A Formal Grammar Approach to Human Factors Research

Herbert E. Dunsmore
Purdue University, dunsmore@cs.purdue.edu

Report Number:
86-623
A FORMAL GRAMMAR APPROACH
TO HUMAN FACTORS RESEARCH

H. E. Dunsmore

CSD-TR-623
August 1986
A Formal Grammar Approach to Human Factors Research

Dr. H. E. Dunsmore
Associate Professor
Department of Computer Sciences
Purdue University
West Lafayette, Indiana 47907-2004

ABSTRACT

Formal grammars and languages are constructed to describe the actions that human subjects must take in using an interactive system for non-programmers. The interactive system has been constructed for experimentation in attempt to determine some guidelines for system design for maximum productivity. Examination of the formal languages suggest that a form of written documentation will be superior to the online documentation provided to some subjects, that a compact presentation of information by the system will be more difficult for the users to deal with, and that providing users a means of anticipating system questions and answering ahead will not make them any more productive. Results from two separate experiments support these hypotheses.

INTRODUCTION

The field of human factors research is frequently judged to be lacking in rigor [Brooks, 1980]. There have been some models of programming behavior and computer user behavior (cf., the syntactic/semantic model in [Shneiderman, 1980]). But, the experimenter's intuition is typically the only support for hypotheses about experimental treatments.

There have been some attempts to introduce formalism into human factors research. Foley and Wallace [1974, 1978] expressed the notion that user actions at a terminal can be viewed as a language. Embley [1978] used a variation of BNF to examine a control construct for programming interactive dialogues for computer-aided instruction. Embley et al. [1978] proposed a model for predicting terminal session times for interactive editor users. The model parameters were "think time", typing rate, and system response time. The model successfully predicted the superiority of one editor over another.

Moran [1978] developed a Command Language Grammar (CLG) formalism to describe all levels of a system. Card, Moran, and Newell [1980a] proposed a simple "Keystroke-Level Model" to predict the time it takes a user to perform a task using an interactive system. The model counts keystrokes and other action level operations. When some 28 experimental subjects were required to perform 14 tasks on 10 different systems, the observed times agreed with the hypothesized times within about 20%. Using a similar model to analyze a text-editing system, the same researchers [Card et al., 1980b] were able to predict with about 80% accuracy the method a user would employ for a given task. Thomas [1978] attempted to build a partial model of man-computer communication using game theory and information theory. The model involves encoding-decoding and has a design-interpretation aspect (based upon assumptions about functions of the communication).
In the next section we briefly describe research that suggested that a formal grammar approach to human factors might be useful [Reisner, 1979]. The following section describes ongoing research concerning interactive systems for use by non-programmers [Dunsmore, 1980]. We next describe how the formal grammar approach was used to predict subjects' performance when using the interactive system. Finally, we report experimental results that support the formalism's hypotheses.

USING A FORMAL GRAMMAR IN THE DESIGN OF AN INTERACTIVE GRAPHICS SYSTEM

In previous work, Reisner [1979] was among the first to use a formal grammar in human factors research. Specifically, an action language for an interactive system was examined. The intent was to show that an action language could be formally described and that the formal description could be used to compare alternative designs for simplicity and consistency. The particular system studied, ROBART, is an experimental interactive color graphics system for creating slides for technical presentations.

There were two versions of ROBART, called ROBART1 and ROBART2. ROBART1 employed a physical switchbox for selecting shapes while ROBART2 replaced this with a menu of icons (symbolic pictures) on the terminal screen. ROBART2 also used a single EXECUTE key on the keyboard in place of ROBART1’s START, END, and GO buttons.

It could be argued that ROBART1 is more "efficient" than ROBART2. To start continuous shapes in ROBART2 it is necessary to position the cursor with a joystick, and then lift the hand from the stick to press the EXECUTE key. ROBART1 employs a knob on the top of the joystick and does not require that the hand be removed. Furthermore, ROBART2 is more stringent in its requirements for connecting lines. In ROBART1 the endpoint of one line may be used as the beginning of the next. In ROBART2 the start and end points of all lines must be explicitly indicated.

