DOCUMENT RESUME

ED 175 836 IR 007 602

AUTHOR Mantei, Marilyn M.
TITLE Modeling User Behavior in Computer Learning Tasks.
SPONS AGENCY Advanced Research Projects Agency (DOD), Washington, D.C.: Office of Naval Research, Washington, D.C.
PUB DATE 2 Apr 79
CONTRACT F4462[illegible]73-C-0074; N00014-76-0874
NOTE 29p.: Paper presented at the Annual Conference of the American Educational Research Association (San Francisco, California, April 8-12 1979)

EDRS PRICE MF01/PC02 Plus Postage.
DESCRIPTORS Computational Linguistics: *Computer Assisted Instruction: *Computer Programs: *Man Machine Systems: *Task Analysis

ABSTRACT

Model building techniques from Artifical Intelligence and Information-Processing Psychology are applied to human-computer interface tasks to evaluate existing interfaces and suggest new and better ones. The model is in the form of an augmented transition network (ATN) grammar which is built by applying grammar induction heuristics on a sequential record of a user's interaction with a computer system. A computer-aided instruction experiment is described which illustrates the model building technique. Variations in the task presentation are constructed in the ATN grammar and used to pinpoint interface design problems. (Author/RAO)

Modeling User Behavior in Computer Learning Tasks

Marilyn M. Mantei

Department of Psychology
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

2 April 1979

Abstract

Model building techniques from Artificial Intelligence and Information-Processing Psychology are applied to human-computer interface tasks. The model is in the form of an augmented transition network (ATN) grammar which is built by applying grammar induction heuristics on a sequential record of a user's interaction with a computer system. The goal of the modeling is to evaluate existing interfaces and to suggest new and better ones. A computer-aided instruction experiment is described which illustrates the model building technique. Variations in the task presentation are constructed in the ATN grammar and used to pinpoint interface design problems.

Paper presented at the annual conference of American Educational Research Association, San Francisco, California, April 8-12, 1979.

[1]This work was supported by the Office of Naval Research under contract number N00014-76-0874 and partially by the Defense Advanced Research Projects Agency under contract number F44620-73-C-0074.

I would like to thank Pat Langley and Don McCracken for discussions which led to many of the ideas presented in this paper.

1. Introduction

Over the past decade, researchers have been concerned with the problem of designing suitable interfaces for the computer user (Bennett, 1972; Nickerson, Elkind & Carbonell, 1968; Rouse, 1975). One major hurdle facing the design of good user-oriented systems has been the lack of a suitable and general representation of the user interface[1]; such a representation would describe a variety of user environments and serve as a communications tool between the human factors psychologist and the systems designer. In this paper, I present a methodology for representing person-computer interfaces and for analyzing interface difficulties from this representaion. To build my representation, I perform a task analysis and express the user's interaction with the computer system as an augmented transition network (ATN) grammar (Woods, 1970).

I perform this task analysis to build a description of the possible streams of behavior that a user might exhibit in a given task environment when performing a specific task. I build this description by performing grammar induction on experimental streams of behavior obtained from a task which represents the task environment under consideration. The approach is similar to Anderson's (1977) work with LAS except that I do not conceive of the resulting representation as a meaningful grammar. An iterative technique of prediction and experimentation is used to tune the representation; however, even with a first order model, the effects of the task environment on the user can be hypothesized from this structured view. The effects can be listed as desirable or undesirable and the task environment modified to optimize the desirable interface effects. This representation of a user's behavior can also indicate what modifications to the task environment will have a negligible effect.

In the second section of this paper, I present a simple user task and its accompanying representation. Following this overview, I give a detailed set of steps describing how the task analysis is performed. I then describe the computer system, ZOG,[2] on which I test my methodology. In the fifth section, I apply the task analysis techniques to a computer-aided instruction task, built in ZOG. Following the application of the task analysis on a test case, I discuss the method's generality and shortcomings. I also compare it briefly to the standard regression methods being used to study the user interface.

---

[1] Moran's work (1979) is a single exception to this problem. He builds a top-down representation which does not interface directly to experimental data.

[2] ZOG is not a mnemonic, but simply a name for the system.

2. An Example: A Letter Creation Task

In this section I give an example of what the finished product, the task analysis might look like. I have applied my technique to the task of typing a letter. The operations required to create the final finished version of the letter can be either motor movements of the typist, cognitive decisions on the part of the typist, or operations executed by the typewriter. The motor movements of the typist are such operations as the striking of a key or the space bar. The cognitive operations are the choices made to lay out the letter on the page according to some desired format. Typewriter operations constitute such items as bell ringing at the end of a long line or locking the keyboard. A sequence of these operations in the correct order generates the desired letter.

The task analysis for the typing task consists of breaking up the task into the recurring operations that form the task. These subunits are called *task elements* or simply, *elements*. They will fall into four categories; *goal*, *cognitive*, *motor*, or *interface* elements.

Goal elements are those dictated by the nature of the task, itself. In the letter typing task, goal elements are the requirements that the letter have a date, address, salutation, body and closing. The goal elements for the letter typing task are:

<generate date>
<generate address>
<generate salutation>
<generate body of letter>
<generate closing>
<generate notations>

Goal elements govern the entire task. Motor and cognitive elements describe how the goal elements are executed. A combination of motor, cognitive and interface elements in the correct order form the goal elements. Examples of motor elements are:

<put paper in carriage>
<hit carriage return>
<set spacing to 1>
<type character>
<strike space bar>

Cognitive elements are those parts of the task which require problem solving behavior on the part of the individual executing the task. They are reflected in the verbal or motor behavior that results from these internal states. Examples of cognitive elements in the typing task are such items as:

<scan item in address list>
<check if item is desired name>
<check if spacing = 10 lines>
<retrieve today's date from memory>
<discover error>

The cognitive elements govern the execution of the motor elements, e.g., spacing is discontinued by a cognitive check on the number of spaces already generated.

