COLING 82, J. Horecký (ed.)
North-Holland Publishing Company
© Academia, 1982

A MESSAGE-PASSING CONTROL STRUCTURE FOR TEXT UNDERSTANDING

Brian Phillips and James A. Hendler
Texas Instruments Inc.
Dallas, Texas, USA

This paper describes an object-oriented, message-passing system for natural language text understanding. The application domain is the texts of Texas Instruments' patent descriptions. The object-oriented environment permits syntactic analysis modules to communicate with domain knowledge modules to resolve ambiguities as they arise.

1.0 INTRODUCTION

As syntactic and conceptual coverage increase to meet the requirements of practical language understanding systems the computational effort to search the larger knowledge spaces tends to grow exponentially in current systems. Clearly, this search problem is one of (many) problems that have to be resolved.

One solution is to eliminate many of the alternatives as they are encountered. We are investigating a control structure that allows syntactic and semantic knowledge sources mutual access to allow early selection of appropriate alternatives.

We wish to include in our system the multiple facets of linguistic structure and to maintain their descriptive autonomy, accordingly other suggestions that have been made to constrain searching (e.g., Hendrix (1977), Schank (1975)) do not satisfy this design criterion. We also want the system to simultaneously build a conceptual representation of the text and to be able to feed semantic predictions to syntax. In the Rus system (Bobrow (1978)) the semantic component critiques the constructs of syntax but does not generate predictions.

2.0 OBJECTS AND MESSAGE-PASSING

We have adopted a pseudo-parallel, object-oriented approach to writing our system (Hewitt (1976)). Objects encapsulate data and their operations. Actions on data can only be performed by sending messages to appropriate objects. A request for action may require that the object enlist the aid of other objects, which it does by further message-passing. Any object can communicate with any other object (though objects can receive messages they cannot process). Further, an object need not get a reply to a message. Objects have memory and can retain their state between activations. This flow of control among objects is more general than stack-oriented activation of subprograms.

Interlisp, Simula, Smalltalk, and Lisp Machine Lisp (Weinreb & Moon (1981)) have features that encompass the object notior. Our system is being implemented using the "flavor" system in Lisp Machine Lisp.

3.0 THE APPLICATION

We are using working abstracts of descriptions of Texas Instruments' issued patents. These are written in a restricted style by by the attorneys and the topics are limited to solid-state micro-electronic devices. Thus we are able to use naturally occurring data without immediately confronting the problems of incomplete syntactic and conceptual coverage that would be encountered in many other domains. An example is:

> A modulator comprising two transistors each having collector, emitter and base electrodes, means for applying a direct voltage across said emitter electrodes, a center-tapped source of alternating signals connected between said base electrodes, said collector electrodes being connected together and to the center tap of said source. A load impedance connected between said collector electrodes and said emitter electrode of one of said transistors, and a variable resistor connected between the base electrode and the emitter electrode of said one transistor.

The interaction of embedding and conjunction gives a high degree of syntactic ambiguity to the texts. The texts can also be ungrammatical, whence the desire to be building the conceptual representation in parallel with the syntactic analysis in order that some meaning will be extracted from the text even when the syntactic analysis is not completed.

The goal of the project is to build a conceptual representation for the text, then add retrieval capabilities that will be more flexible than a word-matching scheme.

4.0 THE SYSTEM

The objects of our system correspond to the organizing principles of the components. In syntax we have constituency objects that can take grammar rules and try to match them against input. In semantics we have taxonomy objects, case-frame objects, causal-chain objects, meta objects (that handle a form of lambda-abstraction), etc.

Small & Rieger (1981) also have an object viewpoint of language analysis. However, in their scheme each word is an object, motivated by their view that "human knowledge about language is organized primarily as knowledge about words rather than as knowledge about rules" (p.1). Our system is organized around rules.

Objects in different components do not necessarily have the same vocabulary: in syntax there are words and phrase structure and semantics has concepts and relations; accordingly there is a translation object through which messages pass.

Kornfeld (1979) has given an example of a (pseudo-)parallel communication system that passes information between objects to reduce the respective search spaces; he terms the phenomenon "combinatorial IMplosion". The interaction between syntax and semantics allows the the conceptual representation of sentence fragments to be built in parallel with the syntactic analysis. Semantic predictions are fed back to syntax to try to achieve the combinatorial implosion.

