strength its exceptions need in order to apply reliably. But exceptions are designated with such low strength that one of the two mechanisms that can strengthen productions that have not actually applied must rescue them if they are ever to come to apply reliably. Since it is unlikely that a newly-designed exception is a generalization of any existing productions, inheriting the strengthenings given to specializations is not a solution in this case. Instead, repeated designations of the exception can provide the initial strength required for occasional placement on the APPLYLIST. Once this is achieved, a series of successful applications will be enough to produce consistent execution of the exception instead of the general rule.

The following example, which shows ACT learning to refer to objects with definite and indefinite articles, illustrates this interaction between strength and specificity. The example begins with ACT in the situation of a young child who knows how to refer to objects with nouns, but who does not yet know how to modify them with articles. ACT's knowledge here takes the form of the production:

P57: IF the goal is to refer to LVobj
and LVc is the concept for LVobj
and LVword is the word for LVc
THEN say LVword

By some unspecified process, ACT forms the general hypothesis that the speaker's choice of article is determined by the listener's

relation to the object being referred to. This hypothesis also takes the form of a production:

P58: IF GVmodel is referring to LVobj
and LVc is the concept for LVobj
and LVword is the word for LVc
and the listener has LVrelation to LVobj
and the model says "LVword1 LVword"

THEN BUILD

```
IF the goal is to refer to 'LVobj'
    and 'LVc' is the concept for 'LVobj'
    and 'LVword' is the word for 'LVc'
    and the LVlistener has LVrelation
      to 'LVobj'
THEN say "LVword1 'LVword'"
```

(In the process of designating new productions, P58 will substitute variables--see p.xx above--for the items in single quotes.) Whenever there are new data relevant to this general hypothesis about the dependence between the speaker's choice of article and the state of the listener, P58 designates a production to embody the specific hypothesis supported by these new data. In particular, one of the productions P59 or P60 will be designated by P58 on almost every occurrence of articles in adult speech:[7]

P59: IF the goal is to refer to LVobj
and LVc is the concept for LVobj
and LVword is the word for LVc

and the listener is aware of LVobj

THEN say "THE LVword"

P60: IF the goal is to refer to LVobj

and LVc is the concept for LVobj

and LVword is the word for LVc

and the listener is unaware of LVobj

THEN say "A LVword"

The conditions of P59 and P60 are both supersets of the condition of P57. Therefore, if either one of these productions which use articles in referring is on the APPLYLIST it will apply instead of P57, which only uses nouns, by the specificity-ordering principle.

Once ACT has the designating production P58, the course of learning may be observed in which a training trial consists of providing an example of reference using articles. The amount of learning that has occurred can be assessed with test trials, produced by entering propositions into the data base that satisfy the productions that have been designated. There must be a proposition to the effect that ACT has the goal of referring to an object. There must also be a statement about the listener's awareness/unawareness of the object in question. For example, if the listener is said to be aware of the dog that ACT wants to refer to, either production P57 will apply generating dog or production P59 will apply generating the dog (errors like a dog were not possible in this simulation). The proportion of test trials on which an article is used is a measure of ACT's learning.

The details of the simulation were as follows: Production P57 which refers without articles was given an initial strength of 20. The designating production P58 was given a strength of only .1 reflecting the fact that it is a new hypothesis about articles. Training trials alternated with test trials and definite articles alternated with indefinite articles. Thus a series of four trials had the form: train with definite article, test use of definite article, train with indefinite article, test use of indefinite article. A complete simulation of learning to use articles required ten such blocks of four trials. (ACT undoubtedly learns too rapidly to be an accurate model of humans; however, the computational expense of a more accurate simulation would be prohibitive.) Ten replications were performed of the complete simulation in order to obtain proportions of article use in each block. The course of learning was different in each replication because of the probabilistic nature of production selection.

In qualitative terms the results of the simulations were as follows: On the first few training trials the designating production P58 applied unreliably due to its low strength; even when it did apply, the productions it designated, (P59 and P60) were too weak themselves to apply reliably on test trials. However, when P58 did manage to apply, it was strengthened, resulting in more reliable designation on subsequent training trials; this led, in turn, to the strengthening of P59 and P60. The combined strengthening influences of frequent re-designation and successful application were enough to produce reliable generation of articles by the end of the simulation.

Insert Figure 2 about here

The results are shown in quantitative terms in Figure 2. There is a relatively rapid, but not all-or-none, change in the level of performance. The best and the worst simulations show much the same pattern as the average of all ten. These rapid changes can be explained by the tendency for success to feed on itself in ACT. A successful execution of a production results in an increase in its strength and consequently greater opportunity for further execution and strengthening. Roger Brown (1973) reports that young children show just these sharp, but not all-or-none, changes in their percentage of correct use of grammatical morphemes.

Designation Takes Precedence Over Strength

An argument can be made for adding yet another principle of production selection to those already operative in the previous example. For several cycles following the designation of a production, an attempt should be made to apply this new production before applying any of the productions on the APPLYLIST.[8] This principle allows us to explain the fact that, with some effort, it is possible for adults to deliberately override highly overpracticed rules. For instance, it is possible to replace the "add s" rule for pluralizing nouns with an "add er" rule (e.g., three booker). The explanation runs as follows: The production which implements the "add er" rule is repeatedly designated as long as a deliberate effort is being made to perform the new pluralization. By

virtue of having been just designated, it is applied in preference to the "add s" rule. When the deliberate effort is no longer maintained, designation ceases, the "add er" production fails to be placed on the APPLYLIST because of its low strength, and the strong "add s" rule re-asserts itself.

The results of some experiments by LaBerge (1974) have a similar explanation involving the precedence of designation over strength. LaBerge had subjects make same-different judgements for familiar alphabetic symbols and for unfamiliar letter-like symbols. Reaction time in this task can be thought of as determined by the number of cycles required to select the relevant productions. This quantity will be inversely related to the strengths of the productions unless designation causes automatic selection. Since alphabetic symbols presumably have very strong productions responsible for their recognition, the reaction-time advantage usually found for these symbols can be explained as due to strength differences. However, when subjects knew ahead of time what symbol would be involved in the judgement, there was no advantage for the familiar symbols. This can be explained as due to the automatic selection of productions designated to recognize the expected symbol.

Discrimination by Restriction vs. Discrimination by Exception

There is an important distinction to be made in ACT between two types of discrimination, only one of which can be formed by automatic-discrimination. ACT's automatic-discrimination mechanism cannot form an exception to a general rule because the exception would need a different

action. Productions with new actions can only be formed by designation. The automatic discrimination mechanisms merely modify the range of situations in which an existing action will be performed--that is, they correct over-generalizations of that action. This we call discrimination by restriction to distinguish it from the discrimination by exception required in the pluralization example of the previous section.

It is interesting to compare the ways in which exceptions and restrictions are integrated into the behavior of the system. First, consider the similarities: We have seen previously that after being designated with low strength initially repeated re-designation allows exceptions to accumulate the strength required for occasional placement on the APPLYLIST. Nothing prevents the automatic discrimination mechanism from choosing, on different occasions, the same proposition from the data base to use in forming new productions. Thus, in all likelihood, the same restriction of an overgeneral production will be formed multiple times; therefore just as is the case with exceptions, it is possible for multiple formations to provide the strength necessary for placing restrictions on the APPLYLIST. Once ~~consistent~~ occasional placement on the APPLYLIST is achieved, a history of successful applications will increase the strength of both exceptions and restrictions to the point where they will apply reliably in the future.

However, interesting differences between exceptions and restrictions emerge when we consider circumstances in which these discriminations do not apply. When an exception is not applicable its

general rule will take over and presumably be strengthened for correct performance. The intention is that both the exception and the general rule should co-exist in the system and, in fact, as long as occasions to apply the exception are frequent enough, neither will grow in strength at the expense of the other.

