Improving the encoding of verbal reports by using MPAS: A computer-aided encoding system

ROBERT J. CRUTCHER
University of Illinois, Chicago, Illinois

K. ANDERS ERICSSON
Florida State University, Tallahassee, Florida

and

CHRISTOPHER A. WICHURA
University of Illinois, Chicago, Illinois

The dramatically increased use of verbal report methodologies in psychological research has created a need for new tools to improve the efficiency and reliability of encoding these data. A computer-aided protocol encoding system called MPAS (Multiple Protocol Analysis System) presents individual protocol segments in a randomized order to one or more coders and then stores computer keyboard-entered codes for later output to an SPSS formatted data file. In the present paper, MPAS is described, and a brief example is provided of how MPAS can be used in a controlled laboratory study to rigorously analyze verbal protocol data.

In recent years, the use of verbal report methodologies in psychological research has increased dramatically. Verbal reports are now collected across diverse task domains, and a number of researchers have argued that given the high temporal density of protocol data, verbal reports are important in the evaluation of psychological theories (Anderson, 1987; Ericsson & Simon, 1993; Simon & Kaplan, 1989). Given this increased interest in verbal report methodologies, it is crucial that the methods for collecting and analyzing verbal reports be as rigorous as possible. A number of concerns have been dealt with elsewhere, such as the importance of task analysis and of establishing clearly defined rules for coding individual protocol segments (Ericsson & Simon, 1993). Here we are mainly concerned with the use of computer-aided tools in improving the encoding process. Specifically, we describe a new computer tool called MPAS (for Multiple Protocol Analysis System)—a general-purpose tool designed to improve the efficiency and reliability of coding a large number of protocols.

One way to increase the objectivity and reliability of encoding is to try to automate the encoding process so that it can be accomplished entirely by a computer program. The PAS-I system of Waterman and Newell (Waterman & Newell, 1971) represents the most ambitious effort in this direction. In this program, a problem space is first defined to delimit the possible encodings that can be derived from the protocol statements, and the protocols are then segmented manually (e.g., by syntactic units) in preparation for coding. The segments are processed by a limited linguistic parser, mapped onto a set of propositional relations, and reduced to a smaller set of semantic primitives. These primitives are eventually realized as a problem behavior graph displaying the successive knowledge states of a given subject and the productions executed in moving from one state to the next.

The main advantage of the automated encoding approach is that writing such programs requires that the underlying assumptions, encoding vocabulary, and inference rules be explicitly defined and consistently applied, so that a given protocol will be encoded similarly regardless of the particular context. However, such systems have disadvantages. First, there is the difficulty of developing linguistic and semantic parsers. Although it is feasible in limited and well-defined domains, the development of parsers in more complex domains can be difficult and expensive. Second, many protocol segments consist of incomplete sentences and ungrammatical constructions that would challenge even the most sophisticated parser. Finally, given that such parsers are necessarily limited to the domain for which they have been developed, automated programs do not have the widest applicability.

This research was supported in part by a subcontract between the University of Illinois at Chicago and the University of Colorado at Boulder as part of a larger ARI contract award (MDA903-93-K-0010) to the University of Colorado. A previous MPAS (for Mini Protocol Analysis System) was described by Ericsson and Simon (1984) and since then has been modified a number of times. The current system, also called MPAS (for Multiple Protocol Analysis System to emphasize a key feature of the program), is a completely new version of the program. It is written in C to ensure maximum portability, and it is designed to run on small computer systems such as the Macintosh and IBM PC clones. Inquiries regarding availability of the program as well as other correspondence should be addressed to R. J. Crutcher, Department of Psychology (MC 285), University of Illinois, 1009 BSB, 1007 West Harrison Street, Chicago, IL 60607-7137 (e-mail: rcrutcher@uic.edu).

An alternative approach to automated encoding systems consists of semiautomatic encoding systems in which one tries to improve the objectivity, reliability, and efficiency of human coders with the aid of computer encoding tools (Bhaskar & Simon, 1977; Ericsson & Simon, 1993; Fisher, 1988; Sanderson, James, & Seidler, 1989). For example, after encountering a number of problems with their automatic system (PAS-I), Waterman and Newell (1973) developed an interactive version of the system (PAS-II) that relied on human coders. In semiautomated systems, the responsibility for encoding decisions is divided between the human coder and the computer program, with the relative division of responsibility varying from one system to another. At one extreme, the program may have considerable knowledge of the domain and take major responsibility for generating encoding decisions, with the human coder serving merely to confirm or deny and perhaps revise encoding decisions made by the program. At the other extreme, the program may have little or no knowledge of the encoding domain, and it is responsible only for presenting individual protocol segments and perhaps cuing the coder with possible encoding categories, leaving all important aspects of encoding decisions to the human coder.

