SESSION VII

COMPUTERS IN THE ANALYSIS OF LANGUAGE AND INTERPERSONAL INTERACTION

Amy Herstein Gervasio, *Presider*

A propositional analyst's assistant

GUY GROEN, CARL FREDERIKSEN, and MICHAEL DILLINGER
McGill University, Montreal, Quebec, Canada

A system (PAN) that is designed to assist both the inexperienced and the experienced user in performing propositional analysis is described. The system is implemented in UCSD Pascal on the Apple II and Apple IIe microcomputers.

A major problem with systems of propositional analysis such as those of Frederiksen (1975) and Kintsch (1974) is that they are extremely difficult to use. There are a number of reasons for this. First, the existing documentation is inadequate. Second, the methods involve the application of an extremely large number of detailed rules. Third, a large amount of housekeeping is necessary.

This paper describes a system, which we call PAN, that is designed to assist both the inexperienced and the experienced user in performing a propositional analysis. We should emphasize that it is an assistant rather than an automated analyst. More than anything else, it facilitates housekeeping and the application of the rules. For the beginner, it can also serve as a teaching aid. The system is implemented in UCSD Pascal on the Apple II and Apple IIe microcomputers. Although developed specifically in terms of Frederiksen's (1975) model, parts of it have more general applicability to other kinds of protocols and to other propositional analysis techniques. In describing it, we will proceed from the general to the specific.

In order to understand what the system is attempting to do, it will help if we begin by briefly considering the task that an individual is faced with when performing a propositional analysis by hand. First, the text must be broken up into numbered segments. This is done on the basis of Winograd's (1972) classification of surface structure into major and secondary clauses, with the major clauses usually (but not always) forming separate

segments. Then, each segment is analyzed for the propositions it contains. Basically, a proposition is an *n*-ary predicate (a truth-valued function with *n* arguments). As written, it consists of a head element, which is essentially the "name" of the proposition, followed by its arguments.

This is illustrated in Table 1, which is taken from an analysis by Patel and Frederiksen (1984) of text used as a stimulus in a free-recall experiment. Each segment is followed by a list of the propositions it contains. The propositions are numbered by segment. Thus, 1.1 is the first proposition of Segment 1. This is followed by the head element of the proposition, which is in turn followed by the arguments. Thus, in Proposition 1.1, PRESENT is the head element and there are three arguments. Unfortunately, the rules by which these are generated are too complex to be given in detail, and a brief summary would be misleading. A complete exposition is given in Frederiksen (1975). Fairly extensive summaries are to be found in Frederiksen (1979) and Tierney and Mosenthal (1982).

This is an example of what might be called a direct analysis, in which text is directly decomposed into propositions. With empirical protocols, a more indirect method, in which each segment is analyzed in terms of the propositions recalled or inferred from the original text, may be appropriate. This is called a recall/inference analysis. The difficulties in these methods are somewhat different. The basic problem with the direct method is in ensuring that the correct rule is being applied (and not misapplied). The basic problem with the indirect method is keeping track of the correspondences between the segments in the data and the propositions in the stimulus.

The authors' mailing address is: Centre for Medical Education, McGill University, Montreal, Quebec, Canada H3G 1Y6.

Table 1
An Example of Propositional Analysis

A 31 YEAR OLD MAN PRESENTS WITH SEVERE LEFT-SIDED CHEST PAIN.	
1.1 PRESENT	PAT: MAN, ACT: WITH, 1.3 = TENS: PRES;
1.2 MAN	ATT: OLD, DEG: 31, POS:
1.3 PAIN	= ATT: CHEST, ATT: LEFT-SIDED;
HE WAS PREVIOUSLY WELL UNTIL 5 DAYS EARLIER.	
2.1 BE	PAT: HE = ATT: WELL, TEN: PREVIOUSLY,
2.2 (OP): TEN	TEN: UNTIL, TNS: PAST;
2.3 DIFF: 5 DAYS: TEN	(2.1), (1.1), (PREVIOUSLY)
	(2.1), (1.1): EARLIER)
WHEN HE DEVELOPED FEVER, NON PRODUCTIVE COUGH, MYALGIA AND FATIGUE.	
3.1 DEVELOP	PAT: HE, RSLT: FEVER, 3.2, MYALGIA,
3.2 COUGH	FATIGUE = TEN: (WHEN, TNS: PAST;
3.3 EQUIV: TEN	= ATT: NON PRODUCTIVE;
	(2.1) (UNTIL), (3.1);
HE INITIALLY SAW HIS FAMILY DOCTOR.	
4.1 SEE	PAT: HE, OBJ: DOCTOR = TEN: INITIALLY, TNS:
4.2 DOCTOR	PAST;
4.3 (POSS)	ATT: FAMILY;
	PAT: HIS, OBJ: DOCTOR =;
AND HE WAS TREATED WITH A FIVE-DAY COURSE OF SEPTRA (WHICH HE IS PRESENTLY FINISHING).	
5.1 TREAT	REC: (AND) = CAT: WITH, 5.2, TNS: PAST;
5.2 COURSE	INST: SEPTRA = DUR: 5 DAY;
5.3 (WHICH)	PAT: HE = TEN: PRESENTLY, TNS: PRES,
	ASPECT: CONT, ASPECT: COMPL (FINISH);
GRADUALLY, THESE SYMPTOMS RESOLVED.	
6.1 RESOLVE	SOURCE: (THESE) SYMPTOMS = ATT:
	GRADUALLY, TNS: PAST;
WHILE SITTING AT WORK	
7.1 SIT	PAT: (SIT) = LOC: AT WORK, TEN:
	WHILE, ASPECT: CONT;
HE DEVELOPED A SEVERE SHARP CHEST PAIN (WHICH INCREASED ON INSPIRATION AND COUGHING)	
8.1 DEVELOP	PAT: HE, RSLT: 8.2 = TEN: , TNS: PAST;
8.2 PAIN	= ATT: CHEST, ATT: SHARP, DEG: SEVERE;
8.3 INCREASE	ACT: (WHICH) = TNS: PAST;