On the other hand, assuming that our restriction is the right one (i.e., its action is called for in just those situations described in its condition) whenever this restriction is not applicable any application of its overgeneral source results in errors. These errors will presumably be punished, costing the overgeneralization 1/4 of its strength each time. Here the intention is that the correct restriction should come to replace its overgeneral source in the operation of the system, and, in fact, the restriction grows rapidly in strength relative to its source. It can be strengthened in all situations in which its source is strengthened, but it avoids all punishment the source receives for misapplication.

It is relative loss of strength of the source that is important here. Since production selection evaluates the strength of a production relative to the strengths of all productions with active constants, a production will be selected for the APPLYLIST with a probability of 1.0, regardless of its strength on occasions in which it is the only active production. This implies that negative feedback would not be effective in the Act system if it only reduced strength and did not also result in the creation of competing productions through automatic discrimination. On this issue, ACT is supported by the learning literature (Estes, 1970;

Hilgard & Bower, 1974) which indicates that ~~punishment~~ negative feedback works not so much by "stamping out" behaviors as by producing alternative behaviors.

Another prediction that follows from the ACT model is that negative feedback should play an important role in the learning of any complex procedure, since without it the automatic discrimination mechanism cannot operate. This prediction is in direct conflict with the widespread belief that negative feedback is completely ineffectual in first-language acquisition. For example, Cazden (1964) has reported that providing children with corrected versions of their ungrammatical utterances does not result in more rapid acquisition of the correct forms. If this claim is accurate (and there is some evidence that it is not; see McNeil, 1970), then it can only be explained in ACT terms by assuming that the children were for some reason incapable of determining just which productions should have been punished from the negative feedback that was provided.

The Origin of Designating Productions

Although procedural learning involves the acquisition of new behaviors, as noted above, ACT's automatic generalization and discrimination mechanisms cannot add new actions to productions. The designation process is thus indispensable to the ACT theory of procedural learning because it alone has the ability to introduce productions with new actions into the system. Once this is appreciated, it becomes necessary to account for the acquisition of the designating productions themselves. In our work to date, the only requirement we placed on

ourselves in proposing designating productions for ACT in learning some skill was that a human learner of that same skill might plausibly possess the knowledge incorporated in those productions. Given our interest in the learning of complex procedures, this seemed like a good strategy since it would be very difficult to give any detailed account of the origins of the sophistication that is demanded from the learner of any complex procedure. Of course, this is only defensible as a short-term strategy--the ACT learning theory is distressingly incomplete as long as the origin of designating productions is unexplained. The function of the present section is to present some speculations on the origin of ACT's designating productions.

Experience can always be expected to function, in at least a crude way, to recommend certain new behaviors; it would be reasonable for ACT to start out already having designating productions that capitalize on this expectation. For example, we saw the following modeling production earlier (p. xx):

P24: IF when LVmodel sees LVevent1
another event, LVevent2, occurs
consisting of LVmodel doing LVaction

THEN build
```
IF LVevent1
THEN LVevent2
```
substituting ACT for all
occurrences of LVmodel

Actions performed by models in various situations have a high likelihood of being appropriate for ACT in those situations as well, and this makes P24 a good candidate for membership in the set of original designating productions. Other candidates for this set are inspired by the principles of traditional learning theory. For example, there is production P61 of Table 8 which incorporates a reinforcement principle.

Table 8 about here

Now it might appear that Production P61 is useless for producing new behaviors since it requires that ACT has already performed the behavior in question. However, in conjunction with a mechanism that randomly generated all the behaviors that ACT is capable of, P61 would enable a reinforced behavior to be incorporated into a production where it could be performed under stimulus control for the first time. A (rather anthropomorphic) example would have ACT reinforced for accidentally saying mama when its mother is near. The following production would be designated which represents a modest, but necessary, step towards the lexicalization of natural language; i. e., it introduces a connection between the word mama and the concept @Mommy:

P66: IF ACT sees @Mommy
THEN ACT say "mama"

Alternatively, the environment can act in a highly directive way to produce a passive action on ACT's part; as, for example, when an adult

takes a child's hand and makes it go through the motions required to tie a shoe. Production P61 would allow ACT to produce such behaviors on its own subsequently.

It is just possible that original designating productions of these sorts, in combination with the automatic generalization and discrimination mechanisms, is all the "innate endowment" that ACT requires to account for human procedural learning. The remainder of this section will attempt to provide support for this possibility by demonstrating that one of the designating productions required to comprehend verbal instructions can be formed from generalizations and discriminations of some original designating productions. The original designating productions that will be used are the reinforcement production P61 and the ~~predictive~~ production P63 from Table 8. Production P63 designates new productions that predict the consequences of ACT's behavior. These new productions will apply whenever that behavior is performed in the future and will predict the same consequences that were obtained previously. The automatic discrimination mechanism can form two new productions, P62 and P64 in Table 8, from the original designating productions P61 and P63. Both of these discriminations result from ACT's observation that occasions on which useful designations are formed are often those on which teachers use a particular kind of sentence (if-then) that refers to the events involved in the designation.[9]

Once these two discriminations have been formed, a generalization over them produces the designating production P65 in Table 8 which is

responsible for comprehending verbal instructions. Thus by processes of discrimination and generalization two designating productions that record events surrounding ACT's own actions ultimately give rise to a designating production of a very different character. Our hope is that all of the designating productions ACT requires for procedural learning can be produced in this same manner.

Future Directions: Inspection of Productions

Currently, one ACT production cannot inspect the contents of another ACT production because the productions themselves are not represented in the data base. As a consequence it is impossible to use productions to analyze the procedures that ACT has available for performing some task in order to isolate and correct "bugs" in those procedures. The idea that procedural learning consists of a debugging process has motivated a great deal of recent work in cognitive science (Brown, Burton, Hausman, Goldstein, Huggins & Miller, 1977; Goldstein, 1974; Sussman, 1975). While we think that debugging processes require too much domain-specific knowledge to account for much of human procedural learning, it is undeniable that experts can analyze the procedures they are using to find and correct bugs. An example comes from our experiences in learning to program in the language C, where all indexing initiates at 0 rather than the more customary 1. Introspection suggests that this requires systematically reworking familiar procedures for searching arrays, looping, etc. to compensate for this unusual convention. To make it possible to model such debugging processes we

intend to modify the ACT system to allow productions to treat other productions as data; that is, to allow productions to test for the existence of various other kinds of productions and, upon finding them, to add to them or make other modifications.

While the primary motivation for this change is to expand ACT's learning capabilities, it appears that making productions inspectable will provide benefits for the non-learning (performance) aspects of the system as well. One expected benefit is that it should become easier for ACT to direct its behavior in service of its goals. For example, in the LISP-learning simulation discussed earlier(p. xx), we had a production for categorizing a sequence of symbols as a dotted-pair:

P23: IF an expression begins with a @LEFT-PARENTHESIS
and this @LEFT-PARENTHESIS is before an @S-EXPRESSION
and this @S-expression is before a @DOT
and this @DOT is before an @S-EXPRESSION
and this @S-EXPRESSION is before a @RIGHT-PARENTHESIS
THEN it is a @DOTTED-PAIR

Notice that this production depends on the subsequences having already been categorized as S-expressions; that is, it assumes a bottom-up sequence of processing where all decisions about high-level constituents must wait on decisions about all low-level constituents. The difficulty with this scheme is that the failure of a single production to apply--due, to low strength or to a failure to spread activation to all of the required memory structure--holds up the entire sequence of processing.

In addition, there is a great deal of wasted effort because low-level categorizations are made without regard to their usefulness for deciding between the various high-level categorizations that are viable at the moment.