One way to describe the use of the two systems is to describe the actions that a ROBART user must make. In each case a BNF-type formal grammar was constructed and the derived language was termed an action language for ROBART1 or ROBART2. The start symbol for each system was the non-terminal draw picture. The nonterminals represent sets of similar actions (such as draw colored shape, draw continuous line, etc.). The terminal symbols represent actions the ROBART user has to learn and remember (e.g., PRESS BUTTON, ROTATE KNOB, POSITION CURSOR, etc.).

With these formal descriptions of ROBART1 and ROBART2 it was possible to compare design alternatives in the two systems. Three aspects of the grammars and action languages were employed:

1. The number of different terminal symbols represent the total vocabulary of actions the user can take.
2. The lengths of terminal strings indicate the number of actions the user must perform for given subtasks. This aspect is termed string simplicity.
3. The number of rules necessary to produce some set of terminal strings may be used to compare related subtasks. It seems appropriate that tasks that are cognitively similar should require nearly the same number of rules to go from the start symbol to the terminal strings. This aspect is called structural consistency.

Based on action language considerations several predictions were made concerning ROBART1 and ROBART2. Ten experimental subjects learned and used both ROBART systems. They were required to perform the same tasks in both. Considering string simplicity, it was predicted that remembering how to select shapes in ROBART1 should vary in difficulty. The lengths of the terminal strings could vary from one action to five. The empirical results supported this hypothesis. Of ten subjects, all remembered how to select a line, eight recalled how to select a continuous line, six - a box, four - a
continuous box, two - a circle, but only one could remember how to select a continuous circle.

Selecting any shape in ROBART2 required exactly one action. Thus, the prediction was that remembering how to select shapes in ROBART2 should not vary. Furthermore, this recall should be much better in ROBART2 than for some of the trickier shapes in ROBART1. The experimental subjects performed exactly as this hypothesis suggested. In every case nine or all of the ten subjects recalled how to select the given shape.

Several other predictions were made using the action grammars and languages. In general, empirical results supported those hypotheses. Formal analysis suggested that ROBART2 was "better" than ROBART1. The behavior of the subjects reflected this as well. Subjects spent less time learning ROBART2 (51 minute vs. 76 minute mean times) and made fewer errors while using ROBART2 (6.4 vs. 8.6 mean errors).

This earlier work shows that a formal description of an interactive system can be an analytic tool. It clearly demonstrated that: (1) User actions at a terminal can be described by means of a formal grammar, (2) The formalism can be used for making predictions, and (3) The predictions can be empirically tested. The remainder of this paper describes how a similar formalism was applied to another interactive system.

AN EXPERIMENT CONCERNING DESIGN OF AN INTERACTIVE SYSTEM FOR NON-PROGRAMMERS

A research project is in progress at Purdue University to explore computer systems for non-programmers [Dunsmore, 1980]. The approach is to conduct empirical investigation using prototype systems and representative subjects. The type of system considered is one in which an "outer layer" communicates with the user determining what his problem is and/or what software is available in the "inner layer" to solve that problem. The inner layer is envisioned as a set of packages, programs, modules, routines, etc. that may be placed into execution for the non-programmer. The outer layer supervises the entire process. Work has progressed by constructing elementary prototypes and then by enhancing them while subjecting the enhancement decisions to experimentation.

The version of the system we will consider in the remainder of this paper is one that has an information retrieval flavor. It allows the non-programmer to gather information from specific major categories. The system communicates with the user via a question-and-answer format. Instead of a response, the user might instead make a request to receive help, to backup one question, to restart the questioning process, or to exit the system.

