An Equipment Model and its Role in the Interpretation of Noun Phrases

Tomasz Ksiezyk and Ralph Grishman and John Sterling
Department of Computer Science
Courant Institute of Mathematical Sciences
New York University
New York. New York 10012

Abstract

For natural language understanding systems designed for domains including relatively complex equipment, it is not sufficient to use general knowledge about this equipment. We show problems which can be solved only if the system has access to a detailed equipment model. We discuss the structure of such models in some detail and, in particular, the mixed static/dynamic nature of the model. As an illustration, we describe parts of a simulation model for an air compressor. Finally, we demonstrate how to find referents in this model for noun phrases.

I. Introduction

The work presented here is part of PROTEUS* (PROtotype TExt Understanding System), currently under development at the Courant Institute of Mathematical Sciences, New York University.** The objective *of* our research is to understand short natural language texts about equipment. Our texts at present are CASualty REPorts (CASREPs) which describe failures of equipment installed on Navy ships. Our initial domain is the starting air system for propulsion gas turbines. A typical CASREP consists of several sentences, for example:

> Unable to maintain lube oil pressure to SAC *[Starting Air Compressor],* Disengaged immediately after alarm. Metal particles in oil sample and strainer.

It is widely accepted among researchers that in order to create natural language understanding systems robust enough for practical application, it is necessary to provide them with a lot of common-sense and domain-specific knowledge. However, so far, there is no consensus as to what is the best way of choosing, organizing and using such knowledge.

The novelty of the approach presented here is that, besides general knowledge about equipment, we also use a quite extensive simulation model for the specific piece of equipment which the texts deal with. We see the following merits of having a simulation model:

- The model provides us with a reliable background against which we can check the correctness of the understanding process on several levels: finding referents of noun phrases, assigning; semantic cases to verbs, establishing causal relationships between individual sentences of the text
- The requirements of simulation help us to decide what kind of knowledge about the equipment should be included in the model, how it could best be organized and which inferences it should be possible to make. It appears that the information needed for simulation largely coincides with that necessary for language understanding.
- The ability to simulate the behavior of a piece of equipment provides a very nice verification method of the understanding process at the level of interaction with a user - a dynamic graphical interface provides the user with insight into the way his input has been understood by the system.

In the remainder of this paper we shall address three issues: Why is a detailed equipment model needed? How should such a model be structured and, in particular, what balance should we strike between a static model and one created dynamically as the text requires? How should a noun phrase analyzer be organized to utilize such a model, and how is it affected by this static/dynamic balance?

II. Need for a Model

In many natural language processing systems, the domain knowledge consists of *general* information about the objects and operations in the domain. In the equipment domain, this would include knowledge of the possible states and actions of equipment components, such as valves, pumps, and gears. It is clear, however, that such knowledge is not sufficient for a complete understanding of the CASREP messages we are studying.

One feature of technical texts is the heavy use of nominal compounds. It seems that their average length is proportional to the complexity of the discourse domain. In the domain of the starting air system, examples like

•This research was supported in part by the Defense Advanced Research Projects Agency under contract N00014-85-K-0163 from the Office of Naval Research and the National Science Foundation under grant DCR-85-01843.

••This work is being done in collaboration with Unisys Defense Systems (formerly the System Development Corp.) as part of the DARPA Strategic Computing Program.

stripped lube oil pump drive gear

are by no means seldom occurrences.

The problem with nominal compounds is their ambiguity. The syntactic analysis is of almost no help here. Even using semantic (selectional) constraints, as in [Finin, 1986], substantial ambiguity often remains. When we know that the nominal compounds refer to objects existing in the system, and have access to a model of the system. we can impose much tighter constraints, thus reducing the ambiguity.

The need for an equipment model is even more evident when we consider the analysis of a multi-sentence text such as

Starting air regulating valve failed.
Unable to consistently start nr lb turbine.

(this is an excerpt from an actual CASREP). In the starting air system (our initial domain) there are three different valves regulating starting air. Two questions might be posed in connection with this short text: (1) which of the three valves was meant in the first sentence? (2) could the failure of the valve mentioned in the first sentence be the cause of the trouble reported in the second sentence?