THE PAN SYSTEM

The system is menu driven with a tree-structured set of commands. There are three main subprograms, all of which are entered from the top level of PAN via a menu. SEG allows the user to transform raw text into labeled numbered segments and stores these segments on a master file. RECALL is an aid in analyzing segmented recall protocols. PROP is designed to aid in direct propositional analysis. There is another program, UTIL, which is used mainly to change certain properties of the other programs. SEG has been in existence for 1 year and has been used extensively by graduate students. Because the other programs use the master file created by SEG, we delayed implementing them until some experience had been gained with SEG. However, trial versions of RECALL and PROP are now running.

The user entering SEG either can edit the master file, altering segments or portions of segments, or else can create new segments out of raw text. In the latter case, the text is displayed on the screen. The user moves the cursor from the top left of the display to the position where the segment is desired to end. Either a C or a B is then pressed, standing for Create and Bury, respectively. If B is pressed, then the segment is entered on the master file without a number. If C is pressed, then an appropriately numbered segment is entered in

the master file. It is possible to enter the Winograd (1972) clause type (although this is not usually necessary in practice). An optional field is also reserved for general comments. This can have labeled subfields, which can be created or deleted by means of the UTIL program. Thus, quite arbitrary labels can be used, and this gives this program some potential generalizability outside the domain of propositional analysis.

The design of PROP is based on the fact that Frederiksen's (1975) model has a well-defined syntax that is representable as a set of augmented transition networks (ATNs) or, equivalently, as a formal grammar in Backus-Naur form (BNF). These are the standard techniques for defining the syntax of programming languages. As a result, there exist well-known algorithms for syntax checking (e.g., Wirth, 1976) that are easy to implement. Although developed for programming languages, they can be used for any system of notation with a formal definition of this type.

Using these algorithms, PROP can straightforwardly check the syntax of the propositional analysis, printing an appropriate message when an error is discovered. However, it can do considerably more. The BNF definition essentially generates a tree. The propositional analysis of a segment is some kind of traversal of this tree. For each nonterminal node, PROP contains a set of commands that tell the system which node to visit next.

The use of this feature is best illustrated by an example. Suppose the user wishes to analyze the first segment of Table 1. The computer displays the segment, and the following dialogue then ensues.

SEGMENT 1

A 31 YEAR OLD MAN PRESENTS WITH SEVERE LEFT SIDED CHEST PAIN

HEAD ELEMENT?

The user types PRESENT.

S STATE E EVENT A ALGEBRAIC RELN

D DEPENDENCY RELN I IDENTITY RELN
P PROPOSITIONAL RELN

The user presses E.

P PROGRESSIVE CASE FRAME R RESULTIVE CASE
FRAME

I IDENTIFYING RELATION

The user presses P.

P PATIENT O OBJECT I INSTRUMENT A ACT
R RECIPIENT

S SOURCE R RESULT G GOAL T THEME

The user presses P.

1.1 PRESENT PAT: (the current state of the analysis)

T TEXT G GO DEEPER

The user presses T and types MAN.

1.1 PRESENT PAT: MAN, (note the automatic insertion of the comma)

O OBJECT I INSTRUMENT A ACT R RECIPIENT

S SOURCE R RESULT G GOAL T THEME

The system has gone up one level in the tree. However, the PATIENT prompt no longer appears because the syntax forbids it. The user continues with the propositional analysis by pressing A, upon which a new prompt appears.