The interactive elements are those parts of the task execution which come from the environment but control what cognitive elements might follow. In the case of the typing task, they can be such simple indicators to the typist as:

<bell ringing warning margin's end>
<keyboard lock>
<light print indicating ribbon change>

In the typewriter task, these form a negligible part of the task analysis. With a more mutable environment such as a computer interface, the interactive elements form a major part of the task. They prompt what cognitive elements will be executed in reaction to the information they provide.

Insert Table 1 about here

Table 1 summarizes the types of task elements and their representations in the typing task. Their ordering is non-deterministic; however, several general heuristics can be applied to their sequencing.

1. a <goal element> subsumes <interactive elements>, <cognitive elements>, and <motor elements>.

2. a <cognitive element> follows a <cognitive element> or an <interactive element>.

3. a <motor element> follows a <motor element> or a <cognitive element>.

4. an <interactive element> follows a <motor element> or a <cognitive element>.

The task of typing a letter can now be broken into its task elements using the element types listed in Table 1. This task representation is presented as an ATN grammar describing the sequences of elements that can occur within a given context. Figure 1 is an illustration of

Goal Elements

<insert letter>
<generate letter>
    <generate date>
    <generate address>
    <generate salutation>
    <generate body>
    <generate closing>
    <generate notations>
<remove letter>

Motor Elements

<type character>
<hit carriage return>
<strike space bar>
<roll carriage>
<set spacing>
<type tab>
<type backspace>

Cognitive Elements

<retrieve today's date from memory>
<decide to begin new line>
<recognize that character is mistyped>
<decide to stop spacing>
<decide to begin new page>
<scan item in address list>
<check if item is desired name>

Interface Elements

<lock keyboard>
<ring bell>
<return carriage>

Table 1. Elements which form the components of a letter typing task.

how the ATN grammar describes the typing task. An ATN grammar is used because it allows one to incorporate semantics in the behavior description. These semantics are instrumental in understanding those areas of the representation which are potential trouble spots.

Insert Figure 1 about here

The cognitive elements of the task serve as choice points in the task and represent the selection of a given path from several possible ones; they appear at nodes in the grammar from which several links emanate. Note that the task analysis presented does not represent a single letter typing task but the class of such tasks. Letters will differ depending on the selections made at the choice points. The example I am considering is that of typing a letter on a standard office electric typewriter. The letter to be typed is illustrated in Figure 2. The copy from which the letter is typed can be assumed to be in reasonably finished form, simplifying this example.

Insert Figure 2 about here

The first choice made in generating the letter selects the line on which to start typing the date. This depends on the size of the letter and the default option for typing this particular type of letter. This is a cognitive action and follows the motor actions of inserting the paper and aligning it. In the case of this example, two motor elements describe the process, <set spacing to 1> and <hit carriage return>*[1].

<[illegible]rate date> fires the cognitive element to <retrieve date>, typically, from one's own memory. This, in turn, fires the motor elements to position the paper and type the date:

<strike space bar>*
<type month>
<type day>
<type comma>
<type year>
<hit carriage return>

These elements can be broken into further subelements such as <type '1'> <type '9'>, etc., but

[1]The asterisk in this notation means repeat as many times as needed

CREATE LETTER

→ <INSERT LETTER> → <GENERATE LETTER> → <REMOVE LETTER> →

INSERT LETTER

→ <PUT PAPER IN CARRIAGE> → <ADJUST PAPER> → <ROLL CARRIAGE> → <HIT CARRIAGE RETURN> →

GENERATE LETTER

→ <GENERATE DATE> → <GENERATE ADDRESS> → <GENERATE SALUTATION>
→ <GENERATE BODY OF LETTER> → <GERATE CLOSING> → <GENERATE NOTATIONS> →

GENERATE DATE

→ <STRIKE SPACE BAR> → <RETRIEVE TODAY's DATE FROM MEMORY> → <TYPE MONTH>
→ <TYPE DAY> → <TYPE YEAR> → <HIT CARRIAGE RETURN> →

TYPE MONTH

→ <TYPE KEY> → <STRIKE SPACE BAR> →

TYPE DAY

→ <TYPE KEY> → <STRIKE SPACE BAR> →

TYPE YEAR

→ <TYPE KEY> → <TYPE CARRIAGE RETURN> →

GENERATE ADDRESS

→ <SCAN ITEM IN ADDRESS LIST> → <CHECK IF ITEM IS DESIRED ONE> → <TYPE KEY>
→ <STRIKE SPACE BAR> → <TYPE KEY> → <HIT CARRIAGE RETURN> →

FIGURE 1. AN ATN GRAMMAR FOR TYPING A LETTER.

March 31, 1979

Dr. Joseph X. Smith
Department of Psychology
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

Dear Joe,

In regard to our telephone conversation of March 30, 1979, I agree that now is perhaps the best time to purchase new minicomputer equipment for the NET4 learning experiment. I suggest that we look at the four vendors on the list I am enclosing.

I will be in Pittsburgh next Thursday to discuss these matters further with you.

Yours sincerely,

Anne Browne

AB/jk
encl.

Figure 2. Sample of letter modeled in ATN grammar.

this will not be necessary for the example.

Note that if a wrong choice is made in selecting the number of lines to space before inserting a date, the following badly formatted letters might result.

1. A short letter squeezed at the very top of the page.
2. A letter which fits on page one except for one or two lines.
3. A letter positioned too close to the bottom of the page.

Any one of these letters needs retyping. Therefore, that particular choice point is crucial for task efficiency. A typewriter carriage which covers the untyped portion of the paper from view makes the correct decision more difficult. A task representation of this typing situation would indicate the problem and its cause. The task analysis which follows on the computer interaction tasks pinpoints similar choice points which can then be used to determine what computer system environment characteristics generate a poor choice of task elements to follow.