The general knowledge of equipment may tell us a lot about failures, such as: if a machinery element fails, then it is inoperative, or if an element is inoperative, then the element of which it is part is probably inoperative as well, etc. Unfortunately, such knowledge is not enough: there is no way to answer these two questions (not only for an artificial understanding system, but even for us, humans) without access to rather detailed knowledge about how various elements of the given piece of equipment are interconnected and how they work as an ensemble. In our case we could hypothesize (using general knowledge about text structures) that there is a causal relationship between the facts stated in the two sentences. To test this, we would have to consider each *of* the three valves in turn and check how its inoperative state could affect the starting of the specific (i.e. nr lb) turbine. To perform these tests we would need a simulation model. If one of the three valves, when inoperative, would make the turbine starting unreliable, then we could claim that this valve is the proper referent for the *starting air regulating valve* mentioned in the first sentence. This finding would let us also answer question (2) affirmatively.

The above two considerations demonstrate that in cases where the domain is very specialized and complicated (a typical situation for real-life equipment), language understanding systems should be provided not only with general knowledge about the equipment but also have access to its model.

III. PROTEUS structure

PROTEUS consists of a syntactic analyzer, a semantic analyzer and a discourse analyzer. The semantic analyzer translates the regularized syntactic analysis into a predicate-argument structure. As part of the semantic analysis, the noun phrase analyzer (described below) identifies elements of the equipment model corresponding to the referents of noun phrases. The discourse analyzer [Joskowicz *et al.*. 1987] , using the equipment model, identifies implicit causal and temporal relations in the message. [Grishman *et* a/., 1986] describes the overall organization of PROTEUS. The system is implemented on Symbolics LISP machines.

IV. Simulation Model

A. Structure of the simulation models

The target domains for PROTEUS are *equipment units* (EU): complex technical systems which accomplish physical tasks on demand. These tasks are carried out as serial and parallel combinations of simpler tasks, which are performed by constituent EUs of the main equipment unit. Often these simpler tasks can be decomposed further, leading to a hierarchy of tasks and EUs.

The EUs transmit their effects through various *media,* such as gases, liquids, mechanical movement, and electric current. These media travel from one EU to another through *conduits* appropriate to the different types of media.

PROTEUS models have the structure of a set of transition networks. They consist of *nodes* connected by directed *links.* The nodes correspond to the constituent EUs of the system; the links to the conduits connecting the EUs. The hierarchical structure of the EUs is reflected in the hierarchical structure of the networks. To represent the internal structure of an EU, we have the corresponding node point to another network in the model.

Associated with each link is a working-substance (WS). These WSs correspond to the media entering and leaving an EU (for example, the rotary motion provided to a pump and the fluid entering and leaving the pump). We can think of the WSs associated with links entering and leaving a node as the input and output data of the node.

Associated with the nodes, links, and working substances are properties, recording the structure, function and (time-dependent) state of the system elements.

One design criterion for our models is that they make possible the qualitative simulation of the modeled EUs (in most cases a precise quantitative simulation is not required for language understanding). The model includes time-dependent values for the states of modeled components and functions which determine the state and outputs of the nodes. Simulation is performed by an event-driven algorithm which is triggered by an external event (such as operator action or reported failure) and continues until a stable state is reached.

We have implemented the model by defining prototypes for the various types of components (valves, gearboxes, etc.) and then assembling a system as a collection of instances of these prototypes (implemented in *Flavors).* Information about each type of component is stored in the

prototype, so that only information specific to a particular component need be stored in the instance. For example, in the case of a gearbox, the information about its function (speed change) should be stored in the prototype, and only the ratio of this change should reside in the instance of a specific gearbox. The 'library'of prototypes should greatly simplify the creation of new equipment models within the system.

B. Level of detail

How detailed a model should we construct? A first re sponse might be to include everything which potentially may be referred to in the reports. This, however, seems impractical. Consider a typical sentence from one of the reports:

