Causal and Temporal Text Analysis: The Role of the Domain Model

Ralph Grishman

Computer Science Department
New York University
New York, NY 10003, USA
grishman@nyu.edu

Tomasz Ksiezyk*

Human Interface Lab
Microelectronics and Computer Technology Corp.
Austin, TX 78759, USA
ksiezyk@mcc.com

Overview

It is generally recognized that interpreting natural language input may require access to detailed knowledge of the domain involved. This is particularly true for multi-sentence discourse, where we must not only analyze the individual sentences but also establish the connections between them. Simple semantic constraints — an object classification hierarchy, a catalog of meaningful semantic relations — are not sufficient. However, the appropriate structure for integrating a language analyzer with a complex dynamic (time-dependent) model — one which can scale up beyond 'toy' domains — is not yet well understood.

To explore these design issues, we have developed a system which uses a rich model of a real, nontrivial piece of equipment in order to analyze, in depth, reports of the failure of this equipment. This system has been fully implemented and demonstrated on actual failure reports. In outlining this system over the next few pages, we focus particularly on the language analysis components which require detailed domain knowledge, and how these requirements have affected the design of the domain model.

The Domain

The texts we are analyzing are CASREPs: reports of equipment failure on board U.S.Navy ships. We have restricted ourselves to one subsystem, the starting air system, which generates compressed air for starting gas turbines. With nearly 200 functional components, it is complex enough to raise many of the problems of real systems, yet still remain within the range of exploratory model-building efforts. We have collected 36 reports concerning this subsystem. A typical report is

> While diesel was operating with SAC [starting air compressor] disengaged, the SAC LO [lubricating oil] alarm sounded. Believe the coupling from diesel to SAC lube oil pump to be sheared. Pump will not turn when engine jacks over.

A central task of text analysis is to determine (as best one can from the report) the cause-effect relation between events. This information is rarely stated explicitly; rather, it is assumed that it can be inferred from a reader's background knowledge. We can illustrate this with a simple example from a more familiar domain — car repair. If we compare the reports

> Battery low.
> Engine won't start.

and

> Battery low.
> Generator won't start.

we recognize that, although the texts are very similar, in the first case "Battery low" *causes* "Engine won't start.", whereas in the second, "Battery low" *is the result of* "Generator won't start." We make these inferences quite naturally based on our knowledge of how cars work. The challenge is to organize our system so that it can effectively make similar inferences using complex domain models.

Analyzing these causal relations helps us in turn to understand the temporal structure of the text. This is important because the narrative order in these reports typically reflects the order in which events were discovered rather than the order in

* Work performed while at the Computer Science Dept., New York University

which they occurred.

The Language Analyzer

The language analyzer has three top-level components: syntactic analysis, semantic analysis, and discourse analysis. Syntactic and semantic analysis are applied to each sentence in turn; discourse analysis is applied to the entire report at the end of processing.

Syntax analysis is performed using an augmented context-free grammar based on linguistic string theory. The parse tree is regularized (primarily transforming all clause structures into a standard form) by a set of translation rules associated with the grammar productions and applied compositionally.

Semantic analysis is split into predicate semantics (which handles clauses and nominalizations) and noun phrase semantics (for references to domain objects). Predicate semantics performs a mapping from verbs and syntactic relations to domain-specific predicates and relations. Noun phrase semantics maps noun phrases into references to components of the domain model.

Noun phrase semantics has to cope with the long compound nominals which occur frequently in this and other technical text. Our reports contain phrases such as

starting air temperature regulating valve
SAC [starting air compressor] spline input drive shaft

Syntactic constraints offer almost no help in resolving the ambiguity of such phrases, and semantic constraints, as described by Finin [2], are in many cases not sufficient. We instead adopt a two-stage approach to analyzing these phrases, described in more detail in [3], [4], and [5]. The noun phrase is first parsed with a grammar based on broad semantic categories appropriate to the domain; this may produce several alternate analysis trees. These analyses are then submitted to a compositional procedure which determines for each subtree, and finally the whole tree, the referents in the model. By eliminating analyses which yield null referents, we resolve much of the ambiguity in these noun phrases.

When semantic analysis is complete, it will have transformed the report into a set of propositions (predicate-argument structures). Discourse analysis now has the task of interrelating these propositions.