4.1 The Syntax

The formalism we are using is "local grammar" (Saenz (1980), Ross (1981)) which consists of a context-free phrase structure grammar with augmentations. The augmentations are blocking and percolation rules. For example, the auxiliary rule in our system is

```
verb-group => aux verb-group        structure rule
  aspect (1) = affix (2)            blocking rule
  affix (0) = affix (1)             percolation rule
```

Figure 1: The auxiliary rule

The structure segments are numbered left to right, starting at 0 for the left-hand-side of the rule. The values of the features "aspect" and "affix" are established in the dictionary for terminal items and percolated up the analysis tree for higher level phrases.

The parsing algorithm is a modified left-corner routine (Griffiths & Petrick (1965)). The modifications are to use the object environment to produce all parses in parallel and to merge common subparses.

```
#CONSTITUENT 22731061#, an object of flavor CONSTITUENT,
has instance variable values:
      CATEGORY:            VERB-GROUP
      GOALS-LIST:          ((VERB-GROUP . #CONSTITUENT 22731051#))
      PART-PARSE:          ((1. BEING ((AUX (AFFIX PROGRESSIVE)
                                             (ASPECT PASSIVE)))))
      RULE-TAIL:           ((VERB-GROUP))
      AUGMENTATIONS:       ((EQUAL (ASPECT 1.) (AFFIX 2.))
                            (PERCOLATE AFFIX (AFFIX 1.)))
      SEGMENT-COUNTER:     2.
      INPUT-WORD:          BEING
```

Figure 2: A syntactic constituent object

Figure 2 gives an example of the state variables of a constituent object that is using the auxiliary rule of Figure 1. "Category" is the left-hand-side of the structure rule, "part-parse" is the fragment of the right-hand-side so far matched, and "rule-tail" is the remaining part of the structure rule. "Augmentations" are the percolation and blocking rules. The "goals-list" gives other constituent objects that are awaiting completion of this constituent (there may be several because of merged paths). The word "being" has just been processed as the first segment of the part-parse and the segment counter now points to the second position. The object has processes for (a) advancing the analysis with a new input word, (b) for advancing the analysis when a subparse has been completed (the parse can only be immediately advanced when the next element of the rule-tail is a terminal symbol; otherwise it has to create another object to process a rule expanding the non-terminal category), and (c) for merging with another path.

4.2 The Knowledge Base

General knowledge of the domain is represented in a semantic network (Phillips (1978)). The conceptual analyses will be instantiations of general knowledge with novelty introduced from the texts.

Semantic nets are usually seen as data (e.g., Qillian (1968), Brachman (1978)) with various routines for performing operations such as taxonomy searches, finding paths from node to node, and binding variables in the network. Each node of our network is an object having associated processes dependent on the types of links it has to other nodes.

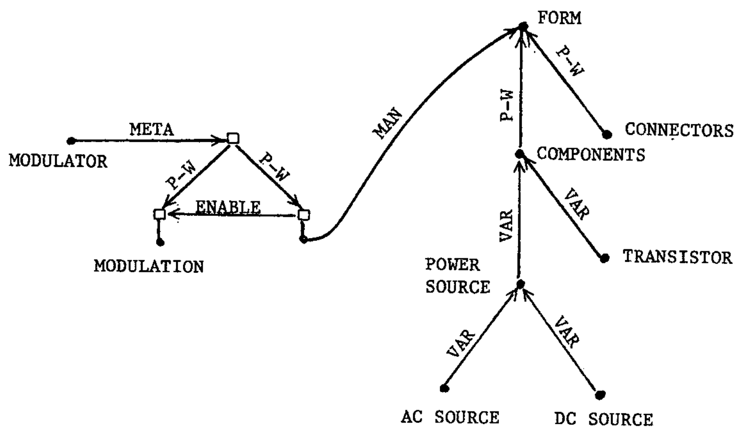

Figure 3: A fragment of a semantic net

Thus in Figure 3, which shows part of our semantic network, links indicate the other nodes to which a node can send a message, as opposed to a physical pointer. A node is actually implemented as a "mix" of objects for the kinds of links it has; thus as new links are added so are new processes. A node need not know anything about the internal format of data in other nodes to get information from them. Further, when a message is sent to a node, the sender need not know whether this is a "simple" or "complex" message: a simple message can be answered by the node itself, a complex message requires the node to send messages to other nodes. Thus a "part-whole" message to establish whether a direct-voltage source can be part of a modulator or part of a transistor, will also use intervening taxonomy, decomposition (meta) links (Figure 3) without this being specified in the original query. A node receiving this message would pass a similar message to its neighbouring nodes if it cannot itself respond.