SAPA (Bhaskar & Simon, 1977) is a good example of a semiautomatic system in which the domain knowledge is embodied in the program itself. SAPA embodies a rough model of the subject's problem-solving strategy so that as each protocol segment is input it prompts the coder by predicting the next operation that the subject will apply. The coder then has the option of correcting the program's decision and returning the program to the correct path. Examples of semiautomatic systems in which no specific domain knowledge is contained in the program itself are SHAPA (Sanderson et al., 1989) and the MPAS program described here. These systems aid in the encoding process by focusing the coder's attention on individual segments one at a time and in some cases by cuing possible interpretations of a given segment. The advantage of systems like SHAPA and MPAS is that they have wider applicability because the domain-specific knowledge is not built into the system. In addition, the difficult task of developing language parsers to handle the protocol segments is not required. The disadvantage of such systems is that often they cannot entirely eliminate all of the subjective aspects of coding. Whereas a machine program can be designed to encode a given segment in the same way regardless of the surrounding context, a human coder's encoding may be influenced by context. For example, in coding a specific segment, the coder may recall information concerning previous segments from the same subject or experimental treatment. Furthermore, whereas an automated program can code hundreds or even thousands of protocol segments without tiring, there are obvious limitations on the capacity of human coders, especially if the mechanical aspects of the coding process are difficult. Finally, whereas more automated programs can easily apply specific rules consistently, human coders

may fail to apply coding rules consistently unless the rules are explicit and can be applied to the protocols in a straightforward fashion. We want to stress, however, that these disadvantages can be minimized if semiautomatic systems are properly implemented and if the principle of localized encoding, described below, is followed.

Using MPAS to Implement the Localized Encoding Principle

MPAS is a general-purpose, semiautomatic encoding system whose primary goal is to implement the localized encoding principle and improve the efficiency with which large numbers of individual protocol segments are encoded. The use of context in coding protocol segments always involves a tradeoff. On the one hand, interpreting a segment in terms of the surrounding context may sometimes permit a more complete and accurate description of the meaning of the segment. On the other hand, using the surrounding context violates independence assumptions and reduces the number of degrees of freedom available to analyses. A segment that has been used to disambiguate another segment cannot be considered an independent observation. Naturally, the decision as to how much context to employ in making coding decisions must be determined by the nature of the task domain; theoretical coding categories should capture enough of the important psychological processes to be useful in making empirical predictions. In some cases, it may be necessary to sacrifice some number of degrees of freedom in order to adequately capture a specific phenomenon. However, whenever possible, it is preferable to code individual segments without reference to the surrounding context. This, referred to as the *localized encoding principle*, is a key feature of MPAS.

The localized encoding principle is evident in the design and operation of the MPAS program. First, MPAS is designed so that the coder is forced to code each protocol segment independently, without reliance on any other contextual information. This is accomplished by storing all of the protocol segments for a set of subjects in an experiment in a single file that is then encoded with the use of MPAS. MPAS randomly selects individual protocol segments from the pool of segments, presenting them one at a time, in random order, to the coder. Although information concerning the identity of each protocol segment (e.g., subject number, condition, etc.) may be stored in the raw protocol file along with the actual transcribed protocol, this information is hidden during the actual encoding process.

Another way in which the MPAS program supports localized encoding relates to the improved efficiency of protocol encoding that it enables. Because of this efficiency, it becomes practical to actually code many more protocol segments from a larger pool of subjects than would be possible without such a program. As the number of subjects and segments increases, the likelihood of a coder remembering and using information from previous segments in coding a current segment decreases.

Finally, with respect to the earlier point concerning the limitations of human encoders in applying coding rules consistently, it should be noted that the reliability of semi-automated encoding systems like MPAS can be improved considerably by using well-defined tasks. With such tasks, a careful task analysis can specify all possible sequences of thoughts that subjects might be expected to employ in generating target responses. As a result, the coding rules should fall out of the task analysis and be much more explicit (see Ericsson & Oliver, 1988, for additional discussion concerning these matters).