In the spring of 1980 we conducted an experiment involving three factors. First, we examined the effect of the mode of information display. Our system uses a scrolling technique in which the top line rolls off the screen as the user or system enters a new line at the bottom. One version of our system employed a horizontal format in which we attempted to fill up each system information line as much as possible. This is contrasted with the vertical format in which separate elements are placed on separate lines. For example, if the system were displaying elements A-F (and if three elements fit on a line), then the horizontal subject would see

A B C
D E F

and the vertical subject would see
The horizontal mode has the advantage of maintaining more potentially-useful information on the screen. The vertical mode seems a little easier to use since information is less crowded.

In the standard single response version of our prototype, the user progresses by supplying responses to each question asked by the system. We provided a multiple response version that allowed users (if they desired) to anticipate system questions and supply several responses at once. For example, suppose a subject knew from his experience with our system that it was going to ask him questions Q1, Q2, Q3 to which he desired to respond R1, R2, R3. In the multiple response version he could respond to Q1 with all three responses. In the single response version he had to wait for the questions to serve as prompts. From earlier research we had observed annoyance in such a situation; thus we created the multiple response enhancement.

Finally, we were interested in the effect on productivity of the documentation available to the non-programmer user. Our base system began each session by presenting the subject with a cryptic description of system capabilities. We termed this the null documentation version. Pilot studies had suggested that more documentation might be useful, so we enhanced the base system as follows:

We created a two-page (8.5 x 11 front and back) document that described more fully the system capabilities, commands, and data available. The subjects who received this written documentation used the base system and the piece of paper. They could refer to the paper as often as they desired during their experimental session.

We enhanced the base system so that it gave initially the same information as appeared in the document. Furthermore, subjects in this online documentation group could view the entire documentation (two screens worth) at any time during their experimental session by responding to a system question with the character "1".

Our subjects in the spring of 1980 were thirty-six Purdue University students. Most were upper-level Management majors with minimal computer training. All subjects were selected to be representative of the non-programmer professional. Each subject was given a standard set of forty-five problems to solve using the system. They were told to answer as many questions as possible in their forty-five minute session, and that they need not answer them in any particular order.

Our experiment was a posttest-only different treatment design [Campbell and Stanley, 1963] in which each subject was randomly assigned to one of twelve treatment cells from the following:
(1) horizontal or vertical presentation format,
(2) single or multiple response mode,
(3) online, written, or null documentation mode.
The system constructed a file of all user responses so that we could re-create any experimental session. Primarily, we wanted to investigate whether any of the three treatments affected productivity (the number of tasks completed). We assumed that no subject would come close to completing all 45.
A FORMAL GRAMMAR CONSIDERATION OF THE EXPERIMENTAL TREATMENTS

We have constructed "action grammars" for the three experimental treatments. We can use these grammars to suggest what results the experiment should yield. Let us first consider the documentation treatment (specifically the written and online versions). In both instances a subject can interrupt his session, peruse the documentation, and resume problem-solving. The grammar below produces the written documentation action language for this activity:

```
getdoc ::= interrupt + peruse + resume
interrupt ::= TURNAWAY
peruse ::= READ | peruse + FLIP + peruse | NULL
resume ::= TURNBACK
```

The start symbol `getdoc` declares that this grammar describes only the process of "getting documentation". The other actions which the subject would take would be described in other grammars. The grammar given above is straight-forward and produces a fairly simple language. The terminals may be interpreted as follows: `TURNAWAY` is what the subject does when he turns his eyes (and attention) from the screen to the piece of paper documentation. He can then `READ` from the side of the paper that is showing and/or `FLIP` the paper over and `READ` from the other side. (The grammar reflects that a user could `FLIP` and `READ` several times. We assume that someone of normal intelligence will not operate in this manner. The same type of assumption has been made for the online language below.) The term `NULL` is not really a terminal; it represents that no action is taken.

