

THE ROLE OF OBJECT PRIMITIVES IN NATURAL LANGUAGE PROCESSING *

Wendy G. Lehnert and Mark H. Burstein
Computer Science Department
Yale University
Box 2158 Yale Station
New Haven, CT 06520

A computer program, OPUS, is described which uses a set of Object Primitives to represent knowledge of physical objects and provide an organizing structure for associative memory. OPUS applies this representational system to the problem of analyzing natural language sentences dealing with objects.

1. INTRODUCTION

It is widely recognized that the process of understanding natural language texts cannot be accomplished without accessing mundane knowledge about the world [1,2,4,7]. In this paper, we are concerned with the way functional knowledge of objects, and associations between objects can be exploited in an understanding system.

Consider the sentence

- (1) John opened the bottle so he could pour the wine.

Anyone reading this sentence makes assumptions about what happened which go far beyond what is stated. We assume without hesitation that the wine was poured from inside the bottle. However, there are many other interpretations which are equally valid. For example, John could have been filling the bottle rather than emptying wine out of it. Yet, some causal inference mechanism, making use of functional knowledge of objects, causes us (as human understanders) to find the common interpretation in the process of connecting these two events causally.

In interpreting this sentence, we also rely on our knowledge of bottles and what it means for a bottle to be "open", when interpreting the sentence. Only by drawing on knowledge of what is possible when a bottle is open are able we understand why John had to open the bottle to pour the wine out of it.

When reading the sentence

*This work was supported in part by the Advanced Research Projects Agency of the Department of Defense and monitored under the Office of Naval Research under contract N00014-75-C-1111.

- (2) John turned on the faucet and filled his glass.

we immediately assume that John filled his glass with water from the faucet. Yet, water is never mentioned in the sentence. The glass could conceivably be filled with milk from a carton. However, in the absence of some greater context which forces a different interpretation on us, we immediately assume that the glass is being filled with water from the faucet.

This paper describes a computer program, OPUS (Object Primitive Understanding System) which constructs representations of the meanings of sentences such as those above, including assumptions that a human understander would normally make. To do this requires the use of stereotypic knowledge of physical objects, captured in OPUS using Object Primitives [3] which were designed to act in conjunction with the primitives of Schank's conceptual dependency representational system [7].

A complete description of the seven Object Primitives can be found in [3]. The two Object Primitives which we will exploit in our examples here are CONNECTOR, and SOURCE. A CONNECTOR is an object which enables some action between spatial regions, and a SOURCE is an object which "produces" another object. In addition, four associative links have been used, each pointing to a particular type of OP description. An example is the OUTPUTFROM link which connects objects such as wine with their normal SOURCE objects, in this case, wine bottles.

2. THE PROGRAM

2.1 The Initial Analysis

In the processing of the sentence (1) above, the phrase "John opened the bottle" results roughly in a representation of the fact that

John did something which caused the bottle to assume a state where the CONNECTOR description shown below applied.

```
*bottle* IS a CONNECTOR
      which ENABLES
?HUMO <=> PTRANS ?OBJ FROM (INSIDE SELF)
      (or)
?HUMO <=> PTRANS ?OBJ TO (INSIDE SELF)
      (or)
?HUMO <=> ATTEND ?SENSE TO ?OBJ
      (loc INSIDE SELF)
Where SELF refers to the object being described
(the bottle) and ? indicates an unfilled slot.
```

The CONNECTOR description indicates that something can be removed from the bottle, put into the bottle, or its contents can be smelled, looked at, or examined by some other sense modality. This CONNECTOR description is not part of the definition of the word 'open'. It is specific knowledge that people have about what it means to say that a bottle is open. To arrive at this representation, the program retrieves from memory the OP description of what it means for a bottle to be open, stored beneath its prototype for bottles. Presumably, there is also script-like information about the different methods for opening bottles, the different types of caps (corks, twist-off, ...), etc. However, for the purpose of understanding a text which does not refer to a specific type of bottle, cap, or opening procedure, what is important is the information about how the bottle can then be used once it is opened. This is the kind of knowledge that Object Primitives were designed to capture.

2.2 Concept Driven Inferences

When the phrase "so he could pour the wine." is analyzed by OPUS, the fact that the wine was poured from the bottle is not represented. This inference is made in the program by a slot-filling demon called the CONTAINER-FINDER, attached to the primitive act PTRANS. This demon looks on the list of active tokens (a part of short term memory) for objects that might normally contain the substance poured, in this case wine. The object found must be either a container (RELATIONAL = INSIDE) with the default object contained being wine, or be listed as a SOURCE of wine via an associative OUTPUTFROM link from the memory token for wine. If either test succeeds for some object in STM, then that object is inferred to be the container poured from. In the case of (1) above, the test succeeds since wine is OUTPUTFROM wine bottles.