4.3 Flow Of Control

Processing of text is initiated from a task specific knowledge object, in this case a "patent knowledge expert" that has an expectation of finding a patent. It passes a message to the translator that sees if it has any data that will match this expectation. Since no part of the text has been examined by syntax, nothing can be found. A start message is sent to syntax.

The translator object should pass predictions to syntax but this does not, in general, seem possible as the realizations of a concept include all possible descriptive references. Thus the translator maintains its list of predictions and, when syntactic constituents are received, matches their translations against the predictions. A match causes a message to be sent to the source of the prediction, which can extend the conceptual representation and produce further predictions.

When no matches are found, the translator seeks a knowledge structure that corresponds to the syntactic structure. This occurs, for example, when the syntactic objects are confronted with the attachment of the "means for applying ..." phrase in the example given above: is the correct analysis "modulator comprising ... means ..." or "transistors each having ... means ..."? A path through the network of Figure 3 shows that a modulator can have a direct voltage source but no such path exists for transistors. The appropriate instantiation of the network is created and the unacceptable syntactic path is eliminated from further consideration.

Processing is complete when the text has been consumed and all the concepts from the text are connected to the topic of patent, though ungrammaticality may cause an earlier end.

5.0 CONCLUSION

Our understanding of language and cognitive processes is growing and novel programming languages are developing. With this knowledge and these tools, we are getting closer to viable natural language systems in limited domains.

There are other developments that will contribute to building natural language systems, namely the decreasing cost and increasing power of hardware. Also advances in computer-aided design give promise of cost-effective special purpose machines with hardware routines for processes now implemented in software (Fahlman (1979)). More power will certainly aid in constructing language understanding systems. But the power will be wasted if it is used to attack a problem that can be resolved by some other approach, say by constructing an object-oriented system as presented above.

6.0 REFERENCES

[1] Bobrow, R.J., The RUS System, Report 3878, Bolt Beranek & Newman Inc., Cambridge, MA. (July 1978).

[2] Brachman, R.J, A structural paradigm for representing knowledge, Report 3605, Bolt, Beranek, & Newman Inc., Cambridge, MA. (May 1978).

[3] Fahlman, S.E., NETL: A System for Representing and Using Real World Knowledge (MIT Press, Cambridge, MA, 1979).

[4] Griffiths, T.V., & Petrick, S.R, On the relative efficiencies of context-free grammar recognizers, Comm. ACM 5 (1965) 289-300.

[5] Hendrix, G.G., Human engineering for applied natural language processing, in Proceedings of the 5th International Joint Conference on Artificial Intelligence (Cambridge, 1977).

[6] Hewitt, C., Viewing control structures as patterns of passing messages, AI Memo 410, MIT AI Laboratory, Cambridge, MA. (December 1976).

[7] Kornfeld, W.A., Using parallel processing for problem solving, AI Memo 561, MIT AI Laboratory, Cambridge, MA. (December 1979).

[8] Phillips, B., A model for knowledge and its application discourse analysis. American Journal of Computational Linguistics, Microfiche 82 (1978).

[9] Quillian, M.R., Semantic memory, in Minsky, M. (ed.), Semantic Information Processing (MIT Press, Cambridge, MA, 1968).

[10] Ross, K.M., Parsing English Phrase Structures, Ph.D. Thesis, Dept. of Linguistics, University of Massachusetts (September 1981).

[11] Saenz, R.M., Local Grammar, Unpublished paper, Dept. of Linguistics, University of Massachusetts (February 1980).

[12] Schank, R.C., Conceptual Information Processing (American Elsevier, New York, 1975).

[13] Small, S.L., & Rieger, C., Parsing and Comprehending with Word Experts, Technical Report 1039, Artificial Intelligence Group, University of Maryland (April 1981).

[14] Weinreb, D., & Moon, D., Lisp Machine Manual (MIT AI Laboratory, Cambridge, MA, 1981).