Using this grammar, the following strings can be produced:

```
getdoc --> interrupt + peruse + resume
    --> TURNAWAY + peruse + resume
    --> TURNAWAY + READ + resume
    --> TURNAWAY + READ + TURNBACK
getdoc --> interrupt + peruse + resume
    --> TURNAWAY + peruse + resume
    --> TURNAWAY + peruse + FLIP + peruse + resume
    --> TURNAWAY + FLIP + peruse + resume
    --> TURNAWAY + FLIP + READ + resume
    --> TURNAWAY + FLIP + READ + TURNBACK
getdoc --> interrupt + peruse + resume
    --> TURNAWAY + peruse + resume
    --> TURNAWAY + peruse + FLIP + peruse + resume
    --> TURNAWAY + READ + FLIP + peruse + resume
    --> TURNAWAY + READ + FLIP + READ + resume
    --> TURNAWAY + READ + FLIP + READ + TURNBACK
```

Of course, there are other strings which the grammar can produce, but these seem the most realistic. Thus, the three usual activities by which the written documentation subject can obtain documentation are described by

- `TURNAWAY + READ + TURNBACK` (in which the user glances away, reads something from the exposed side of the paper and turns back),
- `TURNAWAY + FLIP + READ + TURNBACK` (the same except that he flips the paper over before reading), and
- `TURNAWAY + READ + FLIP + READ + TURNBACK` (in which he reads from both sides of the paper).
The three formalism measures suggested in the earlier section are applied to the language and these three most typical strings:
(1) This language has 4 terminals;
(2) These strings are 3, 4, and 5 terminals long; and
(3) It takes 4 or 6 rules to go from the start symbol to a terminal string.

We will contrast this with the grammar which produces the action language for the online version:

```
getdoc ::= interrupt + peruse + resume
interrupt ::= STRIKEI
peruse ::= consider + NEWLINE + consider
consider ::= READ | NULL
resume ::= STRIKEB + RESPOND | resume + STRIKEB + RESPOND | NULL
```

The terminal symbols may be interpreted as follows: STRIKEI (i.e., strike the character "I" on the keyboard) is how the user requests documentation. This produces the first screen of documentation. The user can either READ this screen or not (NULL). He must press the NEWLINE key to continue in either case after which he has the same READ or not alternative. The resumption process is a good deal more complicated than for written documentation. Documentation is scrolled onto the screen and information that was there before is scrolled off. Our system had no single command to return the screen contents to their earlier form. This left our subjects with a choice - they could either attempt to respond to a system question which they have somehow remembered, or they could use our "backup" feature STRIKEB (i.e., strike the character "B") to be prompted with the previous question. The system even allowed several consecutive backups of which some users took advantage. Thus, to resume, the subject can do nothing (NULL) or STRIKEB and RESPOND to the question; the latter two actions can be employed recursively.

Using this grammar, the following typical strings can be produced:

```
getdoc --> interrupt + peruse + resume
   --> STRIKEI + peruse + resume
   --> STRIKEI + consider + NEWLINE + consider + resume
   --> STRIKEI + READ + NEWLINE + consider + resume
   --> STRIKEI + READ + NEWLINE + READ + resume
   --> STRIKEI + READ + NEWLINE + READ + STRIKEB + RESPOND

getdoc --> interrupt + peruse + resume
   --> STRIKEI + peruse + resume
   --> STRIKEI + consider + NEWLINE + consider + resume
   --> STRIKEI + READ + NEWLINE + consider + resume
   --> STRIKEI + READ + NEWLINE + READ + resume
   --> STRIKEI + READ + NEWLINE + READ + resume + STRIKEB + RESPOND
   --> STRIKEI + READ + NEWLINE + READ
      + STRIKEB + RESPOND + STRIKEB + RESPOND
```
Thus, the typical activity in which the *online documentation* subject obtains documentation is described by

\[ \text{STRIKEI + peruse + resume} \]

(in which the subject interrupts his work, peruses the system-supplied documentation and then recovers as much of the original screen contents as desired).