The above typing task description illustrates what constitutes a task analysis for the particular form of computer interaction tasks I am discussing. The particular task elements shown for the typing task are hypothesized; in the next section, I describe a methodology for obtaining task elements experimentally, and for generalizing from several experiments to a general case representation.

3. Performing the Task Analysis

As shown in the previous section, the analyses I perform on human/computer interactions consists of breaking a task into subunits or what I termed task elements. I implied in the typing task analysis, that the selected elements were general, and that they could be found occurring again and again in various parts of the task. Below I list my criteria for an operation to be selected as a task element.

1. The time to perform the task element should be relatively constant.
2. The time to perform the task element must be within the resolution of the time measuring device.
3. The task element must have a clearly observable start and stop point.
4. Task elements must be independent units of behavior.
5. Each task element must occur more than once.
6. A task element must occur in more than one combination of other task elements.

7. The properties that characterize a task element must be present in almost all instances of its occurrence.

8. The set of task elements must be exhaustive, that is, they must account for all of the observed stream of behavior.

The task under consideration in this paper is a computer-aided instruction task. The user is seated in front of a computer terminal. Text and questions about the text are presented to the user. The user responds to these questions by keying in a number for a selection from a list of possible answers displayed to him or her. Examples from this task are used to clarify the steps in the algorithm I am presenting for performing a task analysis.

Insert Table 2 about here

The bottom up execution of this task analysis is described in the steps listed below. Task goals and environment characteristics provide top-down constraints for building this representation. Table 2 is a synopsis of these steps. Below is a description of the steps to take in performing the task analysis; these steps are expanded further in terms of the computer learning task being analyzed in section 5.

Steps for Performing and Analyzing a Task Analysis

1. Select a simple example of the task to be studied. If the task is learning from a computer display, select a string of three or four displays that characterize the entire learning

2. Make a *record* of an individual performing this task either on videotape, on audiotape, via computer recorded keystrokes, by manual observation and note-taking or by any combination of the above. Elapsed time indicators must be included in any such record.

3. Mark the stream of behavior with *task delimiters.* Task delimiters are changes in the behavior stream; what changes will be observable depends partially on the recording medium. For a task with multiple observing methods, several delimiters will overlap each other. The process of marking the task delimiters occurs in the following order.

   - Mark all cues that govern the task. If the task requires page turning, then each page turn start is a delimiter. If the task requires reading, then text propositions and paragraphs are delimiters.

   - Select all motor operations that have distinct start and stop points. If the operation is typing a key, then the start delimiter occurs when the finger touches the desired key. The stop delimiter occurs when the key rises to

1. Select representative task.
2. Record task behavior.
3. Mark delimiters in recorded behavior.
4. Break task into units.
5. Assign units to behavior classes.
6. Mark task goals on recorded behavior.
7. Combine units into new units until only one remains.
8. Modifiy task and redo grammar induction, adding new details.
9. Calculate the amount of time each path takes.
10. Search for paths with long execution times.
11. Generate and test hypotheses for long execution times.

Table 2. Steps for performing and analyzing a task analysis.

the initial position again.

- Select all changes in the environment as a task delimiter. If the computer finishes a previous task and indicates its completion by displaying a flashing asterisk, this display serves as a task delimiter. The appearance of another character (possibly typed by the user) on the display screen marks still another task delimiter.

- Look for verbal cues such as statements of task completion or commentary that indicates the end of a sequence or the start of a new one. Comments like "There" or "Now, let's do" or "Yes, that's what I want" serve as verbal task delimiters.

4. Break the task under consideration into units based on these delimiters and label these units. If the stream of behavior is a verbal record and contains the comment "Oh, new page" followed by the comment "Now let's see what do I want to do here," the unit of behavior between these verbal delimiters is a likely candidate to be labeled <turn page>. This is a simple example; other labels can often be <A1> <A2>, etc. because missing delimiters or too many delimiters make it difficult to perceive what behavior is taking place.

5. Once the units of behavior have been labeled, determine the amount of time it takes to execute these elements and calculate their descriptive statistics, i.e., means, variance and frequencies of occurrence. Use these statistics to help decide if units are members of the same class. Attempt to reclassify elements that appear as outliers by grouping them into larger elements or breaking them into smaller ones.

6. Lay out the goals of the task being executed in as detailed a form as possible. Mark these goals on the record of user behavior. Mark higher level goals with stronger weights than lower level goals.

7. Form groupings of two behavioral units based on their co-occurrence. For example, if one type of element follows another in several cases, consider this sequence a new group. Calculate the probabilities of one unit following a second one and throw out those groups below a set threshold level. If groups overlap, find those instances where the overlap occurs together in the behavioral stream and select the grouping which occurs first. If there is no overlapping, retain both groupings. Rename these groupings as higher level elements.

8. If an element follows itself, remove one of the elements from the stream of behavior, but indicate the repetition on the remaining element.

9. If groupings overlap a goal marker, they are required to completely represent the behavior between adjacent goal marks of equal weight. If this is not the case, redo the groupings, recombining the elements within each goal unit. Only cross the goal boundary after such elements have been combined.

10. Repeat steps 6 through 8 until a single remaining group exists. What remains is a structure that forms the first stage of an augmented transition network grammar. Rewrite this structure such that its nodes are states and its arcs are actions that pass the elements that combined to form a given state.

11. Rerun the prototype task with small modifications and different subjects. In each case, execute steps 1 through 8, generating a new structure. Compare each structure looking for differences and resolving these differences by expressing all possibilities in a larger ATN representation. For example, if subject 1 exhibits behavior <A> <B> and subject 2 exhibits behavior <A> <C>, the resulting expression should look like Figure 3.

12. Calculate from the experimental data the average probability of occurrence of each arc in the ATN grammar. Select every path in the representation and calculate the amount of time it would take to execute using the probabilities to obtain values for paths that have multiple choice points.

13. Using the times for these paths, at each level in the grammar, look for paths which take a long time to execute. Those paths which represent long completion times point to possible areas of difficulty with the user interface.