Giving productions the ability to inspect other productions makes it possible to implement a top-down scheme that avoids some of these difficulties. Productions will respond to the top-level goal of showing that a particular expression is a dotted-pair by searching for other productions that make this categorization as part of their action. This search will find production P23 and then productions will notice that the condition of P23 can be satisfied if there are S-expressions on both sides of the dot. This leads, in turn, to a search for productions that categorize symbol sequences as S-expressions and the entire process repeats itself until a production is found whose condition is satisfied, but which has not yet applied. If it is low strength that has prevented this production from applying previously, then re-designating it will enable it to apply now. Alternatively, since the process of finding this production involved focusing the system's attention on successively smaller constituents of the dotted-pair, this refocusing can be expected to activate any memory structure whose inactivity blocked the application of this production previously. In any case the ability to implement this top-down process should result in more reliable achievement of the system's goals.

It is generally acknowledged that the design of a performance

system will have strong influences on the learning system. That is, our learning principles will be strongly influenced by our conception of what the end product of the learning process is like. On the other hand, it is also the case, as just illustrated, that work with a learning theory will affect the performance theory. There is a complex and intimate relationship between the two. It is preferable, and fortunately it is possible for us, to pursue both endeavors in parallel.

References

Anderson, J.R. Language acquisition by computer and child. Technical Report No. 55, Human Performance Center, University of Michigan, 1974.

Anderson, J.R. Computer simulation of a language acquisition system: A first report. In R.L. Solso (Ed.), Information Processing and Cognition: The Loyola Symposium, Hillsdale, N.J.: Erlbaum Assoc., 1977.

Anderson, J.R. Language, memory, and thought. Hillsdale, N.J.: Erlbaum Assoc., 1976.

Anderson, J.R. Induction of augmented transition networks. Cognitive Science, 1977, 1, 125-158.

Anderson, J.R. Computer simulation of a language acquisition system: A second report. In D. LaBerge & S.J. Samuels (Eds.), Perception and comprehension, Hillsdale, N.J: Erlbaum Assoc., 1978.

Anderson, J.R., Kline, P. and Lewis, C. A production system

model for language processing. In P. Carpenter & M. Just (Eds.), Cognitive Processes in Comprehension. Hillsdale, N.J.: Erlbaum Assoc., 1977.

Bailey, N., Madden, C., & Krashen, S.D. Is there a "natural sequence" in adult second language learning? English Language Institute and Linguistics Department, Queens College and the Graduate Center, City University of New York, 1974.

Brown, J.S., Burton, B.R., Hausmann, C., Goldstein, I., Huggins, B. Miller, M. Aspects of a theory for automated student modelling. BBN Report No. 3549, 1977.

Brown, R. A first language. Cambridge, Mass.: Harvard University Press, 1973.

Cazden, C.G. Environmental assistance to the child's acquisition of grammar. Unpublished Ph.D. dissertion, Harvard University, 1965.

Chomsky, N. Aspects of the theory of syntax. Cambridge, Mass.: MIT Press, 1965.

Estes, W.K. Learning theory and mental development. New York: Academic Press, 1970.

Goldstein, I.P. Understanding simple picture programs. AI-TR-294. MIT A.I. Laboratory, 1974.

Hayes-Roth, F. The role of partial and best matches in knowledge systems, Rand Corporation, P-5802, 1977.

Hayes-Roth, F. & McDermott, J. Learning structured patterns from examples. Proceedings of the Third International Joint Conference on Pattern Recognition, 1976, 419-423.

Hilgard, E.R. & Bower, G.H. Theories of learning, New York: Appleton-Century-Crofts, 1966.

LaBerge, D. Attention and the measurement of perceptual learning. Memory and Cognition, 1973, 1, 268-276.

McNeil, D. The acquisition of language. New York: Harper & Row, 1970.

Newell, A. A theoretical exploration of mechanisms

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

Newell, A. Production systems: Models of control structures. In W.G. Chase (Ed.), Visual Information Processing. New York: Academic Press, 1973.

Rychener, M.D. & Newell, A. An instructible production system: Basic design issues. In D.A. Waterman & F. Hayes-Roth (Eds.), Pattern directed inference systems, New York: Academic Press, in press.

Sussman, G.J. A computer model of skill acquisition. New York: American Elsevier Publishing Co., 1975.

Vere, S.A. Induction of concepts in the predicate calculus. Proceedings of the Fourth International Joint Conference on Artificial Intelligence, Tbilisi, USSR, 1975, 281-287.

Vere, S.A. Induction of relational productions in the presence of background information. Proceedings of the Fifth International Joint Conference on

Artificial Intelligence, Boston, 1977.
349-355 (b).

Vere, S.A. Inductive learning of relational productions. In D.A. Waterman & F. Hayes-Roth (Eds.), Pattern directed inference systems, New York: Academic Press, in press.

Weissman,C. LISP 1.5 primer, Belmont, Ca.: Dickenson, 1967.

Footnotes

1. This research is supported by grants N00014-77-C-0242 from the Office of Naval Research and NIE-G-77-0005 from the National Institute of Education.

2. The version of ACT described in this paper is called ACTF. Earlier publications (e.g., Anderson, 1976) described the previous version, ACTE.

3. To simplify the exposition a relation-argument syntax for propositions is used in this paper. This is a departure from the actual ACTF syntax which relies on infix operators such as * and OF as described in previous publications (see Anderson, 1976; Anderson, Kline & Lewis, 1977). Also in the interests of simplicity, type-token distinctions required to properly represent several occurrences (tokens) of the same digit (type) in an addition problem are being ignored here and throughout the paper.

4. No discussion of link strength will be provided here. Similarly, the whole question of decay of activation is being ignored. A more complete treatment of spreading activation can be found in Anderson, 1976, Ch. 8 -- although the current ACTF implementation of the spreading activation process differs substantially from the implementation discussed there.

5. The productions presented in this paper are translations of the formal syntax of the implemented productions into (hopefully) more readable prose. The reader interested in the details may request listings of the implemented versions and examples of their operation.

6. These productions and some others in this paper embody some clearly over-simplified notions about language comprehension; a more adequate treatment would only distract attention from the learning processes which are the matters of present interest, however. For a discussion of language processing within the ACT framework see Anderson, Kline, and Lewis (1977). (One complication necessary to any complete analysis of language comprehension is, nevertheless, being observed in some of the examples in this paper--the distinction between words and the concepts underlying them.)

7. Choice of article is more complicated than implied here; see Brown (1973, pp. 340-350) for a ~~partial~~ discussion.

8. In the simulations discussed in the previous section, the ability to apply various productions was used to assess the amount of procedural learning that ACT had accomplished. However, the principle being proposed now means that a production that applies easily after designation might be very difficult to apply later on. For this reason, these earlier simulations were run without giving preference to designated productions in order to get the most accurate estimates of learning.

9. Actually, at present there is no way to punish the designating productions, as is required to produce these discriminations. First of all, they have as their actions the creation of new productions--wholly internal events that could never be evaluated by an outside observer. Secondly, neither the internal event of designation nor the

new production that results from that event can currently be evaluated by the production system itself, because they are not represented in the data base that the system has available for inspection. However, as will be discussed in the next section, we anticipate expanding ACT's data base to take care of this difficulty.