The three formalism measures for the language and the three strings produced above:

1. This language has 5 terminals;
2. These strings are 6, 8, and 10 terminals long; and
3. It takes 6, 7, and 8 rules to go from the start symbol to a terminal string.

There are, of course, several other strings that are in this language. In general if the user wants to back up \( k \) questions and responses in order to resume processing, then it will take \( 5+k \) rules and \( 4+2k \) terminal symbols will be in the produced string. (Note: if the subject \text{READs} only one screen and not the other, then the number of terminal symbols is \( 3+2k \) but the number of rules is still \( 5+k \)).

Thus, string simplicity (\{3, 4, 5\} vs. \{6, 8, 10,...\}) and structural consistency (\{4, 6\} vs. \{6, 7, 8,...\}) favor *written* documentation. Based on this formal grammar analysis we arrive at the (perhaps nonintuitive) conclusion that it should take more actions (and thus more effort and time) to obtain documentation using the *online* system. Thus, we hypothesize that our subjects will be more productive with *written* documentation than with *online* documentation.

The *null* documentation grammar

\[ \text{getdoc ::= NULL} \]

raises a problem. On the surface, it certainly seems superior to *written* and *online* documentation in terms of number of terminal symbols, string simplicity, and structural consistency. But, it is not equivalent to the other two, because subjects receiving this treatment receive no documentation after the initial cryptic presentation. Thus, we have no formal hypothesis concerning how our null documentation subjects will perform relative to the others.

Now we consider the *horizontal* and *vertical* information presentation modes. The critical activity is that of finding relevant information from the screen contents. Recall that in the vertical mode individual elements occupy separate lines, while in the horizontal mode several elements may be "packed" onto one line. The grammar below produces the action language for this activity using *vertical presentation*:

\[ \text{findinfo ::= UPLINE + checkscreen} \]

\[ \text{checkscreen ::= scan | checkscreen + RESPOND + findinfo} \]

\[ \text{scan ::= FIND | UPLINE + scan | NULL} \]

The terminals may be interpreted as follows: We assume a simple searching procedure in which the subject raises his eyes from the last system message (which is always at the bottom of the screen) to the line directly above it; this is the action \text{UPLINE}. Then, he begins to check the screen contents line by line.
As he scans each line he either FINDs what he is looking for or continues UPLINEs until he has exhausted the screen contents. At that point, he may RESPOND to the system message leading to a new screen which he peruses in the same manner.

For comparison purposes, let us look at two strings (finding desired information 3 and 5 elements back):

findinfo --> UPLINE + checkscreen
   --> UPLINE + scan
   --> UPLINE + UPLINE + scan
   --> UPLINE + UPLINE + FIND

findinfo --> UPLINE + checkscreen
   --> UPLINE + scan
   --> UPLINE + UPLINE + scan
   --> UPLINE + UPLINE + UPLINE + scan
   --> UPLINE + UPLINE + UPLINE + UPLINE + scan
   --> UPLINE + UPLINE + UPLINE + UPLINE + UPLINE + FIND

This language has only 3 terminals. The first string is 3 terminals long and requires 4 rules; the second is 5 terminals long and requires 6 rules. In general, to check $k$ elements requires a string of length $k$ via $k+1$ rules.

We now contrast this with the grammar which produces the action language for the horizontal version:

findinfo ::= UPLINE + checkscreen
checkscreen ::= scan | checkscreen + RESPOND + findinfo
scan ::= FIND | ignore + scan
ignore ::= UPLINE | IGNOREELEMENT + ignore | NULL

Again the subject checks the screen contents line by line. But, the major difference is that each line may contain several elements. So he must take the action IGNOREELEMENT to ignore elements on a line as well as UPLINE to move up to the next line.