Discourse Analysis

The central data structure of the discourse analyzer is the *time graph*. The time graph contains a set of directed edges which correspond to time intervals over which a certain state holds or a certain activity is taking place (we call such states and activities *elementary facts*). In addition, the time graph has directed edges which represent the relative time ordering of the elementary facts and the causal relationships between them. This graph is created in three phrases: creation of elementary facts; analysis of explicit temporal relations; and causal analysis. Our approach to temporal analysis, which is described in more fully in [3] and [4], has been influenced by earlier work by Dowty [1] and Passonneau [6].

The first phase creates the elementary facts from the propositions generated by semantic analysis. For propositions representing a continuing state or activity, the mapping is, in general, one-to-one. For propositions representing a change of state, however, we generate several facts: in general, one for the prior state, one for the transition interval, and one for the final state. Higher-order predicates (those which take one or more propositional arguments, such as "began to ___", "unable to ___") do not map directly into elementary facts; rather, they modify or augment the constellation of elementary facts created for their arguments.

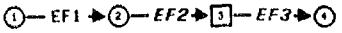
For example, for the (simplified) report

Starting air temperature regulating valve failed. Was unable to start nr. 1A turbine.

we would create the elementary facts shown in Fig. 1. **valve-14** is the internal name for the 'starting air temperature regulating valve', while **turbine-1** is the internal name of the 'nr. 1A turbine' (these references are identified by the noun phrase analyzer). The failure predication in the first sentence is translated into three elementary facts: the state when the valve was OK (between time points 1 and 2), the process of failing (between 2 and 3), and the failed state (between 3 and 4). A predication of 'starting' by itself would be similarly translated into three elementary facts. The adjective 'unable to' introduces an additional elementary fact EF5 — the operator performing the starting action — and modifies the facts representing the change of state of the turbine (EF4, EF6, EF7) so that the turbine is *not* running in the final state (EF7).

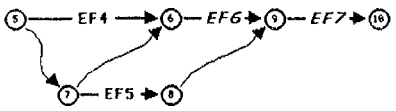
The second phase introduces edges corresponding to temporal relations explicitly mentioned in the text. For example, for the text "While diesel

(a)



ef1: (Static-State *equip-unit:* valve-14 *aspect:*
FUNCTIONAL CONDITION *value:* OK)
ef2: (Transition-State *equip-unit:* valve-14 *aspect:*
FUNCTIONAL CONDITION *value:* BECOMING MALFUNCTIONING
transition-type: UNDESIRED)
ef3: (Static-State *equip-unit:* valve-14 *aspect:*
FUNCTIONAL CONDITION *value:* BROKEN)

(b)



ef4: (Dynamic-State *equip-unit:* turbine-1 *aspect:* OPERATION
value: STANDSTILL *mode:* ROUTINE)
ef5: (Action *actor:* OPERATOR *aspect:* ROUTINE *value:* STARTING
equip-unit: turbine-1 *type:* INITIATION)
ef6: (Transition-State *equip-unit:* turbine-1 *aspect:* OPERATION
value: STARTING *transition-type:* INITIATED)
ef7: (Dynamic-State *equip-unit:* turbine-1 *aspect:* OPERATION
value: NOT RUNNING *mode:* ROUTINE)

Figure 1. Discourse analysis: creation of elementary facts.
(a) *Starting air temperature regulating valve failed.*
(b) *Was unable to start nr. 1A turbine.*

(a) (b)

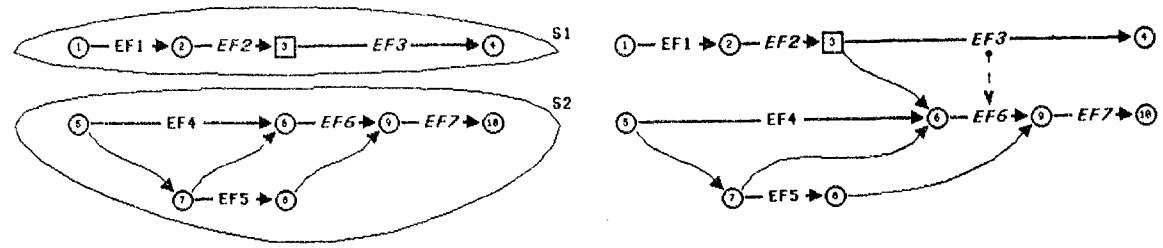


Figure 2. Discourse analysis: causal analysis of a failure report.
(a) Situations S1 and S2.
(b) Adding causal and temporal links between situations.

was operating, alarm sounded." we would indicate that the transition interval when the alarm began to sound is contained in the interval in which the diesel was operating. For the simple example just above, no edges would be added.