Table 1

A Set of Productions for Adding Two Numbers

P1: IF the goal is to add LVnumber1 and LVnumber2
and LVnumber1 begins with LVdigit1
and LVnumber2 begins with LVdigit2
THEN add LVdigit1 and LVdigit2
and set GVdigit1 to LVdigit1
and set GVdigit2 to LVdigit2

P2: IF GVdigit1 and GVdigit2 are being added
and LVsum is the sum of GVdigit1 and GVdigit2
THEN set GVsum to LVsum

P3: IF GVdigit1 and GVdigit2 are being added
and LVsum is the sum of GVdigit1 and GVdigit2
and there is a carry
THEN set GVsum to LVsum

P4: IF GVsum has a value
and there is no carry
and GVsum is not > 9
THEN write GVsum
and go to the next column

P5: IF GVsum has a value
and there is a carry
and LVsum1 is the sume of GVsum plus 1
and LVsum1 is not > 9
THEN write LVsum1
and go to the next column

P6: IF GVsum has a value
and GVsum > 9
and GVsum is the sum of LVdigit3 and 10
and there is no carry
THEN write LVdigit3
and go to the next column with a carry

P7: IF GVsum has a value
and there is a carry
and GVsum > 8
and GVsum is the sum of LVdigit3 and 9
THEN write LVdigit3
and go to the next column with a carry

P8: IF sent to the next column with no carry
and there is a digit, LVdigit3, after GVdigit1
and a digit, LVdigit4, after GVdigit2
THEN set GVdigit1 to LVdigit3
and set GVdigit2 to LVdigit4
and add GVdigit1 and GVdigit2

P9: IF sent to the next column with no carry
and there is a digit, LVdigit3, after GVdigit1
and there is no digit after GVdigit2
THEN set GVdigit1 to LVdigit3
and write GVdigit1
and go to the next column

P10: IF sent to the next column with no carry
and there is a digit, LVdigit4, after GVdigit2
and there is no digit after GVdigit1
THEN set GVdigit2 to LVdigit4
and write GVdigit2
and go to the next column

P11: IF sent to the next column
and there is no digit after GVdigit1
and there is no digit after GVdigit2
THEN problem completed

P12: IF sent to the next column with a carry
and there is a digit, LVdigit3, after GVdigit1
and a digit, LVdigit4, after GVdigit2
THEN set GVdigit1 to LVdigit3
and set GVdigit2 to LVdigit4
and add GVdigit1 and GVdigit2
and note the carry in the new column

P13: IF sent to the next column with a carry
and there is a digit, LVdigit3 after GVdigit1
and there is no digit after GVdigit2
and LVdigit3 is not = 9
and LVsum is the sum of LVdigit3 and 1
THEN set GVdigit1 to LVdigit3
and write LVsum
and go to the next column

P14: IF sent to the next column with a carry
and there is a digit, LVdigit4, after GVdigit2
and there is no digit after GVdigit1
and LVdigit4 is not = 9
and LVsum is the sum of LVdigit4 and 1
THEN set GVdigit2 to LVdigit4
and write LVsum
and go to the next column

P15: IF sent to the next column with a carry
and there is a digit, LVdigit3, after GVdigit1
and there is no digit after GVdigit2
and LVdigit3 = 9
THEN set GVdigit1 to LVdigit3
and write 0
and go to the next column with a carry

P16: IF sent to the next column with a carry
and there is a digit, LVdigit4, after GVdigit2
and there is no digit after GVdigit1
and LVdigit4 = 9
THEN set GVdigit2 to LVdigit4
and write 0
and go to the next column with a carry

P17: IF sent to the next column with a carry
and there is no digit after GVdigit1
and there is no digit after GVdigit2
THEN write 1
and problem completed

Table 2

A Set of Productions for Encoding

Rules About LISP Structures

P18: IF there is a sentence beginning

"If an expression is a LVword...",

where LVconcept is the concept for LVword

THEN save IF there is a LVconcept for a new

condition by attaching it to GVhold

and set GVword to LVword

P19: IF there is a sentence ending

"...GVword it is a LVword,"

where LVconcept is the concept for LVword

THEN BUILD:

IF GVhold
THEN it is a LVconcept

P20: IF there is a sentence beginning

"If an expression begins with a LVword..."

where LVconcept is the concept for LVword

THEN save IF an expression begins with an LVconcept

for a new condition by attaching it to GVhold

and set GVword to LVword

and set GVconcept to LVconcept

P21: IF a sentence has a phrase

"...GVword followed by a LVword..."

where LVconcept is the concept for LVword

THEN save IF there is a GVconcept before a LVconcept

for a new condition by attaching it to GVhold

and set GVconcept to LVconcept

and set GVword to LVword

Table 3

The Productions Involved in Learning Two Possible

Sentence Structures for the Verb Gave

P26: IF the LVteacher says an LVsentence

while pointing to an LVevent

THEN BUILD

```
IF LVsentence
THEN LVevent
```

P27: IF there is a sentence "John gave the ball to Jane"

THEN understand from this sentence that

John caused a change in the possession

of the ball from John to Jane

P28: IF there is a sentence "Bill gave the dolly to Mary"

THEN understand from this sentence that

Bill caused a change in the possession

of the dolly from Bill to Mary

P29: IF there is a sentence "LVagent gave the LVobject to LVrecipient"

THEN understand from this sentence that

LVagent caused a change in the possession

of the LVobject from LVagent to LVrecipient

P30: IF there is a sentence "Mary gave to John the ball"
THEN understand from this sentence that
Mary caused a change in the possession
of the ball from Mary to John

P31: IF there is a sentence "Bill gave to Jane the dolly"
THEN understand from this sentence that
Bill caused a change in the possession
of the dolly from Bill to Jane

P32: IF there is a sentence "LVagent gave to LVrecipient the LVobject"
THEN understand from this sentence that
LVagent caused a change in the possession
of the LVobject from LVagent to LVrecipient

Table 4

A Set of Productions for Learning the Syntactic Structure of Simple Agent-Verb-Object Sentences

P36: IF GVmodel begins a LVsentence
with a LVmorpheme which is used
to refer to objects in LVcategory
THEN save the proposition
LVmorpheme refers to LVcategory
by attaching it to GVrelations
and set GVmorpheme to LVmorpheme
and set GVsentence to LVsentence

P37: IF GVmodel begins a LVsentence
with a LVmorpheme which is not known to
refer to any LVcategory
THEN set GVmorpheme to LVmorpheme
and set GVsentence to LVsentence

P38: IF in GVsentence GVmorpheme is followed
by LVmorpheme which is used to
refer to objects in LVcategory
THEN save the proposition
LVmorpheme refers to LVcategory
by attaching it to GVrelations
and set GVmorpheme to LVmorpheme

P39: IF in GVsentence GVmorpheme is followed by
LVmorpheme which is not known to
refer to any category
THEN set GVmorpheme to LVmorpheme

P40: IF the GVsentence ends with GVmorpheme
and the GVmodel uses this sentence
to describe the event of some number
of LVagents LVacting on some number of
LVobjects at LVtime

THEN BUILD:

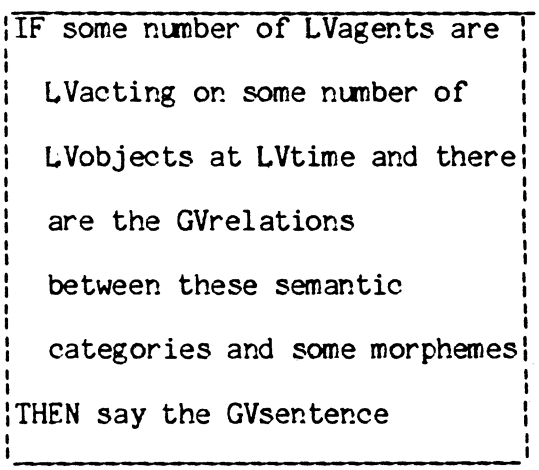


Table 5

An Over-General Set of Productions for Generating Agent-Verb-Object Sentences Referring to Singular or Plural Subjects in Either Present or Past Tense

P43: IF LVagent happens to LVact on LVobject at LVtime
THEN say "The LVagent LVact+s the LVobject"