For comparison we now look at the two strings (finding desired information 3 and 5 elements back). If we assume that 3 elements may be packed onto a line, then the 3rd element is found on the first line up and the 5th element on the second line up:

findinfo --> UPLINE + checkscreen
   --> UPLINE + scan
   --> UPLINE + ignore + scan
   --> UPLINE + IGNOREELEMENT + ignore + scan
   --> UPLINE + IGNOREELEMENT +IGNOREELEMENT + ignore + scan
   --> UPLINE + IGNOREELEMENT +IGNOREELEMENT + IGNOREELEMENT + ignore + scan
   --> UPLINE + IGNOREELEMENT +IGNOREELEMENT + IGNOREELEMENT + scan
   --> UPLINE + IGNOREELEMENT +IGNOREELEMENT + IGNOREELEMENT + FIND
This language has 4 terminals. The first string is 4 terminals long and requires 7 rules; the second is 7 terminals long and requires 11 rules. Obviously, as long as the desired element is on the screen, the vertical mode seems better. Notice that when the current screen contents are inadequate, the rule checkscreen \rightarrow checkscreen + RESPOND + findinfo must be applied to put new information onto the screen. This must be done more often in the vertical mode than in the horizontal. Thus, there are circumstances when grammar and language considerations may favor the latter. On balance, however, we hypothesize that subjects will accomplish more in the vertical presentation mode.

Finally we consider the single and multiple response formats. The equivalent activity that can be performed with both is to make multiple responses to questions asked by the system. The grammar below produces the action language for this activity using single responses:

\[ \text{multresp ::= \text{RESPOND} + \text{SYSRESP} | \text{RESPOND} + \text{SYSRESP} + \text{multresp}} \]

The terminal symbols may be interpreted as follows: The user RESPONDS to any system question and then must wait for the interactive system to alter the screen contents and pose another question. This waiting action we term SYSRESP.

For comparison let us consider a situation in which the single response subject makes three consecutive responses:

\[ \text{multresp} \rightarrow \text{RESPOND + SYSRESP + multresp} \]
\[ \rightarrow \text{RESPOND + SYSRESP + RESPOND + SYSRESP + multresp} \]
\[ \rightarrow \text{RESPOND + SYSRESP + RESPOND + SYSRESP + RESPOND + SYSRESP} \]

This language has only 2 terminals. The string is 6 terminals long (2$k$ in general) and requires 3 rewriting rules ($k$).

We now contrast this with the grammar which produces the action language for the multiple response version:

\[ \text{multresp ::= \text{RESPOND} + \text{SYSRESP} | \text{RESPOND} + \text{multresp} + \text{SYSRESP}} \]
In this version the subject may make multiple responses, but still the system must respond in a sequential fashion to each by altering the screen and ultimately posing the same question as would be transmitted to the single version user.

For comparison, consider the same situation in which the multiple response subject makes three consecutive responses:

\[
\text{multresp} \rightarrow \text{RESPOND} + \text{multresp} + \text{SYSRESP}
\]
\[
\rightarrow \text{RESPOND} + \text{RESPOND} + \text{multresp} + \text{SYSRESP} + \text{SYSRESP}
\]
\[
\rightarrow \text{RESPOND} + \text{RESPOND} + \text{RESPOND} + \text{multresp} + \text{SYSRESP} + \text{SYSRESP} + \text{SYSRESP}
\]

The string again is 6 terminals long (2k in general) and requires 3 rewriting rules (k). Thus, using the formal grammar approach we must hypothesize that there will be no difference in our subjects’ performance with these two treatments.

RESULTS

All subject scores are shown in Figure 1. We have analyzed these scores using the Kruskal-Wallis Analysis of Variance [Hollander and Wolfe, 1973]. This analysis of variance deals with the ranks of the scores rather than the scores themselves. It requires only the assumption of ordinal scaling. The nonparametric measure of central tendency is the median.