2.3 Causal Verification

Once the slot filling inferences have been considered, the process which attempts to make causal connections between conceptualizations is activated. This process first looks for a match between the conceptual representation for the enabled action (pouring the wine), and one of the potentially enabled acts derived from the OP description of the open bottle. For sentence (1), a match is immediately found between the action of pouring from the bottle and the expected action generated from the CONNECTOR description of the open bottle (PTRANS FROM (INSIDE PART SELF)). The match causes a merging of the expected concept with the concept produced by the analyzer, linking the events described in the sentence and filling slots which were filled in one but not both of the original concepts.

2.4 Causal Chain Construction

In processing the sentence

- (3) John turned on the faucet so he could drink.

OPUS builds a token for a faucet as an active SOURCE of water. The principle expectation for SOURCE objects is that the person who "turned on" the SOURCE object wants to take control of (ATRANS) the object output from that SOURCE. This expectation has the important side effect, here, of creating a token for some water, which later is inferred as the substance John drank by a slot filling inference.

The phrase "he could drink" is represented in OPUS by an INGEST with water inferred as the liquid in the OBJECT slot. Since the expectation generated from turning on the faucet is for an ATRANS, the causal chain completion is clearly going to be more complicated here than in the previous example. When the chain connector fails to find a match between the ATRANS and either the INGEST or its instrumental PTRANS, additional inference procedures are called to generate any intermediate states that might connect these two acts. Using a combination of forward (resultative) and backward (causative) inference rules described by Rieger [5] in a restricted intersection search, OPUS infers that the intermediate state (John possess water) can be used to complete the causal chain.

3. CONCLUSIONS

It is important to understand how OPUS differs from previous inference strategies in natural language processing, such as that of Rieger [5]. A cursory comparison of OPUS and Rieger's MEMORY system reveals a number of similarities. The causative and resultative inferences used to complete the causal chain in our last example came directly from that work, and many of the demons used by OPUS are similar in flavor to the forward inferences and specification (slot-filling) inferences described by Rieger.

There are, however, two ways in which OPUS departs from the inference strategies of MEMORY in significant ways. [1] On one the level of computer implementation there is a reorganization of process control in OPUS, and [2] on a theoretical level OPUS exploits an additional representational system which allows inference generation to be more strongly directed and controlled.

In terms of implementation, OPUS integrates the processes of conceptual analysis and memory-based inference processing. By using demons, inferences can be made during conceptual analysis, as the conceptual memory representations are generated, thus eliminating much of the artificial modularity used in MEMORY more for pragmatic than theoretical reasons.

On a more theoretical level, the inference processes used for causal chain completion in OPUS are more highly constrained than was possible in Rieger's system. In MEMORY, all possible inferences were made for each new conceptualization which was input to the program. Causal chains were connected when matches were found between inferred concepts and concepts already stored in its memory. However, the inference mechanisms were not directed specifically to the task of making connections between concepts found in its input text. This led to a combinatorial explosion in the number of inferences made from each new input. In OPUS, forward expectations are based on specific associations from the objects mentioned, and only when the objects in the text are described in a manner that indicates they are being used functionally. During causal chain completion, at most two levels of additional forward or backward inferences are made before the procedure is exhausted, and the

system stops once a match is made. Thus, there is little chance for the kinds of combinatorial explosion Rieger experienced.

OPUS makes use of a well structured set of memory associations for objects, the Object Primitives, to encode information which can be used in a variety of Rieger's general inference classes. Because this information is directly associated with memory representations for the objects, rather than being embodied in disconnected inference rules elsewhere, appropriate inferences for the objects mentioned can be found directly. By using this extended representational system, we can begin to examine the kinds of associative memory required to produce what appeared from Rieger's model to be the "tremendous amount of 'hidden' computation" necessary for the processing of any natural language text.

REFERENCES

- [1] Bobrow, D. G., Kaplan, R. M., Kay, M., Norman, D. A., Thompson, H., and Winograd, T. (1977). GUS, a frame driven dialog system. Artificial Intelligence, Vol. 8, No. 1.
- [2] Charniak, E. (1972). Toward a model of children's story comprehension. ATR-266, Artificial Intelligence Laboratory, MIT, Cambridge, MA.
- [3] Lehnert, W. G. (1978). Representing physical objects in memory. Technical Report #131. Dept. of Computer Science, Yale University, New Haven, CT.
- [4] Minsky, M. (1975). A framework for representing knowledge. In Winston, P. H., ed., The Psychology of Computer Vision, McGraw-Hill, New York, NY.
- [5] Rieger, C. (1975). Conceptual memory. In R. C. Schank, ed., Conceptual Information Processing. North Holland, Amsterdam.
- [6] Riesbeck, C. K. and Schank, R. C. (1976) Comprehension by computer: Expectation-based analysis of sentences in context. Technical Report #78, Department of Computer Science, Yale University.
- [7] Schank, R. C. and Abelson, R. P. (1977) Scripts, Plans, Goals and Understanding. Lawrence Erlbaum Associates, Hillsdale, New Jersey.