MPAS: A General Description

MPAS presents individual protocol segments from a file containing an entire set of protocols for an experiment. Prior to coding, the protocols for all subjects must be transcribed, segmented, and stored in a plain ASCII text file. The individual protocols are formatted so that each protocol is preceded by an asterisk and optionally preceded by a header on the same line as the asterisk. This header typically contains information such as the subject number, experimental condition, and trial number, and will be invisible during the encoding process (see Table 1). Each protocol consists of one or more segments, with each segment terminated by a backslash. In some cases, an entire protocol may constitute a single segment, so it will be necessary only to place an asterisk at the beginning of the protocol and a backslash at the end of the protocol. This was the case with the protocols that we analyzed in our vocabulary studies (described in the next section). During the operation of the coding program, each segment is presented independently and in random sequence so that the coder does not know from which subject, condition, or protocol the segment came.

Once the master protocol file containing the raw protocols and the header information is generated, the protocol segments are coded with the CODING program. Upon startup, the CODING program asks for the filename in which the protocols are stored and for the coder's number. MPAS permits two coders to code the same protocol independently and maintains a separate index file for each coder, keeping track of each coder's progress through the

file of protocols. When both coders are finished, the two sets of codings can be compared and a reliability index generated. After the filename and coder information have been entered, the program enters command mode. To begin coding, the word NEXT or a carriage return is entered, and the program randomly selects a protocol segment from the file and presents it to the coder. All of the information preceding the asterisk (e.g., the subject number, trial number, and experimental treatment) is hidden from the coder. For each segment coded, a character string of up to nine characters is entered, followed by the return key, and the program automatically presents the next segment to be coded. This process continues until all segments are coded or until the coder wants to take a break, at which point all work up until that point can be saved by first returning to the command mode (accomplished by entering a carriage return instead of a code) and typing EXIT.

As noted earlier, upon initial program startup and after entering filename and coder information, the program is in command mode. During the coding process, one can return to command mode by pressing the return key without entering a code. In command mode, a number of options are available. The INFO command provides information about how many segments have been coded and how many are left to code. The UNDO command returns the coder to the last protocol segment coded and permits changes to the last code entered—in the event of a typing error, for example. The SAVE command saves all the coder's work up to the point that the SAVE command has been issued; this command should be used periodically. The EXIT command performs a SAVE and then quits the program. This is useful in coding large files, allowing one to code part of a large file and then quit at some convenient point, saving all work up until that point. Upon restarting, the program consults the index file and returns to the same point in the coding sequence as that at which the program quit. The QUIT command permits quitting the program without saving the current session's work.

In addition to the CODING program, MPAS has a COLLECT program that can be used to generate three different types of output files from the encoded protocols. First, a flat ASCII text file containing the original protocol segments along with each coder's codings can be generated. Second, an output file suitable for analysis by SPSS or other statistical programs can be generated. Selecting this option generates an output file with the following format: on each line are 31 + 13 + 13 + 21 + 21 characters, the first column containing the header information, the second column containing the protocol number, the third column the segment number within a protocol, the fourth column containing coder number one's codes, and the fifth column containing coder number two's codes. Assuming the relevant information concerning treatment conditions, trial numbers, and so forth has been stored in the headers preceding each protocol, this file is ready for analysis with SPSS or some other statistical package. Additionally, the SPSS file generation

Table 1
Examples of How to Format Protocol Files in Preparation
for Encoding with MPAS

Multi-Segment Protocol Example:

Optional Header Information *

```
segment 1\
segment 2\
segment 3\
segment final\
```

Single Segment Protocol Example:

```
8 1 1 1 27 24 32 1 * doronico door leopard
Saw the Spanish word 'doronico'
Remembered 'door'
And then the 'leopard' leaping
Through the door\
```

option indicates how many segments have been coded and the number of segments on which the codes for the two coders match. A third type of output file that can be generated is a frequency file that contains various file statistics, such as the total number of lines, protocol segments, words, and characters. This last feature was designed to be part of a more powerful system that has not been implemented yet.