14. Once possible trouble areas have been selected, mark the hypothesized reasons for various path selections on the arcs. Use these semantics to suggest different interface designs that would lower the path time.

Insert Figure 3 about here

At this point, the ATN grammar has been produced and labeled with time to execute values. Linear models of an individual's behavior can be built by selecting various paths in the grammar. These can be verified by putting the initial data for the time to execute for each task element into a linear regression equation.

The grammar can be refined by additional experimentation using other tasks and subjects. In addition tasks not yet undertaken with the particular computer interface can be broken down into the elements already expressed in the grammar. Their ordering can be hypothesized and a prediction generated of expected task completion times. Thus, the representation serves not only as an analysis tool but as a generator of possible behaviors in new situations.

4. The ZOG System

Before beginning a detailed example of the task analysis I am describing, I will briefly present a description of ZOG, the computer system on which the behavior measures take place. ZOG is described in detail elsewhere (Robertson, Newell, & Ramakrishna, 1978; McCracken & Robertson, 1979); I present only the essentials of its operation here.

ZOG is a communication agent. It is used to aid information flow between an individual seated

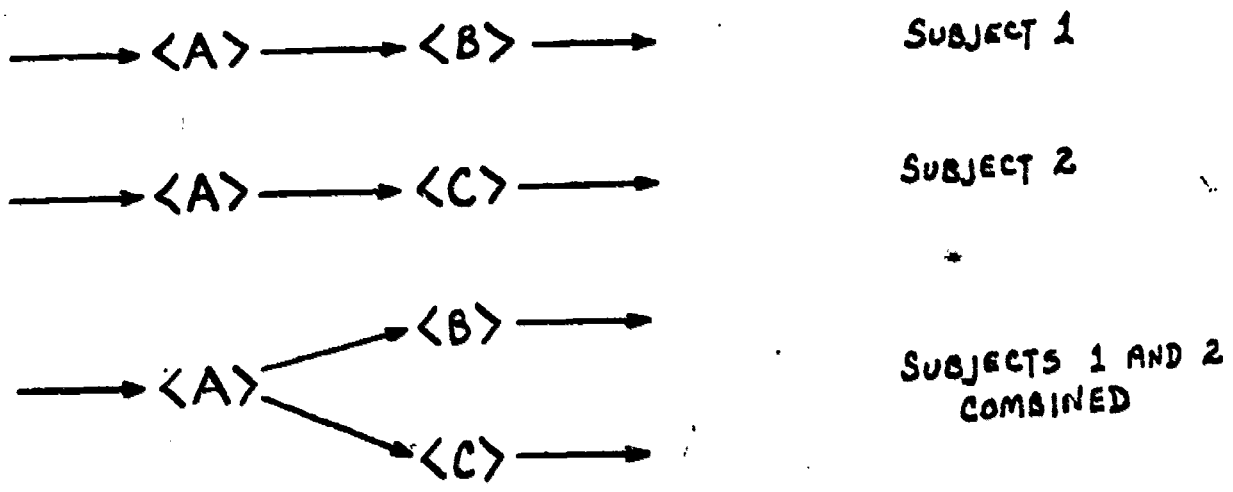

FIGURE 3. COMBINING GRAMMARS FOR TWO SUBJECTS.

at a computer terminal and a ZOG database. The computer aided instruction task occurs in this mode.

ZOG databases have a unique structure. The unit of storage in the database is a computer terminal display of text. Each of these displays is called a *frame*. Figure 4 illustrates a sample ZOG frame. Each of the frames in a ZOG database may be linked to any of the other frames, forming a complete network. Such a network is termed a *zognet*. A link is represented as a numbered selection on a frame. When a current frame is displayed before a user, any one of the frames to which this frame is linked can be brought before the user when he or she types the number associated with a selection. The text for these selections briefly describes the contents of the linked frame. This is illustrated in the five selections on Figure 4's example frame.

Insert Figure 4 about here

A series of selections by a user will present an equivalent number of displays to the user. This sequence of selections is called a *path*. The user has commands available at the bottom of the screen display which allow him to back up on the path he has taken, jump to a new frame not on the current path, or view the context of his path within the zognet he is traversing. Users are able to change a zognet by building new frames, editing existing ones or changing links. ZOG is a menu-based display system; all command options available to the user are displayed before him as a selection menu. ZOG's primary operation is the display of information.

5. Modeling a Solitaire Learning Session

The task being modeled is that of an individual learning how to play a game of solitaire from a set of rules and questions presented to the user with the ZOG system. The card game is the game of Golf.

Each sentence of the set of rules appears on a single ZOG frame. The subject reads the rule and types a 'q' to obtain the question display. The question display contains a question about the rule or an integration of the rule with previous card game knowledge. Each question has two to five possible responses, only one of which is correct. Figure 5 illustrates a sample rule and question frame for this task. If subjects choose an incorrect answer, they are displayed an error frame which explains the correct answer. They then type a 'c' to continue on to the next rule.

Standard Statistical Packages

This net describes the basic statistics methods required of most psychologists. A brief description of the underlying theory for each of the methods is followed by a list of the packages available for performing the analysis.

1. Descriptive Statistics
2. Correlations
3. Analysis of Variance
4. Factor Analysis
5. Multidimensional Scaling Analysis

back edit find goto help info mark return

Figure 4. Example of a ZOG frame

Insert Figure 5 about here

Before learning the game of Golf, subjects pass through several frames telling them how to respond to the CAI frames. They answer questions about their previous card game knowledge, take a short reading speed test on-line, and learn a simpler solitaire game. After finishing all rules and questions about Golf, subjects are allowed to review the rules in consolidated form. Following the computer-aided instruction (CAI) session, subjects are given a deck of cards and asked to play the game. They are advised that this will take place before they start the session. Conceptual and strategic errors in playing the game are noted.