Figure 1. Three Factor Experiment Scores

<table>
<thead>
<tr>
<th>Horizontal</th>
<th>Single</th>
<th>Multiple</th>
<th>Vertical</th>
<th>Single</th>
<th>Multiple</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null</td>
<td>8</td>
<td>9</td>
<td>Null</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>13</td>
<td></td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>15</td>
<td></td>
<td>15</td>
<td>18</td>
</tr>
<tr>
<td>Written</td>
<td>11</td>
<td>10</td>
<td>Written</td>
<td>10</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>12</td>
<td></td>
<td>14</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>13</td>
<td></td>
<td>17</td>
<td>22</td>
</tr>
<tr>
<td>Online</td>
<td>7</td>
<td>5</td>
<td>Online</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>6</td>
<td></td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>6</td>
<td></td>
<td>11</td>
<td>11</td>
</tr>
</tbody>
</table>

As we had hypothesized by using the formal grammar approach, our subjects did better with the vertical display mode (median score 12.0 as opposed to the horizontal median 10.5). However, this difference is not statistically significant. We think the reason that the difference is so slight is because in most instances where scanning up the screen was of any value, the desired element was not far away. Recall from the examples in the last section that if only a few elements must be skipped (ignored), then the differences in number of rules and terminal symbols (actions) are slight. Thus, our subjects performed in just about the way the formal grammar approach predicted.
There was virtually no difference in the median scores for response mode: Single = 11.0, Multiple = 10.5. The grammars did not reveal any string simplicity or structural consistency differences. Thus, these results were exactly as the grammar approach hypothesized.

The median scores for the documentation treatment appear below:

- Written = 13.5
- Online = 7
- Null = 12

Using the Kruskal-Wallis Analysis of Variance, this treatment is significant at the .05 level. Moreover, the written documentation vs. online documentation result is precisely what the formal grammar approach suggested. Our subjects were much more productive with written documentation than with equivalent online documentation. In retrospect, the null documentation result is not out of line. The null mode is more like the written mode since neither set of subjects could alter their screen contents by calling up documentation. Both appear (at least in this instance) superior to online documentation.

Notice the apparent anomaly in Figure 1's horizontal-single-online cell. Here we find two subjects with very good scores (15 and 24). Upon inspection of the retained files of subject responses, we found that neither of these ever requested online documentation after it was presented initially. Neither were really taking "advantage" of the online documentation available; it is not surprising that their scores did not suffer.

We were interested in whether these latter results would replicate. If so, we felt that this would be further support for the formal grammar approach. In the fall of 1980 we were able to obtain thirteen students in a graduate level Industrial Engineering course at Purdue University. We used only three versions of our experimental system. Subjects were randomly assigned to the three documentation modes, but all received "vertical" presentations and could only respond in the "single" mode. Each subject was given the same set of problems, same instructions, and same time limit as we employed in our earlier experiment. The scores from this experiment are shown in Figure 2.

Figure 2. Documentation Experiment Scores

<table>
<thead>
<tr>
<th></th>
<th>Null</th>
<th>Written</th>
<th>Online</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>13</td>
<td>14</td>
<td>7</td>
</tr>
<tr>
<td>14</td>
<td>17</td>
<td>17</td>
<td>10</td>
</tr>
<tr>
<td>15</td>
<td>18</td>
<td>22</td>
<td>15</td>
</tr>
</tbody>
</table>

The median scores appear below:

- Written = 17
- Online = 9.5
- Null = 13.5

Again, these results are significant at the .05 level. The written documentation vs. online documentation formal grammar hypothesis has been supported again.

Similar results were attained by Hansen et al. [1978] when a group of Computer Science students were required to take an examination both on paper and via the interactive system PLATO. The only significant results they observed was that it took about 60% longer to complete the exam with PLATO than on paper. Among the reasons they suggested for this difference were "uncertainty as to how to control the medium" (similarly, our subjects were probably more certain about how to deal with documentation on paper); "uncertainty as to what the system will do" (this might have affected our subjects as well); and
"uncertainty as to what the system has done" (in our case there was the obvious question as to whether rolling information off the screen to accommodate documentation changed one's position in the problem-solving route).