Using MPAS to Analyze Verbal Report Data from a Laboratory Study

We now briefly describe the use of MPAS in analyzing some protocol data from an actual laboratory study that we conducted. We hope with this example to illustrate how, with the aid of a program like MPAS, verbal protocol data can be aggregated and analyzed as objectively as traditional performance measures such as response accuracy and latency data. The task of interest in our study was the retrieval of vocabulary pairs learned by using keyword mediators. For example, the Spanish-English pair *doronico-leopard* can be learned by using the keyword mediator *door*, a word that sounds similar to the Spanish word *doronico* and that can be related to the English equivalent *leopard* by forming an interactive visual image between door and leopard (e.g., *a leopard walking through a door*).

To test the retrieval of these vocabulary pairs, each Spanish word was presented on a computer display and subjects were asked to recall the English equivalent of that word by saying it into a microphone that was connected to a voice key that recorded the retrieval response latency. On many of the trials, a retrospective verbal report was requested immediately after the response. Although we employed a number of other tasks and methodologies in trying to describe the underlying memory retrieval processes, we will restrict our analysis here to the vocabulary retrieval task itself, focusing on relating the analysis of the verbal report data to the other performance data—specifically, the latency data.

The main theoretical issue addressed in this particular study was how the mediators used to learn the Spanish-English pairs are involved in memory retrieval processes initially as well as after extensive retrieval practice. We hypothesized that initially retrieval would depend on using the keyword mediators as intermediate retrieval cues to recover the English definitions via the interactive images, whereas after sufficient retrieval practice on the Spanish-English pairs, this process might be replaced by a direct retrieval process that did not involve the mediators. To test this hypothesis, we collected a variety of dependent measures, including trial-by-trial verbal reports, to monitor the involvement of the mediators in the memory retrieval process.

Subjects learned 32 Spanish-English vocabulary pairs and were tested on retrieval of the pairs immediately. They then received specific retrieval practice designed either to alter potentially the nature of the retrieval process from mediated retrieval to direct access (experimental condi-

tion) or to maintain mediated retrieval (control condition). Subsequent to the extended practice, retrieval of the Spanish-English pairs was retested. The data set consisted of 128 trials for each of 8 subjects, for a total of 1,024 retrieval trials. For half of these trials, retrospective verbal reports were collected, yielding 512 trials for which verbal report data could be analyzed.

A master protocol file was then created in preparation for encoding the protocols. The original data files for each subject were edited as recommended in the previous section on using MPAS, so that all of the relevant trial information (such as the subject number and experimental treatment) were formatted on a single line, followed by an asterisk to indicate the start of a protocol. Since the protocols here were generally quite short, it was possible to code an entire protocol as a single segment and therefore employ a single backslash at the end of the protocol for each trial (see the second example in Table 1).

The entire set of 512 protocols was coded across multiple coding sessions in approximately 6 h. As recommended earlier, a thorough task analysis was carried out prior to running the subjects, and an encoding scheme was developed in which five variables were employed to code the presence or absence of the following elements: the *Spanish word*, the *keyword*, the *English equivalent*, the *keyword-English link* (e.g., an interactive image or semantic relation), and the *retrieval sequence*. The details of this coding scheme are discussed elsewhere (Crutcher, 1992; Crutcher & Ericsson, 1992). Here, we note only that for each of these encoding variables a preestablished set of coding rules determined how each variable was encoded and that the coders were instructed to justify each coding decision with one of the rules from a provided list. For example, to code the presence of a keyword-English link, a specific relation linking the keyword and English definition was required (e.g., ‘‘I saw a leopard *walking through the door*’’). With the use of MPAS and these preestablished coding categories, interrater reliability was 96% or better for each of the five variables.

With the five encoding variables, the following distinct categories of retrieval were generated: (1) *mediated retrieval*—Spanish word and English equivalent present and keyword mentioned; and (2) *unmediated or direct retrieval*—Spanish and English definition present, but keyword not mentioned and no reference to any intermediate steps of any kind. An analysis of the reports in conjunction with our other task methodologies clearly showed that mediated retrieval is the norm immediately after acquisition of vocabulary pairs (as is indicated by reported keyword mediation on nearly every retrieval). However, after sufficient practice, subjects were able to retrieve the English definitions from the Spanish word cues in a single step (as indicated by an almost total absence of reported mediation). This interpretation was supported not just by the verbal report analysis but by other analyses, such as a task decomposition (see Crutcher, 1992, for details). Most importantly, we were able to relate the verbal report measures to our latency data systematically.