The record of continuous behavior for this task is a set of computer recorded keystrokes. This is supplemented by knowledge of the task structure and goals. Because the data record contains only keystrokes, measurement resolution is low (approximately the time it takes to read a ZOG frame). A task analysis can be performed with this low level of resolution; if future measures, such as eye movement data, are of higher density, they can be incorporated in the existing model by expanding one of the terminals of the grammar. The ATN grammar resulting from this low-level data still provides several insights into the CAI task. The results from the analysis of the grammar are not astonishing, but serve to illustrate the methodology and its possibilities.

The obvious delimiters in this task are keystrokes. <Read rule> elements are terminated with a 'q'; <read question> elements are terminated with a numeric keystroke and <read error> elements terminate with a 'c'. Table 3 illustrates a partial computer record of a subject learning the solitaire game. The keystrokes and the interface elements of the ZOG system are used as task delimiters. Column 1 of Table 3 contains hypothesized element names that are assigned to the units between the task delimiters. Columns 2 and 3 list the start and stop delimiters of the task, and column 4 gives the associated time for the task elements.

Insert Table 3 about here

Note in Table 3 that more than one 'q', 'c' or number is sometimes typed following the display of the next frame. The subject in this experiment is a heavy-fingered typist who depressed the terminal keys firmly and held them depressed for approximately one second. Because the terminal and the ZOG system allow *type ahead*, this key depression was interpreted as new keystrokes. The first 'q' recorded caused the question display to appear. Additional q's were

RULE: One card at at ime may be moved form the top of a tableau pile, if it is in sequence with the card on top of the talon. The sequence may be up or down.

QUESTION: Which of the following card sequences cannot be placed one at a time on top of the talon?

1. 6-7-8-7-8
2. A-2A
3. J-10-9-9-8
4. Q-J-Q-K

Figure 5. Example of CAI frame.

Element	Start delimiter	Stop delimiter	Elapsed time
1. <read rule>	display frame	'q'	10.90
2. <wait question>	display frame	'q'	0.65
3. <wait question>	beep	'q'	0.48
4. <wait question>	beep	'q'	0.02
5. <wait question>	beep	'q'	0.03
6. <type '3'>	beep	'3'	2.98
7. <wait rule>	display frame	'3'	0.92
8. <type 'q'>	beep	'q'	31.52
9. <read question>	display frame	'1'	13.05
10. <wait error>	display frame	'1'	1.05
11. <wait error>	beep	'1'	0.05
12. <type 'c'>	beep	'c'	24.33
13. <wait rule>	display frame	'c'	0.98
14. <wait rule>	beep	'c'	0.03
15. <wait rule>	beep	'c'	0.33
16. <type 'q'>	beep	'q'	7.70
17. <read question>	display frame	'1'	19.03
18. <read rule>	display frame	'q'	18.72
19. <wait question>	display frame	'q'	0.67
20. <wait question>	beep	'q'	0.03
21. <wait question>	beep	'q'	0.23
22. <wait question>	beep	'q'	0.03
23. <type '1'>	beep	'1'	7.38

Table 3. Example of computer captured record of solitaire learning task.

considered an error selection (because no 'q' selection existed on a question frame), and the system responded with an audible beep to indicate a mistyped key. The task elements assigned to this additional key interpretation by the system are <wait question> <wait error> or <wait rule>. They can slow the display by up to two seconds, but their presence does not significantly affect overall subject performance.

If no type ahead occurs, for example, in lines 2, 17 and 18 of Table 3, the task element is interpreted as <read rule> or <read question>, representing the time to process a rule or a question frame. Lines 2 thru 6 or 7 thru 8 can be combined to form these elements. In walking through this task analysis, I will temporarily discard the detail of <wait question>, etc.; however, since time is the crucial analysis measure in performing a task analysis, I have categorized each <read question> <read rule> and <read error> element by the amount of time required for their execution. This classification is indicated in Table 4 along with an explanation of the higher level elements that are formed from <read question> <read rule> and <read error> combinations.

Insert Table 4 about here

Using the methodology described in section 3, higher level elements are built from the <read rule> and <read question> elements. Their combination is controlled by the goal of the task which is to learn the game of Golf; this goal is broken into subgoals of learning each rule of the game. Quite naturally, these are further split into the goals of acquiring and integrating the information presented in each frame. Figure 6 shows both the task goals and their application to the sequence of task elements in the behavior record. The sequence of task elements is taken from the behavior record shown in Table 3.

Insert Figure 6 about here

Levels of detail for the goals and task elements are indicated in Figure 6. Each higher number goal level is a set of subgoals of the previous level. The levels for the task elements are similar. They represent the combination of lower-level task elements into new elements; this combination process is controlled by the dashed lines representing the goal levels. All elements within the dashed line boundaries are to be combined before combining occurs across boundaries. Once at the boundary level, the only constraint on combining elements is

Task Element	Explanation
<rule 1>	*Read rule in < 10 seconds*
<rule 2>	*Read rule in 10-20 seconds*
<rule 3>	*Read rule in 20-30 seconds*
<rule 4>	*Read rule in 30-40 seconds*
<rule 5>	*Read rule in > 50 seconds*
<quest 1> : <quest 5>	*Read questions according to above times*
<error 1> : <error 5>	*Read errors according to above times*
<RQ>	*Read rule and answer question*
<RQE>	*Read rule, answer question and read error message*
<Q+>	*Learn set of rules*
<QE>	*Learn two rules making error on second question*
<EQ+>	*Make error on first rule, then none on rest*
<QEQ+>	*Make error on second rule, then none on rest*
<QEQ>	*Learn rule, make two errors, then none following*
<QQ>	*Learn rules with some errors intermingled*
<QQ-	*Learn game of solitaire*

Table 4. Meaning of task elements for solitaire learning task.