P44: IF LVagent happens to LVact on LVobject at LVtime
THEN say "The LVagent+s LVact the LVobject"

P45: IF LVagent happens to LVact on LVobject at LVtime
THEN say "The LVagent LVact+ed the LVobject"

P46: IF LVagent happens to LVact on LVobject at LVtime
THEN say "The LVagent+s LVact+ed the LVobject"

Table 6

Correctly Discriminated Versions

of the Productions in Table 5

P47: IF LVagent happens to LVact on LVobject at LVtime
and LVagent is singular
and LVtime is present
THEN say "The LVagent LVact+s the LVobject"

P48: IF LVagent happens to LVact on LVobject at LVtime
and LVagent is plural
and LVtime is present
THEN say "The LVagent+s LVact the LVobject"

P49: IF LVagent happens to LVact on LVobject at LVtime
and LVagent is singular
and LVtime is past
THEN say "The LVagent LVact+ed the LVobject"

P50: IF LVagent happens to LVact on LVobject at LVtime
and LVagent is plural
and LVtime is past
THEN say "The LVagent+s LVact+ed the LVobject"

Table 7

A Categorization by Number and Tense of the Situations in Which Four Discriminate Productions Can Apply

	Singular	Plural
Present	P53	P48, P54
Past	P49, P53	P54

Table 8

Two Innate Designating Productions and Some Discriminations and Generalizations Derived From Them

P61: IF LVevent occurs just before ACT performs LVaction which is followed by reinforcement

THEN BUILD

```
IF LVevent
THEN LVaction
```

P62: IF LVevent occurs just before ACT performs LVaction which is followed by reinforcement and a teacher has said "If LVclause1 then LVclause2" and LVevent is the meaning of LVclause1 and LVaction is the meaning of LVclause2

THEN BUILD

```
IF LVevent
THEN LVaction
```

P63: IF ACT performs LVaction which is followed by LVeffect

THEN BUILD

```
IF LVaction
THEN LVeffect
```

P64: IF ACT performs LVaction which is followed by LVeffect

and a teacher has said

"If LVclause1 then LVclause2"

and LVaction is the meaning of LVclause1

and LVeffect is the meaning of LVclause2

THEN BUILD

```
IF LVaction
THEN LVeffect
```

P65: IF a teacher has said

"If LVclause1 then LVclause2"