The third phase uses causal inference to determine the causal relation between elementary facts, and to obtain therefrom additional temporal relations. When this phase begins, the time graph consists of several connected subgraphs, which we call *situations*. In essence, we consider each pair of situations, <situation$_1$, situation$_2$>, and use the model to determine whether situation$_1$ is a plausible cause of situation$_2$. We take the domain model *without* the conditions of situation$_1$ and test whether situation$_2$ is true or false; we then alter the state of the model to reflect the conditions of situation$_1$ and again test whether situation$_2$ is true or false. If it is false in the first case and true in the second, we record a plausible causal link from situation$_1$ to situation$_2$. In fact, we need not test all pairs of situations; we can restrict ourselves to *abnormal* situations (those which are not true in the case of normal operation of the equipment).

The example above consists of two situations, S1 and S2 (Fig. 2(a)), both of which are abnormal, so we perform the tests just described. We determine that S1 is a plausible cause of S2. We therefore establish a causal link (shown as the dotted line from EF3 to EF6), and deduce therefrom a temporal link from the start of EF3 to the start of EF6. These are shown in Figure 2(b).

The Domain Model

The detailed equipment model is required primarily at two points in our analysis: for noun phrase semantics and for causal reasoning as a part of discourse analysis. Each imposes particular requirements on the model.[1]

Noun phrase analysis requires a static hierarchical model of the equipment which captures the properties and relations which are used in noun phrases to identify particular components: containment, adjacency, function, parameter values ("high speed").

[1] In addition, the entire system, and particularly semantic analysis, make use of more conventional domain information structures: a hierarchical classification of objects and predicates, and a map from verbs and nominalizations to predicates.

There are two conventional approaches to cause-effect reasoning: a "shallow" approach in which causally related events are recorded directly, typically in a production system, and a "deeper" model-based approach in which effects are propagated through components as they would be in the actual equipment. We have elected to use a model-based simulation, in part because a static model (which provides the framework for the simulation model) was required for semantic analysis, and in part because it offers a more systematic approach to assuring adequate coverage of the cause-effect relations. We have found that a *qualitative* simulation, in which parameters take on only a few values, was adequate for verifying the causal relations mentioned in the reports; correct understanding rarely depended on knowing the correct numerical values of parameters.

Certain cause-effect relations, such as those involving a single system component (e.g., that corrosion of an element might lead to its malfunctioning), cannot be directly captured by the simulation model; we use production rules to express the relation in such cases.

In order to isolate the language analyzer from the particular choices of representation made in the domain model, we have introduced a Model Query Processor as an interface between the analyzer and model. The resulting system structure is shown in Figure 3. The Model Query Processor accepts queries about the static model, either testing a parameter of a component or a relation (adjacency, containment, etc.) between two components. It also accepts queries about the interaction of events, stated in terms of asserting or testing particular elementary facts; these are translated into simulation operations.

Discussion

We have demonstrated a feasible approach to utilizing a complex, dynamic domain model for the analysis of technical text. The hierarchical nature of the model and the simple interface between the model and the language analyzer should allow this approach to scale up to substantially larger domains. The simulation-based approach is suitable primarily for domains where behavior is largely predictable, but this includes a substantial variety of applications.

The chief hurdle to applying this approach is the large amount of domain information which is required. At present, each new piece of equipment requires a new model. We have begun to explore tools, such as graphical editors, to ease the acquisition of new models. In addition, we believe it will