Task Goals			Task Elements				
Level 1	Level 2	Level 3	Level 5	Level 4	Level 3	Level 2	Level 1
Learn Solitaire Game	Learn Rule1	Read Frame1	<RULE2>	<RQ>	<RQ>	<QE>	<QEQ#>
		Read Frame2	<QUEST1>				
	Learn Rule2	Read Frame3	<RULE4>	<RQ>	<RQE>		
		Read Frame4	<QUEST2>				
			<ERROR3>	<E>			
	Learn Rule3	Read Frame5	<RULE1>	<RQ>	<RQ>	<Q#>	
		Read Frame6	<QUEST2>				
	Learn Rule4	Read Frame7	<RULE2>	<RQ>	<RQ>		
		Read Frame8	<QUEST1>				
	Learn Rules	Read Frame9	<RULE1>	<RQ>	<RQ>		
		Read Frame10	<QUEST1>				

FIGURE 6. TASK ELEMENTS FOR SUBJECTS LEARNING SOLITAIRE IN ZOG.

their high probability of co-occurrence.

In the solitaire game, <read rule> followed by <read question> occurs most frequently, so a new element <RQ> is formed. Although <RQ> now follows <RQ> more frequently than any other element, the goal boundaries for learning rules are not to be crossed until the <read error> element is combined; therefore, <RQE> is formed from <RQ> and <read error>. This brings us to level 3 where we can combine strings of <RQ> to form <Q*>. <RQ> and <RQE> are all that remain and are combined to form <QE>. Elements <QE> and <Q*> then form the single start state of the grammar, <QEQ*>. Table 4 gives an explanation of each of these states.

The grammar formed from the elements in Figure 6 is an illustration of ATN grammar generation from the behavior record in Table 3. Figure 7a illustrates the grammar that results from the entire record for this same subject. Since subjects made different errors in answering questions, the grammar resulting from running additional subjects in this task becomes more general. This is shown in Figure 7b.

Insert Figure 7 about here

The ATN representation in Figure 7 is shown in a non-standard form to provide clarity. The elements should be information that is passed along the links, and the nodes in this representation are states from which several links emanate. Which path is taken, depends on the conditions that exist when an individual is in that particular state. The rules for which path to take are the semantics of the ATN representation. At present, no semantics are applied to the grammar that has just been built. The purpose of the representation is to find paths in this grammar which take a long time and then to hypothesize why these paths were executed and to generate task environment changes for making them shorter.

The choice points of the ATN grammar for the solitaire learning task are rather simple. The two main ones diverge to <RQ> and <RQE> states. Five subjects were run on this task of reading and assimilating 12 rules, 12 questions and possibly, 12 error messages. Out of the 60 questions that were asked of the subjects, 20 or 33% of the answers were in error. The average time for execution of <read error> was 9.43 seconds, adding 37.7 seconds to each subject's time. The average time for a subject to read 12 rules and 12 questions was 342.2 seconds. The addition of the error processing increases the time to do the task by 11%. The error link is a trouble point in the CAI task. This pinpointing of reading error messages as a trouble spot in the task is an obvious result obtainable by simple statistics; the result, in this case, was obtained by using a small but general set of heuristics.