and LVcondition is the meaning of LVclause1

and LVaction is the meaning of LVclause2

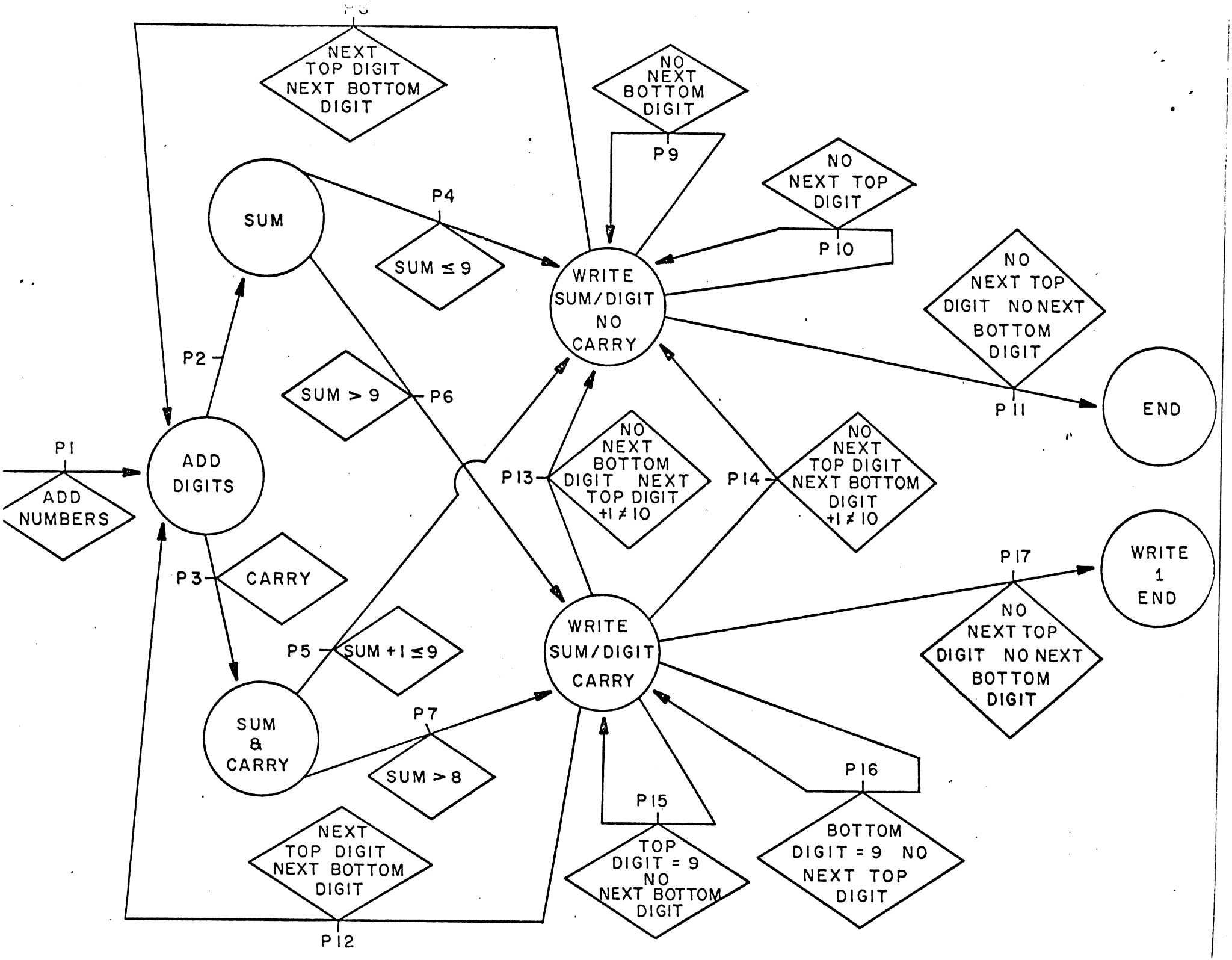
THEN BUILD

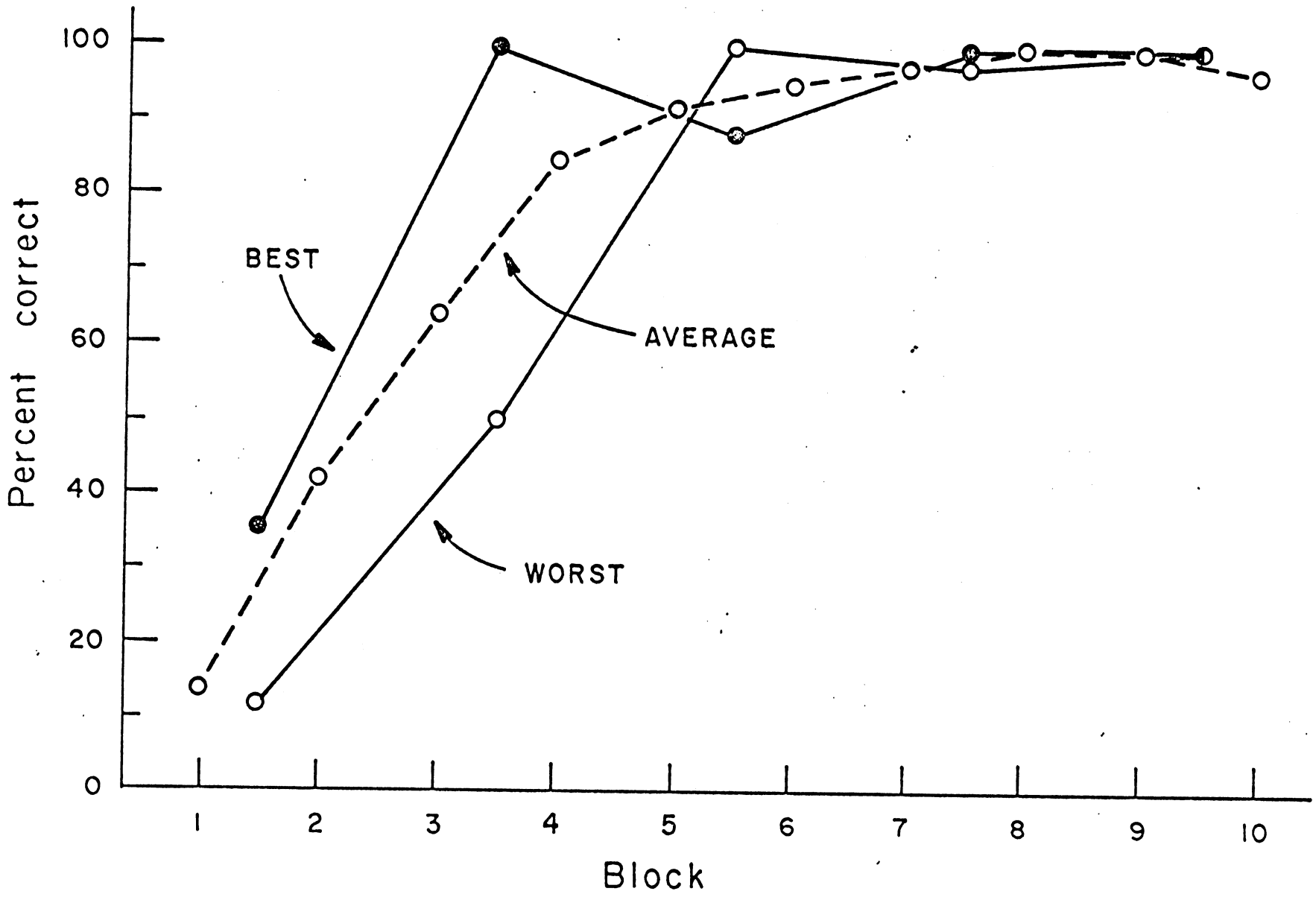
```
IF LVcondition
THEN LVaction
```

Figure Captions

1. The flow of control among the productions in Table 1.

2. The increase with time of ACT's use of the articles a and the. Successive blocks are averaged in reporting the best and worst individual runs.


P8
NEXT TOP DIGIT NEXT BOTTOM DIGIT
NO NEXT BOTTOM DIGIT
P9
NO NEXT TOP DIGIT
P10
P4
SUM
SUM ≤ 9
WRITE SUM/DIGIT NO CARRY
NO NEXT TOP DIGIT NO NEXT BOTTOM DIGIT
P11
END
P2
SUM > 9
P6
P1
ADD NUMBERS
ADD DIGITS
P13
NO NEXT BOTTOM DIGIT NEXT TOP DIGIT +1 ≠ 10
P14
NO NEXT TOP DIGIT NEXT BOTTOM DIGIT +1 ≠ 10
P3
CARRY
P17
WRITE 1 END
NO NEXT TOP DIGIT NO NEXT BOTTOM DIGIT
P5
SUM +1 ≤ 9
WRITE SUM/DIGIT CARRY
SUM & CARRY
P7
SUM > 8
P16
P15
NEXT TOP DIGIT NEXT BOTTOM DIGIT
TOP DIGIT = 9 NO NEXT BOTTOM DIGIT
BOTTOM DIGIT = 9 NO NEXT TOP DIGIT
P12


Percent correct
100
80
60
40
20
0
BEST
AVERAGE
WORST
1
2
3
4
5
6
7
8
9
10
Block

Navy

4 DR. JACK ADAMS
OFFICE OF NAVAL RESEARCH BRANCH
223 OLD MARYLEBONE ROAD
LONDON, NW, 15TH ENGLAND

1 Dr. Jack R. Borsting
Provost & Academic Dean
U.S. Naval Postgraduate School
Monterey, CA 93940

1 DR. JOHN F. BROCK
NAVY PERSONNEL R& D CENTER
SAN DIEGO, CA 92152

1 DR. MAURICE CALLAHAN
NODAC (CODE 2)
DEPT. OF THE NAVY
BLDG. 2, WASHINGTON NAVY YARD
(ANACOSTIA)
WASHINGTON, DC 20374

1 Dept. of the Navy
CHNAVMAT (NMAT 034D)
Washington, DC 20350

1 Chief of Naval Education and
Training Support)-(01A)
Pensacola, FL 32509

1 CAPT. H. J. CONNERY
NAVY MEDICAL R&D COMMAND
NNMC
BETHESDA, MD 20014

1 Dr. Charles E. Davis
ONR Branch Office
536 S. Clark Street
Chicago, IL 60605

1 Mr. James S. Duva
Chief, Human Factors Laboratory
Naval Training Equipment Center
(Code N-215)
Orlando, Florida 32813

5 Dr. Marshall J. Farr, Director
Personnel & Training Research Programs
Office of Naval Research (Code 458)
Arlington, VA 22217

Navy

1 DR. PAT FEDERICO
NAVY PERSONNEL R&D CENTER
SAN DIEGO, CA 92152

1 CDR John Ferguson, MSC, USN
Naval Medical R&D Command (Code 44)
National Naval Medical Center
Bethesda, MD 20014

1 Dr. John Ford
Navy Personnel R&D Center
San Diego, CA 92152

1 Dr. Eugene E. Gloye
ONR Branch Office
1030 East Green Street
Pasadena, CA 91101

1 CAPT. D.M. GRAGG, MC, USN
HEAD, SECTION ON MEDICAL EDUCATION
UNIFORMED SERVICES UNIV. OF THE
HEALTH SCIENCES
6917 ARLINGTON ROAD
BETHESDA, MD 20014

1 CDR Robert S. Kennedy
Naval Aerospace Medical and
Research Lab
Box 29407
New Orleans, LA 70189

1 Dr. Norman J. Kerr
Chief of Naval Technical Training
Naval Air Station Memphis (75)
Millington, TN 38054

1 Dr. James Lester
ONR Branch Office
495 Summer Street
Boston, MA 02210

1 Dr. William L. Maloy
Principal Civilian Advisor for
Education and Training
Naval Training Command, Code 00A
Pensacola, FL 32508

Navy

1 Dr. Sylvia R. Mayer (MCIT)
HQ Electronic Systems Div.
Hanscom AFB
Bedford, MA 01731

1 Dr. James McBride
Code 301
Navy Personnel R&D Center
San Diego, CA 92152

2 Dr. James McGrath
Navy Personnel R&D Center
Code 306
San Diego, CA 92152

1 DR. WILLIAM MONTAGUE
NAVY PERSONNEL R& D CENTER
SAN DIEGO, CA 92152

1 Commanding Officer
Naval Health Research
Center
Attn: Library
San Diego, CA 92152

1 CDR PAUL NELSON
NAVAL MEDICAL R& D COMMAND
CODE 44
NATIONAL NAVAL MEDICAL CENTER
BETHESDA, MD 20014

Navy Personnel R&D Center
San Diego, CA 92152

6 Commanding Officer
Naval Research Laboratory
Code 2627
Washington, DC 20390

1 OFFICE OF CIVILIAN PERSONNEL
(CODE 26)
DEPT. OF THE NAVY
WASHINGTON, DC 20390

1 JOHN OLSEN
CHIEF OF NAVAL EDUCATION &
TRAINING SUPPORT
PENSACOLA, FL 32509

Navy

1 Office of Naval Research
Code 200
Arlington, VA 22217

1 Office of Naval Research
Code 437
800 N. Quincy SStreet
Arlington, VA 22217

1 Scientific Director
Office of Naval Research
Scientific Liaison Group/Tokyo
American Embassy
APO San Francisco, CA 96503