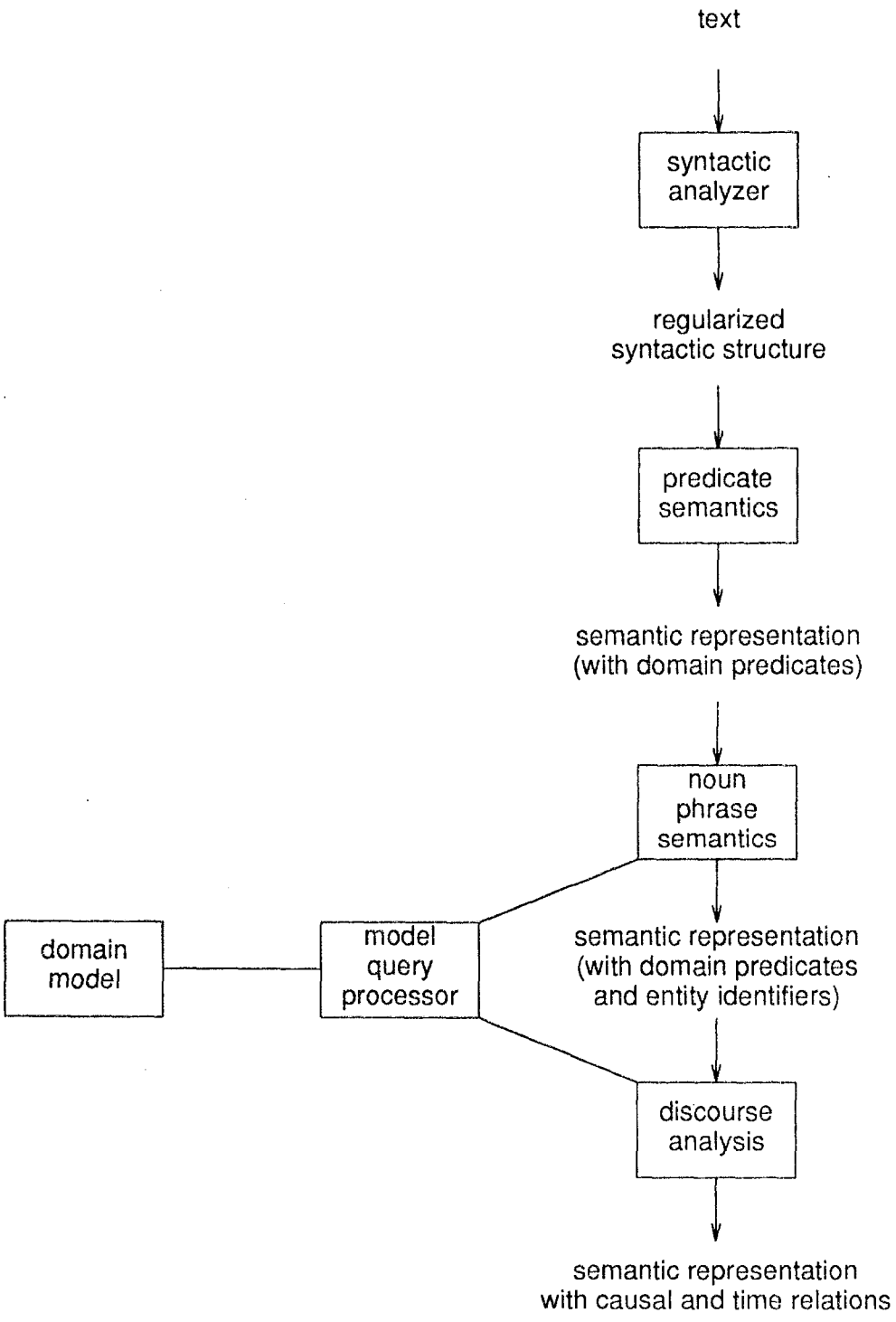


Figure 3. The principal components and data flow of the system.

be necessary to incorporate more general models, which will cover whole classes of equipment.

Acknowledgement

This research was supported by the Defense Advanced Research Projects Agency under Contract N00014-85-K-0163 from the Office of Naval Research.

References

[1] D. R. Dowty. The effects of aspectual class on the temporal structure of discourse: semantics or pragmatics? *Linguistics and Philosophy*, 37-61, 1986.

[2] T. Finin. The semantic interpretation of compound nominals. In *Proc. First Nat'l Conf. on AI*, 1980.

[3] T. Ksiezyk. *Simulation-based understanding of texts about equipment.* PhD Thesis, Computer Science Department, New York University, 1988. Reprinted as PROTEUS Project Memorandum #17, Computer Science Dept., New York University.

[4] T. Ksiezyk and R. Grishman. Equipment simulation for language understanding. *Int'l J. Expert Systems*, **2** (1) 33-78, 1989.

[5] T. Ksiezyk, R. Grishman, and J. Sterling. An equipment model and its role in the interpretation of noun phrases. *Proc. Tenth Int'l Conf. Artificial Intelligence*, 692-695, 1987.

[6] R. Passonneau. A computational model of the semantics of tense and aspect. *Computational Linguistics*, **14** (2) 44-60, 1988.