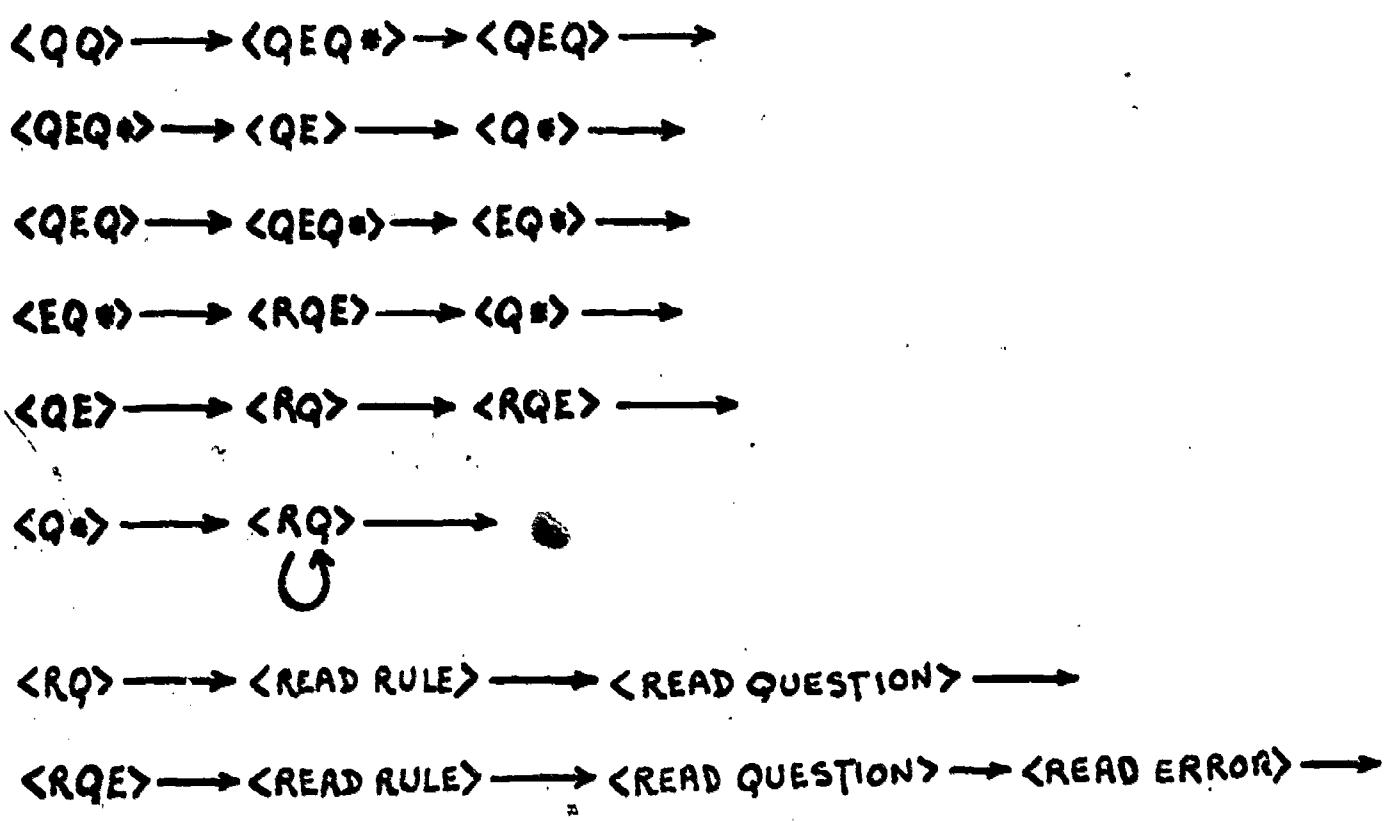

FIGURE 7a. AUGMENTED TRANSITION NETWORK GRAMMAR FOR SUBJECT 1.

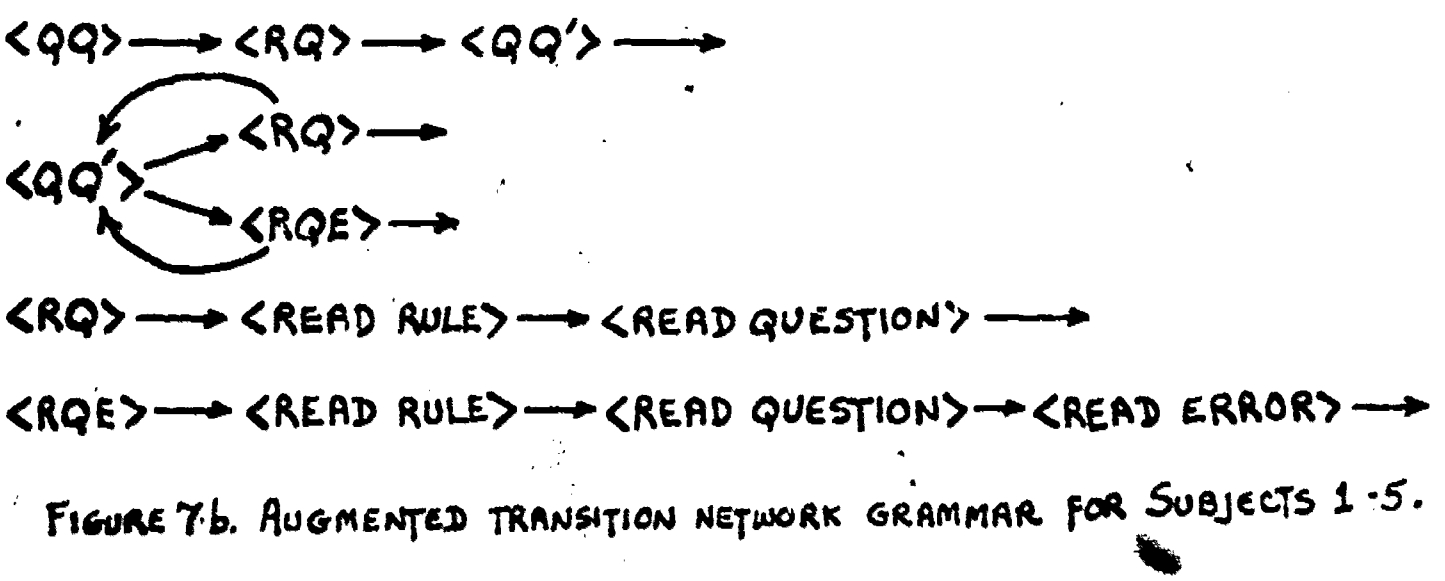

FIGURE 7b. AUGMENTED TRANSITION NETWORK GRAMMAR FOR SUBJECTS 1-5.

A second consideration in looking at the cost of reading error frames is that of the heavy-fingered typist. Given a different CAI interface where all selections were numbers, the unexpected type ahead would have caused numerous error frames to be selected, plus too fast a display of the individual frames for learning purposes. The high cost of such a path would indicate a need for a system design change.

A second variation of the solitaire learning task was tested on six new subjects. In this second task, the rule and the question about the rule are presented on the same frame display. The resulting ATN grammar built from the behavior records is shown in figure 8a. Note that more question and rule frames occur between the occurrence of error frames in this representation.

Insert Figure 8 about here

In this second task, errors are not a serious problem. Nine errors occur for 6 subjects answering 12 questions each (13% errors). The average time to read an error frame in this task was 11.9 seconds. The average task time for each subject was 316.4 seconds so that an average of 17.9 seconds spent on errors for each subject adds 5.6 percent to the time to execute the task. If the grammar in Figure 7b is combined with that of Figure 8a, the grammmar shown in Figure 8b will result. The decision applied to the initial diverging links is whether the rules and questions are combined or separate. The combined path is the least expensive one. This result compares to that already known in the CAI literature (Holtzman, 1970), but again, it falls cleanly from the methodology.

If we examine paths in the grammar at the less aggregate level, we find that some rules take longer to read than other rules. The resolution of the data does not allow us to make many hypotheses about the causes of these differences except by looking at the number of words or propositions in each frame. Those frames for which longer times are explained by text length are considered a necessary part of the task, but outliers from these explanations can be looked at as possible trouble areas. This is a suggestion for deeper analysis of the data using the methodology being presented.