1 SCIENTIFIC ADVISOR TO THE CHIEF
OF NAVAL PERSONNEL
NAVAL BUREAU OF PERSONNEL (PERS OR)
RM. 4410, ARLINGTON ANNEX
WASHINGTON, DC 20370

1 DR. RICHARD A. POLLAK
ACADEMIC COMPUTING CENTER
U.S. NAVAL ACADEMY
ANNAPOLIS, MD 21402

1 Mr. Arnold I. Rubinstein
Human Resoureces Program Manager
Naval Material Command (0344)
Room 1044, Crystal Plaza 5
Washington, DC 20360

1 Library

1 Dr. Worth Scanland
Chief of Naval Education and Training
Code N-5
NAS, Pensacola, FL 32508

1 A. A. SJOHOLM
TECH. SUPPORT, CODE 201
NAVY PERSONNEL R& D CENTER
SAN DIEGO, CA 92152

1 Mr. Robert Smith
Office of Chief of Naval Operations
OP-987E
Washington, DC 20350

Navy

1 Dr. Alfred F. Smode
Training Analysis & Evaluation Group
(TAEG)
Dept. of the Navy
Orlando, FL 32813

1 CDR Charles J. Theisen, JR. MSC, USN
Head Human Factors Engineering Div.
Naval Air Development Center
Warminster, PA 18974

1 W. Gary Thomson
Naval Ocean Systems Center
Code 7132
San Diego, CA 92152

Army

1 HQ USAREUE & 7th Army
ODCSOPS
USAAREUE Director of GED
APO New York 09403

1 DR. JAMES BAKER
U.S. ARMY RESEARCH INSTITUTE
5001 EISENHOWER AVENUE
ALEXANDRIA, VA 22333

1 DR. RALPH CANTER
U.S. ARMY RESEARCH INSTITUTE
5001 EISENHOWER AVENUE
ALEXANDRIA, VA 22333

1 DR. RALPH DUSEK
U.S. ARMY RESEARCH INSTITUTE
5001 EISENHOWER AVENUE
ALEXANDRIA, VA 22333

1 DR. FRANK J. HARRIS
U.S. ARMY RESEARCH INSTITUTE
5001 EISENHOWER AVENUE
ALEXANDRIA, VA 22333

1 Dr. Milton S. Katz
Individual Training & Skill
Evaluation Technical Area
U.S. Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

1 Dr. Harold F. O'Neil, Jr.
ATTN: PERI-OK
5001 EISENHOWER AVENUE
ALEXANDRIA, VA 22333

1 Director, Training Development
U.S. Army Administration Center
ATTN: Dr. Sherrill
Ft. Benjamin Harrison, IN 46218

1 Dr. Joseph Ward
U.S. Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Air Force

1 Air University Library

AUL/LSE 76/443
Maxwell AFB, AL 36112

1 DR. G. A. ECKSTRAND
AFHRL/AS
WRIGHT-PATTERSON AFB, OH 45433

1 Dr. Alfred R. Fregly
AFOSR/NL, Bldg. 410
Bolling AFB, DC 20332

1 CDR. MERCER
CNET LIAISON OFFICER
AFHRL/FLYING TRAINING DIV.
WILLIAMS AFB, AZ 85224

1 Dr. Ross L. Morgan (AFHRL/ASR)
Wright -Patterson AFB
Ohio 45433

1 Research Branch
AFMPC/DPMYP
Randolph AFB, TX 78148

1 Dr. Marty Rockway (AFHRL/TT)
Lowry AFB
Colorado 80230

1 Brian K. Waters, Maj., USAF
Chief, Instructional Tech. Branch
AFHRL
Lowry AFB, CO 80230

Marines

1 Director, Office of Manpower Utilization

HQ, Marine Corps (MPU)
BCB, Bldg. 2009
Quantico, VA 22134

1 DR. A.L. SLAFKOSKY
SCIENTIFIC ADVISOR (CODE RD-1)
HQ, U.S. MARINE CORPS
WASHINGTON, DC 20380

CoastGuard

1 MR. JOSEPH J. COWAN, CHIEF
PSYCHOLOGICAL RESEARCH (G-P-1/62)
U.S. COAST GUARD HQ
WASHINGTON, DC 20590

Other DoD

1 Dr. Stephen Andriole
ADVANCED RESEARCH PROJECTS AGENCY
1400 WILSON BLVD.
ARLINGTON, VA 22209

12 Defense Documentation Center
Cameron Station, Bldg. 5
Alexandria, VA 22314
Attn: TC

1 Dr. Dexter Fletcher
ADVANCED RESEARCH PROJECTS AGENCY
1400 WILSON BLVD.
ARLINGTON, VA 22209

1 Military Assistant for Human Resources
Office of the Director of Defense
Research & Engineering
Room 3D129, the Pentagon
Washington, DC 20301

1 Director, Research & Data
OSD/MRA&L (Rm. 3B919)
The Pentagon
Washington, DC 20301

Civil Govt

1 Dr. Susan Chipman
Basic Skills Program
National Institute of Education
1200 19th Street NW
Washington, DC 20208

1 Dr. William Gorham, Director
Personnel R&D Center
U.S. Civil Service Commission
1900 E Street NW
Washington, DC 20415

1 Dr. Andrew R. Molnar
Science Education Dev.
and Research
National Science Foundation
Washington, DC 20550

1 Dr. Thomas G. Sticht
Basic Skills Program
National Institute of Education
1200 19th Street NW
Washington, DC 20208

1 Dr. Joseph L. Young, Director
Memory & Cognitive Processes
National Science Foundation
Washington, DC 20550

Non Govt

1 PROF. EARL A. ALLUISI
DEPT. OF PSYCHOLOGY
CODE 287
OLD DOMINION UNIVERSITY
NORFOLK, VA 23508

1 Dr. John R. Anderson
Dept. of Psychology
Yale University
New Haven, CT 06520

1 DR. MICHAEL ATWOOD
SCIENCE APPLICATIONS INSTITUTE
40 DENVER TECH. CENTER WEST
7935 E. PRENTICE AVENUE
ENGLEWOOD, CO 80110

1 1 psychological research unit
Dept. of Defense (Army Office)
Campbell Park Offices
Canberra ACT 2600, Australia