This sample dialogue should suffice to illustrate the main features of the interactive process. It is designed to be user friendly and to have an extremely fast response time. It is also semiautomated, in the sense that it inserts as many keywords and delimiters as possible. This does not result merely in keystroke economy. The parentheses, colons, commas, and equal signs scattered so liberally throughout Table 1 are the major source of error in performing a propositional analysis.

RECALL is designed to assist in recall/inference coding. In applying this technique (by hand) in a free-recall experiment, a propositional analysis is performed on the text used as stimulus material. However, the recall protocols are only segmented. The analysis is performed by taking each proposition in turn and comparing it with the segments in a recall protocol. In doing this, an attempt is made to match the elements of the surface structure that appear in the proposition (e.g., MAN in Proposition 1.1 of Table 1) with text in the recall segments. An identical match is a recall. Partial matches may be recalls or inferences, depending, once again, on the application of a complicated set of rules that will not be discussed here.

Because RECALL is the least developed of the three subprograms, our description will be brief. In its current implementation, it performs two functions. First, it serves as a record keeper for a hand analysis. For each proposition in the master file, it can keep track of segments in which the proposition is recalled or inferred. Second, it can be used to search for segments in which surface structure elements of a proposition appear, thus saving some of the labor involved in the coding process.

DISCUSSION

The PAN system was developed with two purposes in mind. The first was to provide a friendly assistant for

a laborious process of analysis. The second was to explore the extent to which UCSD Pascal could be used to develop a very large nontrivial system on a 64K microcomputer.

In this paper, we have focused on the first aspect. Of the three subprograms discussed, SEG fulfilled our goal. Judging from user comments, it seems to be of genuine assistance in the process of propositional analysis. PROP is too new to have been extensively used, but shows promise. RECALL is still under development, so we do not currently have a realistic estimate of how useful it will be. In the future, we would like to develop it so that it is capable of handling synonyms and certain aspects of inference coding. However, it is likely that we will devote most of our effort to the development of PROP. In a certain sense, this subsystem "knows" the syntax of propositional analysis much better than any human being. We suspect that this could have considerable value in exploring the reliability of the technique and also in expanding its utility to certain areas such as problem solving (Patel, Frederiksen, & Groen, 1983) that lie outside the traditional domain of comprehension research.

This may, of course, be beyond the resources of a small microcomputer. However, we have been surprised at the extent to which the system has not been plagued by memory-space limitations or excessive response times. This is almost certainly due to the decision to use the UCSD system as a development tool. It makes such extensive use of memory-swapping routines that it comes close to turning the Apple into a virtual-memory machine. Although one might expect this to slow down the response time, this is not the case. At some point, we will no doubt encounter problems of this type, but we cannot currently predict with any confidence when or why they will occur. However, once the subsystems of PAN described in this paper are reasonably stable, we intend to attempt to implement it on other microcomputers. In doing this, we intend to concentrate on the IBM PC and the new Apple Macintosh. Implementation on the IBM PC should be relatively simple, partly because of the well-known portability of UCSD Pascal and partly because it is possible to transmit files from the Apple to the IBM PC via the McGill mainframe computer. Implementation on the Macintosh may involve a considerable amount of reprogramming. However, this may be worthwhile because its window system may be very useful in clarifying the process of propositional analysis.

REFERENCES

- FREDERIKSEN, C. H. (1975). Representing logical and semantic structure of knowledge acquired from discourse. *Cognitive Psychology*, 7, 371-458.
- FREDERIKSEN, C. H. (1979). Discourse comprehension and early reading. In L. B. Resnick & P. A. Weaver (Eds.), *Theory and practice of early readings*. Hillsdale, NJ: Erlbaum.
- KINTSCH, W. (1974). *The representation of meaning in memory*. Hillsdale, NJ: Erlbaum.

- PATEL, V. L., & FREDERIKSEN, C. H. (1984). Cognitive processes in comprehension and knowledge acquisition by medical students and physicians. In H. G. Schmidt & M. C. de Volder (Eds.), *Tutorials in problem based learning*. Assen, Holland: Van Gorcum.
- PATEL, V. L., FREDERIKSEN, C. H., & GROEN, G. J. (1983, November). *Relationship between comprehension and medical problem solving*. Paper presented at the Research in Medical Education meeting, Washington, DC.
- TIERNEY, R. J., & MOSENTHAL, J. (1982). Discourse comprehension and production: Analyzing text structure and cohesion. In J. Langer & M. Smith-Burke (Eds.), *Reader meets author/bridging the gap: A psycholinguistic and sociolinguistic perspective*. International Reading Association.
- WINOGRAD, T. (1972). *Understanding natural language*. Edinburgh, Scotland: Edinburgh University Press.
- WIRTH, N. (1976). *Algorithms + data structures = programs*. New York: Prentice Hall.