Kozar and Dickson [1978] compared cost and time performances of two groups of subjects making decisions in a simulated production environment. Both groups received the same information, one via paper, the other by means of a CRT. The "paper" group took significantly less time (23 vs. 30 minutes) to make equivalent decisions. The researchers speculated that the novelty of the CRT was largely responsible for this result, and that this difference might diminish in time.

**DISCUSSION**

Reisner [1979] proposed that an action language could be used to describe formally the human factors aspects of an interactive system and to compare alternative system designs. She introduced three measures (number of terminal symbols, terminal string length, and number of rules to produce strings) and used these to examine two interactive systems for creating color slides. Predictions based on formal grammar considerations were supported by empirical results.

Dunsmore [1980] is conducting research to investigate interactive systems for non-programmers. The work is empirical in nature - comparing subjects' performance under various alternative system designs. In spring 1980 we conducted an experiment with horizontal and vertical information display; single and multiple response modes; and written, online, and null documentation. Action grammars were constructed to describe each factor. Relative subject productivity was predicted based on these formalisms.

The written documentation language has one less terminal symbol than the online version. The longest typical written documentation string is five terminal symbols (i.e., actions) long; while the online documentation strings are 6, 8, 10, 12, and more. Structural consistency favors written documentation as well. Thus, the hypothesis was that written documentation subjects should be more productive than the others. This hypothesis was supported by the results from our spring 1980 experiment and from an earlier 1980 experiment as well. In both cases, written documentation subjects were almost twice as productive as their online counterparts.

In the vertical information display mode, to check back k elements requires k+1 rules and produces a string of length k terminal symbols (assuming the desired element is on the current screen). These values are lower limits for the number of rules and string lengths in the horizontal mode. We hypothesized that subjects would be more productive using the former. Our empirical results favored the vertical mode, but were not significant.

Some of our subjects could make multiple responses to a question asked by the system while others were constrained to single responses. Grammar and language formalisms had suggested there would be no difference in the two groups' performance. We could find no difference in the experimental subjects based on this factor.

We feel that the work reported in this paper further supports the notion that human factors experimentation can be more rigorous in nature. We propose that grammars and languages can be used to describe the human interface with systems or programming languages. Manipulation of the formalisms can then suggest features of nearly-equivalent systems or languages in which performances are likely to differ. Hypotheses concerning the direction and severity of difference can be constructed. These human factors can then be subjected to experimentation in attempt to confirm the hypotheses. This type of
experimentation should be more economical than the current style of examining virtually every potential human factor with hypotheses supplied only by the experimenter's intuition.

ACKNOWLEDGEMENTS

The grammar and language work is based on the research of Dr. Phyllis Reisner, IBM Research Laboratory, San Jose, California. The author is indebted to Dr. Reisner for her help, ideas, and encouragement in this work. Valuable comments and suggestions have also been provided by John Thomas, Robert Tabor, and Sam Matsa. Andre Bondi and Charles Amacher supervised the interactive system experiments. This research is supported by the International Business Machines Corporation.

REFERENCES

[Brooks, 1980]

[Campbell and Stanley, 1963]

[Card et al., 1980a]

[Card et al., 1980b]

[Dunsmore, 1980]

[Embley 1978]

[Embley et al., 1978]

[Foley and Wallace, 1974]
[Foley, 1978]

[Hansen et al., 1978]

[Hollander and Wolfe, 1973]

[Kozar and Dickson, 1978]

[Moran, 1978]

[Reisner, 1979]
Reisner, P. Using a formal grammar in human factors design of an interactive graphics system. *IBM Research Report* RJ2505 (32755) (April 11, 1979), IBM Research Laboratory, San Jose, California 95193.

[Shneiderman, 1980]

[Thomas, 1978]