6. Conclusion

By building an ATN grammar from the low resolution data obtained from this task, two problem areas were noted; the first was that of excessive subject errors in the learning task;

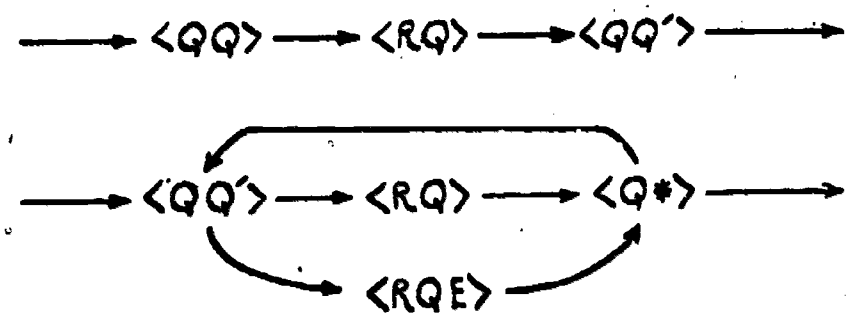

FIGURE 8a. ATN GRAMMAR FOR COMBINED RULE AND QUESTION.

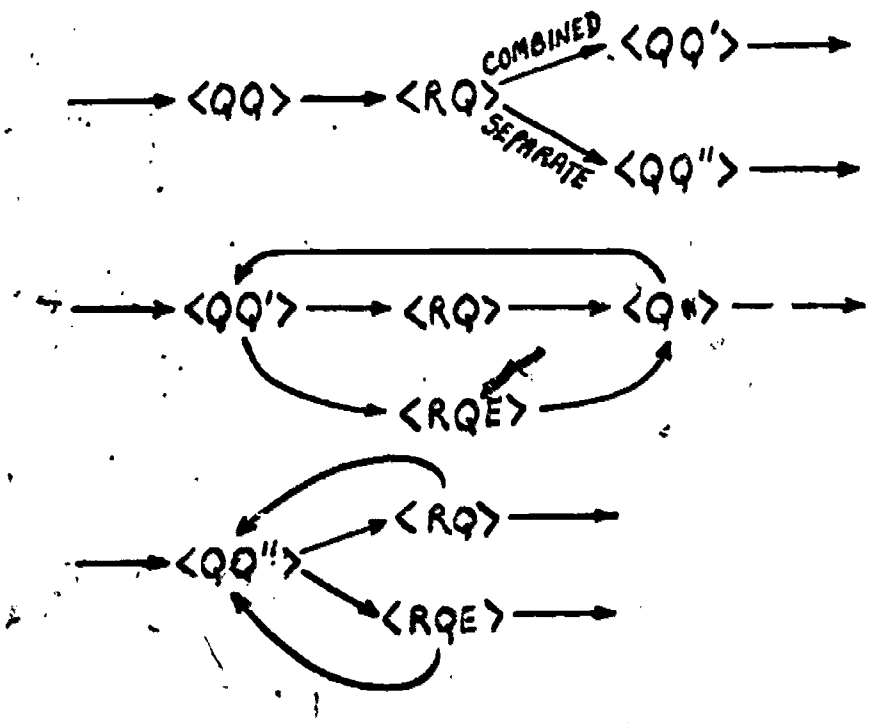

FIGURE 8b. ATN GRAMMAR FOR COMBINED AND SEPARATE TASKS.

the second was an interface problem with the type ahead facility. A resolution to the error problem came from modifying the task slightly. The type ahead problem would not have shown up as a significant difficulty if it had not been generated as a task element in the behavior record and if the bad effect of errors had not been noted.

The CAI task was selected for presentation in this paper because of the simple grammar that resulted and the indications of problem areas that were important and occurred at high levels. Application of the methodology described in this paper to more complex tasks such as computer text editing or bibliographic retrieval would investigate such areas as subject strategies and the effect of computer command structures. In this environment, the resulting grammars would be very complex and expected to discover interesting problem areas. This methodology works best when the task is well-structured. Without goal structures to apply to the task, little control is exercised over the grammar building resulting in incorrect representations of the task.

If a regression analysis is performed on the same task element data, information is lost about the exact change point in the process where less cost effective paths were taken. Regression analysis can be used in conjunction with the ATN grammar building. For example, if a set of causes are hypothesized for a path taking a long time to execute, a regression equation can indicate how important these causes are. Large residuals from a regression run can also be used to indicate new problem areas.

The methodology I am proposing is still in its developmental infancy. It has not been tried on a large variety of tasks or subjects. There are problems of order and structure to be considered in the induction heuristics. Until a large variety of tasks have been tested, the generality and validity is questionable. How much structure is required of a task in order to perform the grammar induction is not known. The results from the computer aided instruction task look promising.

7. Bibliography

Anderson, J. R. Induction of augmented transition networks. *Cognitive Science, 1*, 1977, 125-157.

Bennett, J. L. The user interface in interactive systems. In Cuadra, C. (Ed.) *Annual Review of Information Science and Technology, 7*, American Society for Information Science, Washington, D. C., 1972, 159-196.

Holtzman, W. H. (Ed.) *Computer-assisted Instruction, Testing and Guidance*. New York: Harper and Row, 1970.

McCracken, D. L. and Robertson, G. G. Editing tools for ZOG, a highly interactive man-machine interface. To be presented at the International Conference on Communication, Boston, Massachusetts, June 10-13, 1979.

Moran, T. P. Introduction to the Command Language Grammar: A representation for the user interface of interactive computer systems. Report SSL-78-3, Systems Sciences Laboratory, Xerox Palo Alto Research Center, Palo Alto, California 94304, October 1978.

Nickerson, R. S., Elkind, J. I. and Carbonell, J. R. Human factors and the design of time sharing computer systems. *Human Factors*, 10, 2, 1968, 127-137.

Robertson, G., Newell, A. & Ramakrishna, K. ZOG: A man-machine communication philosophy. Technical Report, Computer Science Department, Carnegie-Mellon University, Pittsburgh, Pennsylvania 15213, 1977.

Rouse, W. B. Design of man-computer interfaces for on-line interactive systems. *Proceedings of the IEEE, Special Issue on Interactive Computer Systems*, 63, 6, June 1975, 847-857.

Woods, W. A. Transition network grammars for natural language analysis. *Communications of the ACM, 13*, 1970, 591-606.