1 MR. SAMUEL BALL
EDUCATIONAL TESTING SERVICE
PRINCETON, NJ 08540

1 Dr. Nicholas A. Bond
Dept. of Psychology
Sacramento State College
600 Jay Street
Sacramento, CA 95819

1 Dr. John Seeley Brown
Bolt Beranek & Newman, Inc.
50 Moulton Street
Cambridge, MA 02138

1 Dr. John B. Carroll
Psychometric Lab
Univ. of No. Carolina
Davie Hall 013A
Chapel Hill, NC 27514

1 Dr. William Chase
Department of Psychology
Carnegie Mellon University
Pittsburgh, PA 15213

Non Govt

1 Library
HumRRO/Western Division
27857 Berwick Drive
Carmel, CA 93921

1 Dr. Earl Hunt
Dept. of Psychology
University of Washington
Seattle, WA 98105

1 Mr. Gary Irving
Data Sciences Division
Technology Services Corporation
2811 Wilshire Blvd.
Santa Monica CA 90403

1 DR. LAWRENCE B. JOHNSON
LAWRENCE JOHNSON & ASSOC., INC.
SUITE 502
2001 S STREET NW
WASHINGTON, DC 20009

1 Dr. Arnold F. Kanarick
Honeywell, Inc.
2600 Ridgeway Pkwy
Minneapolis, MN 55413

1 Dr. Roger A. Kaufman
203 Dodd Hall
Florida State Univ.
Tallahassee, FL 32306

1 Dr. Steven W. Keele
Dept. of Psychology
University of Oregon
Eugene, OR 97403

1 Mr. Marlin Kroger
1117 Via Goleta
Palos Verdes Estates, CA 90274

1 LCOL. C.R.J. LAFLEUR
PERSONNEL APPLIED RESEARCH
NATIONAL DEFENSE HQS
101 COLONEL BY DRIVE
OTTAWA, CANADA K1A 0K2

Non Govt

1 Dr. Frederick M. Lord
Educational Testing Service
Princeton, NJ 08540

1 Dr. Robert R. Mackie
Human Factors Research, Inc.
6780 Cortona Drive
Santa Barbara Research Pk.
Goleta, CA 93017

1 Dr. William C. Mann
USC-Information Sciences Inst.
4676 Admiralty Way
Marina del Rey, CA 90291

1 Dr. Richard B. Millward
Dept. of Psychology
Hunter Lab.
Brown University
Providence, RI 82912

1 Dr. Donald A Norman
Dept. of Psychology C-009
Univ. of California, San Diego
La Jolla, CA 92093

1 Dr. Jesse Orlansky
Institute for Defense Analysis
400 Army Navy Drive
Arlington, VA 22202

1 Dr. Seymour A. Papert
Massachusetts Institute of Technology
Artificial Intelligence Lab
545 Technology Square
Cambridge, MA 02139

1 MR. LUIGI PETRULLO
2431 N. EDGEWOOD STREET
ARLINGTON, VA 22207

1 DR. PETER POLSON
DEPT. OF PSYCHOLOGY
UNIVERSITY OF COLORADO
BOULDER, CO 80302

Non Govt

1 Dr. Micheline Chi
Learning R & D Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15213

1 Dr. Kenneth E. Clark
College of Arts & Sciences
University of Rochester
River Campus Station
Rochester, NY 14627

1 Dr. Norman Cliff
Dept. of Psychology
Univ. of So. California
University Park
Los Angeles, CA 90007

1 Dr. Allan M. Collins
Bolt Beranek & Newman, Inc.
50 Moulton Street
Cambridge, Ma 02138

1 Dr. John J. Collins
Essex Corporation
201 N. Fairfax Street
Alexandria, VA 22314

1 Dr. Meredith Crawford
5605 Montgomery Street
Chevy Chase, MD 20015

1 Dr. Donald Dansereau
Dept. of Psychology
Texas Christian University
Fort Worth, TX 76129

1 DR. RENE V. DAWIS
DEPT. OF PSYCHOLOGY
UNIV. OF MINNESOTA
75 E. RIVER RD.
MINNEAPOLIS, MN 55455

1 Dr. Ruth Day
Center for Advanced Study
in Behavioral Sciences
202 Junipero Serra Blvd.
Stanford, CA 94305

Non Govt

1 MAJOR I. N. EVONIC
CANADIAN FORCES PERS. APPLIED RESEARCH
1107 AVENUE ROAD
TORONTO, ONTARIO, CANADA

1 Dr. Richard L. Ferguson
The American College Testing Program
P.O. Box 168
Iowa City, IA 52240

1 Dr. Victor Fields
Dept. of Psychology
Montgomery College
Rockville, MD 20850

1 Dr. Edwin A. Fleishman
Advanced Research Resources Organ.
8555 Sixteenth Street
Silver Spring, MD 20910

1 Dr. John R. Frederiksen
Bolt Beranek & Newman
50 Moulton Street
Cambridge, MA 02138

1 DR. ROBERT GLASER
LRDC
UNIVERSITY OF PITTSBURGH
3939 O'HARA STREET
PITTSBURGH, PA 15213

1 DR. JAMES G. GREENO
LRDC
UNIVERSITY OF PITTSBURGH
3939 O'HARA STREET
PITTSBURGH, PA 15213

1 Dr. Ron Hambleton
School of Education
University of Massechusetts
Amherst, MA 01002

1 Dr. Barbara Hayes-Roth
The Rand Corporation
1700 Main Street
Santa Monica, CA 90406

Non Govt

1 DR. DIANE M. RAMSEY-KLEE
R-K RESEARCH & SYSTEM DESIGN
3947 RIDGEMONT DRIVE
MALIBU, CA 90265

1 MIN. RET. M. RAUCH
P II 4
BUNDESMINISTERIUM DER VERTEIDIGUNG
POSTFACH 161
53 BONN 1, GERMANY

1 Dr. Mark D. Reckase
Educational Psychology Dept.
University of Missouri-Columbia
12 Hill Hall
Columbia, MO 65201

1 Dr. Joseph W. Rigney
Univ. of So. California
Behavioral Technology Labs
3717 South Hope Street
Los Angeles, CA 90007

1 Dr. Andrew M. Rose
American Institutes for Research
1055 Thomas Jefferson St. NW
Washington, DC 20007

1 Dr. Ernst Z. Rothkopf
Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974

1 PROF. FUMIKO SAMEJIMA
DEPT. OF PSYCHOLOGY
UNIVERSITY OF TENNESSEE
KNOXVILLE, TN 37916

1 DR. WALTER SCHNEIDER
DEPT. OF PSYCHOLOGY
UNIVERSITY OF ILLINOIS
CHAMPAIGN, IL 61820

1 DR. ROBERT J. SEIDEL
INSTRUCTIONAL TECHNOLOGY GROUP
HUMRRO
300 N. WASHINGTON ST.
ALEXANDRIA, VA 22314

Non Govt

1 Dr. Robert Singer, Director
Motor Learning Research Lab
Florida State University
212 Montgomery Gym
Tallahassee, FL 32306

1 Dr. Richard Snow
School of Education
Stanford University
Stanford, CA 94305

1 Dr. Robert Sternberg
Dept. of Psychology
Yale University
Box 11A, Yale Station
New Haven, CT 06520

1 DR. ALBERT STEVENS
BOLT BERANEK & NEWMAN, INC.
50 MOULTON STREET
CAMBRIDGE, MA 02138

1 DR. PATRICK SUPPES
INSTITUTE FOR MATHEMATICAL STUDIES IN
THE SOCIAL SCIENCES
STANFORD UNIVERSITY
STANFORD, CA 94305

1 Dr. Kikumi Tatsuoka
Computer Based Education Research
Laboratory
252 Engineering Research Laboratory
University of Illinois
Urbana, IL 61801

1 DR. PERRY THORNDYKE
THE RAND CORPORATION
1700 MAIN STREET
SANTA MONICA, CA 90406

1 Dr. Benton J. Underwood
Dept. of Psychology
Northwestern University
Evanston, IL 60201

Non Govt

1 DR. THOMAS WALLSTEN
PSYCHOMETRIC LABORATORY
DAVIE HALL 013A
UNIVERSITY OF NORTH CAROLINA
CHAPEL HILL, NC 27514

1 Dr. Claire E. Weinstein
Educational Psychology Dept.
Univ. of Texas at Austin
Austin, TX 78712

1 Dr. David J. Weiss
N660 Elliott Hall
University of Minnesota
75 E. River Road
Minneapolis, MN 55455

1 Dr. Anita West
Denver Research Institute
University of Denver
Denver, CO 80201

1 DR. SUSAN E. WHITELY
PSYCHOLOGY DEPARTMENT
UNIVERSITY OF KANSAS
LAWRENCE, KANSAS 66044