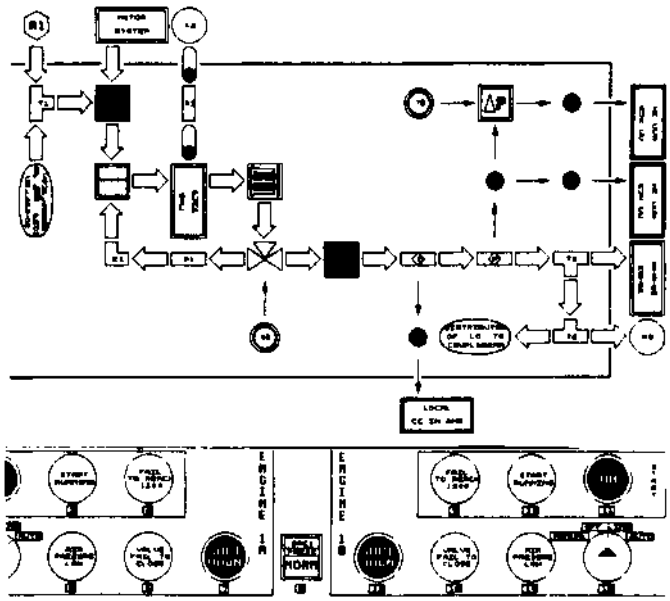
> Investigation revealed a broken tooth on the hub ring gear.

Considering that there are several different gears in our starting air system and each of them has many teeth which are very much alike, it's obvious that creating a separate description for each of them wouldn't be reasonable. The same remark is true for balls bearings or for connecting elements like screws, bolts or pins. On the other hand, information about the tooth conveyed in the above sentence cannot go unnoticed. The solution we accepted for such elements is not to include their descriptions in the model on a permanent basis but to keep open the possibility to create and add them to the model if such a need arises during the analysis. A rule of thumb for deciding, whether a particular element deserves a permanent place in the model can be formulated as a question: How much information specific to this element is necessary to solve understanding problems, like finding referents or making inferences?

If there is substantial specific information, the element is included as part of the permanent model. On the other hand, if all the needed information can be derived from the larger unit of which this element is a part (the *gear* of *tooth on gear),* and this larger unit is always mentioned in descriptions of the part, then the element can be added to the model dynamically when it is mentioned in a message.

C. An example: the starting air system

As our initial domain we have chosen the "starting air system" used for starting gas turbines on Navy ships. The model consists of 15 networks with a total of about 175 nodes. One of these networks is shown in Fig. 1. This figure is a *Symbolics* screen image generated by PROTEUS from the model networks. Some parts of the display are dynamic: gears rotate, oil moves as visible particles, etc. This provides a direct visual presentation of the system's understanding of a message (oil may stop flowing or a gear rotating). The dynamic displays are achieved as a side effect of the simulation used for understanding purposes.



Figuie 1. The Lube Oil System

V. Noun Phrase Analysis

A. The role of noun phrase analysis

The goal of the' Noun Phrase Analyzer (NPA) is to convert a noun phiase into a set of referents that may be used by subsequent stages of the system. To accomplish this it uses the equipment model two ways. Fust, the model is used to confirm possible relations between noun phrase constituents (for example, that there is a gear which is an adjacent part to a pump) Second, the model provides a set of referents for the phrase

The NPA converts from the linguistic representation to one in terms of domain predicates. The interface between the NPA and the model is called the Model Query Processor (MQP). The MQP evaluates the domain predicates relative to the equipment model, and creates internal representations for EUs when needed. Major equipment units are part of the static model; others must be created dynamically.

B. The Analysis Procedure

The NPA fetches a semantic class and various other features for each word in the noun phrase. Constituents are combined bottom-up based on a set of rules stated in terms of these classes. These rules identify relationships of the form (Pred Arg-1 Arg-2 ...) where *Pred* is a predicate of the domain and the *Arg-i* are constituents of the noun phrase. We have analyzed a corpus of 38 sentences and identified a set of 13 predicates for analyzing noun phrases:

> adjacent-to alarm couple drive lube made-of measure name operate-on part-of regulate start location

Most current systems validate the application of these rules through such selectional checks (constraints on the argument classes of each predicate) [Finin, 1986] We perform such checks, then go a step further and check for the existence of the specific relation between specific entities. This is done through the MQP. The MQP is first invoked for each word to obtain its internal representation in the model, and then for each proposed predicate to verify the existence of the corresponding relation in the model. If the relation exists, the MQP returns the representation for the head constituent of the relation.