An analysis of the different categories of reported mediation and the retrieval latencies, for example, showed that as the number of mediating steps increased, the retrieval latencies increased as well. This analysis was even more compelling in light of previous studies that we had conducted in which we demonstrated the same correspondence for additional categories of mediation. For example, Crutcher (1990) examined categories of reported mediation that included *keyword + image mediation*, *keyword mediation*, or *no reported mediation*, and again found that retrieval latencies became longer as the number of reported mediating steps increased.

Conclusions

As a tool for improving the encoding of verbal protocol data in psychological research, we think MPAS has great potential, especially if used in conjunction with well-defined tasks and coding schemes based on careful task analysis. Our success in using MPAS to analyze the verbal report data in our vocabulary studies suggests the potential of semiautomatic encoding tools, such as MPAS and SHAPA, for analyzing verbal report data as objectively and rigorously as the more traditional cued recall accuracy and response latency data are analyzed. MPAS made it possible to code a large set of protocols while adhering to the localized encoding principle. Given the coding efficiency of MPAS, it was possible to collect verbal reports on a trial-by-trial basis and later to aggregate the protocol data and systematically relate them to the other performance measures. In addition, our intercoder reliability was extremely high. This was due in part to the efficiency of coding the protocols with MPAS, as well as to the rigor of the encoding scheme, which in turn was made possible by a careful task analysis.

In conclusion, we think that MPAS represents a powerful tool for coding and analyzing verbal protocol data by ensuring that localized encoding is employed in the encoding process. In the future, we hope to extend the capabilities of MPAS in various ways. In particular, we would like to have MPAS present the coding categories and rules to the coder and require the coder to justify each coding

by a specific rule choice. This would provide a trace of the rules used for each encoding decision and provide a further check on the adequacy of the coding rules developed by the experimenter.

REFERENCES

- ANDERSON, J. R. (1987). Methodologies for studying human knowledge. *Brain & Behavioral Science*, **10**, 467-505.
- BHASKAR, R., & SIMON, H. A. (1977). Problem solving in semantically rich domains: An example from engineering thermodynamics. *Cognitive Science*, **1**, 193-215.
- CRUTCHER, R. J. (1990). *The role of mediation in knowledge acquisition and retention: Learning foreign vocabulary using the keyword method* (Tech. Rep. No. 9010). Boulder: University of Colorado, Institute of Cognitive Science.
- CRUTCHER, R. J. (1992). *The effects of practice on retrieval of foreign vocabulary learned using the keyword method*. Unpublished doctoral dissertation, University of Colorado, Boulder.
- CRUTCHER, R. J., & ERICSSON, K. A. (1992, November). *Mediation processes in memory retrieval before and after extended retrieval practice*. Poster presented at the meeting of the Psychonomic Society, St. Louis.
- ERICSSON, K. A., & OLIVER, W. (1988). Methodology for laboratory research on thinking: Task selection, collection of observations, and data analysis. In R. J. Sternberg & E. E. Smith (Eds.), *The psychology of human thought* (pp. 392-428). New York: Cambridge University Press.
- ERICSSON, K. A., & SIMON, H. A. (1984). *Protocol analysis: Verbal reports as data*. Cambridge, MA: MIT Press.
- ERICSSON, K. A., & SIMON, H. A. (1993). *Protocol analysis: Verbal reports as data* (rev. ed.). Cambridge, MA: MIT Press.
- FISHER, C. (1988). Advancing the study of programming with computer-aided protocol analysis. In G. Olson, E. Soloway, & S. Sheppard (Eds.), *Empirical studies of programmers: 1987 workshop* (pp. 198-216). Norwood, NJ: Ablex.
- SANDERSON, P. M., JAMES, J. M., & SEIDLER, K. S. (1989). SHAPA: An interactive software environment for protocol analysis. *Ergonomics*, **32**, 1271-1302.
- SIMON, H. A., & KAPLAN, C. A. (1989). Foundations of cognitive science. In M. I. Posner (Ed.), *Foundations of cognitive science* (pp. 1-47). Cambridge, MA: MIT Press.
- WATERMAN, D. A., & NEWELL, A. (1971). Protocol analysis as a task for artificial intelligence. *Artificial Intelligence*, **2**, 285-318.
- WATERMAN, D. A., & NEWELL, A. (1973). PAS-II: An interactive task-free version of an automated protocol analysis system. In *Proceedings of the Third IJCAI* (pp. 431-445). Menlo Park, CA: Stanford Research Institute.