For post-nominal modifiers the predicate is strongly indicated by the preposition, and arguments (the head noun and the object of the preposition are explicit and delimited). Pre-nominal modifiers are more difficult. The problem is to decide what predicates should be used and what are the arguments of these predicates. Both the predicates as well as their arguments may be given explicitly or implicitly. Examples are: *temperature regulating valve* (the predicate and both its arguments are explicit), *drive gear* (the predicate and one of its arguments are explicit, the other argument, the object of DRIVE, is implicit), *pump shaft* (the predicate, PART-OF, is implicit, both of its arguments are explicit). The NPA considers the semantic features of the items, together with order constraints, to match the items with arguments of some canonical predicate. A match is considered successful, if it is possible to identify some (not necessarily all) of the arguments of the ptedicate among the modifiers. For verification purposes, it is assumed that the empty arguments match anything. Once a matching canonical predicate has been found and as many of their arguments matched with the modifiers as possible, the NPA poses a verification query to the model.

In many cases the noun phrase does not determine a unique entity, so a set is returned. Context and default information are used to resolve such references. This is handled by the Reference Resolution Module, following [Pahnei *et ai,* 1986].

C. Model Query Processor

We have observed the mixed static/dynamic characteristic of our representation. Since the other modules are designed to be independent of the model representation, this distinction must be hidden by the MQP. We discuss here only queries posed by the NPA, i.e. requests for the representation of a word, and predicate verification queries.

The main EUs are recorded permanently in the equipment model. For words corresponding to these EUs, the MQP contains pointers to all the nodes in the model to which the word may refer (for example, the entry for *pump* will point to all pumps in the model). However, two classes of EUs are created dynamically by the MQP. The first class consists of components too small to justify including in the model (eg. *connecting pin in pump drive assembly* or *tooth of hub gear.)* The second class involves aggregates of elements which are described in the text and treated as a unit but do not correspond to a single unit in the model hierarchy. There are several examples in our corpus, such as *the coupling from diesel to SAC lube oil pump.*

The predicate verification queries must also distinguish between static and dynamically created EUS. Where all the arguments are statically modeled, the MQP need only check the model attribute corresponding to the predicate. When dynamically created, the MQP will generally have to modify theargument to reflect the constraint expressed by the predicate. For example, in verifying *{part-of clutch SAC),* where both *clutch* and *SAC* are statically modeled, we *check* the PART-OF role of *clutch:* Whereas, in verifying *[part-of tooth gear),* with *tooth* dynamically created, we *fill* the PART-OF role of *tooth.*

VI. Future Work

In addition to noun phrase analysis, the equipment model is used heavily in discourse analysis — identifying implicit causal and temporal relations. We have developed a preliminary implementation of discourse analysis [Joskowicz *tt al.,* 1987] but much work remains to be done, particularly regarding time dependencies. We also intend to develop tools for the more efficient acquisition of equipment models, and to study techniques for dealing with gaps in the domain model.

The initial motivation for PROTEUS was text understanding for subsequent querying, summarization, and trend analysis. Our use of a detailed equipment model similar to that employed in simulation systems (eg. STEAMER[Hollan *et al,* 1984]) and diagnostic systems suggests that PROTEUS is also useful as an interface to such systems.

References

[Finin, 1986] T. W. Finin. Constraining the interpretation of nominal compounds in a limited context. In R. Grishman and R. Kittredge, editors, *Analyzing Language in a Restricted Domain,* Lawrence Erlbaum Assoc, Hillsdale, NJ, 1986.

[Grishman *et* a/., 1986] Ralph Grishman, Tomasz Ksiezyk, and Ngo Thanh Nhan. *Model-based Analysis of Messages about Equipment.* PROTEUS Project Memorandum 2, New York University, 1986.

[Hollan *et* a/., 1984] J. Hollan, E. Hutchins. and L. Weitzman. Steamer: an interactive inspectable simulation-based training system. *AI Magazine,* 5(2):13-27, Summer 1984.

[Joskowicz *et* a/., 1987] Leo Joskowicz, Ralph Grishman. and Tomasz Ksiezyk. *Causal and Temporal Relations in Equipment Messages,* PROTEUS Project Memorandum 8, New York University, February 1987.

[Palmer *et* a/., 1986] M. Palmer, D. Dahl. R. Schiffman. L. Hirschman, M. Linebarger, and J. Dowding. Recovering implicit information. In *24th Annual Meeting of the ACL,* pages 10-19, 1986.