The Use of Memory in Text Processing

Michael Lebowitz
Department of Computer Science
Columbia University
New York, NY 10027
November, 1985

CUCS-200-85

The Use of Memory in Text Processing

Michael Lebowitz[1]
Department of Computer Science
Computer Science Building, Columbia University
New York, NY 10027
23 January 1986

Abstract

The performance of natural language processing systems should improve as they read more and more texts. This is true both for systems intended as cognitive models and for practical text processing systems. Permanent long-term memory should be useful during all stages of text understanding. For example, if, while reading a patent abstract about a new disk drive, a system can retrieve information about a similar object from memory, processing should be simplified. However, most natural language programs do not exhibit such learning behavior. In part, this is because it is not obvious exactly how memory access should be integrated with other aspects of processing. We describe in this paper how RESEARCHER, a program that reads, remembers and generalizes from patent abstracts, makes use of its automatically generated memory to assist in low-level text processing, primarily involving disambiguation that could be accomplished no other way. We describe both RESEARCHER's basic understanding methods and the integration of memory access. Included is an extended example of RESEARCHER processing a patent abstract by using information about several other abstracts already in memory.

CR classification: I.2.7, Natural language processing, language parsing and understanding, text processing, memory, integrated understanding.

1 Introduction

Virtually all natural language processing systems that have been developed suffer from the same flaw -- reading texts does not make them smarter. They are not able to process texts better over time, even texts that are similar to those read before. This is a clearly a problem for systems intended as cognitive models since human language performance in a domain improves as knowledge is acquired. It is also troublesome for practical systems, since the lack of learning ability requires all information needed for understanding to be hand-coded by the implementor. In this paper we describe a computer system, RESEARCHER, whose language understanding ability improves as it reads, primarily by using a dynamic

---

[1]This research was supported in part by the Defense Advanced Research Projects Agency under contract N00039-84-C-0165. Many people have contributed to RESEARCHER's text processing abilities including John Akbari, Ben Beecher, Tom Ellman, Larry Hirsch, Barbara Moo, Charles Ortiz, Cecile Paris, John Sabella, Alexander Volf and Kenny Wasserman. Part of the memory access algorithm described here was implemented by Tsy Yee. Comments by Kathy McKeown on an earlier draft of the paper were most useful as were comments by Bonnie Lynn Webber and anonymous reviewers.

memory to resolve ambiguities that would otherwise require *ad hoc* disambiguation rules.

RESEARCHER reads patent abstracts and automatically builds up a generalized long-term memory (knowledge base) from them [25, 29]. It provides a framework for studying the ways in which increased knowledge in memory can improve the understanding process. Our main goals in this paper are: 1) to describe RESEARCHER's conceptual analysis understanding structure, which is rather general and easily applicable to a number of tasks that do not require fine syntactic analysis (Section 3), and 2) to show how memory access has been incorporated into this structure, primarily to handle difficult disambiguation problems (Section 4), resulting in a program whose understanding ability improves over time. Section 2 provides background on the ambiguity problems that RESEARCHER must deal with and Section 5 illustrates the operation of RESEARCHER on a typical patent abstract, showing how several other abstracts already in memory are used during processing.

It should be emphasized at the outset that RESEARCHER is an *experimental* system. We use it as a testbed to study issues in generalizing complex object descriptions [25, 30, 50], intelligent question answering [35] and other areas, as well as the natural language processing issues discussed in this paper. As an experimental system, we do not expect RESEARCHER to be able to handle every example we give it perfectly from beginning to end. We believe that the development of RESEARCHER to date indicates that the methods we are using have great potential for the development of intelligent information systems. Despite its experimental nature, RESEARCHER has been developed into a substantial computer system over the last 3 years. It has a dictionary with over 1300 word definitions. It runs on a DECSystem/2060 and processes each patent abstract in a few seconds. Almost any simple example can be processed accurately. Although RESEARCHER is not at the stage where quantitative evaluation of its abilities is useful, the system has been tested on over 130 real patent abstracts. While not all of these are processed perfectly, many are handled quite well, particularly considering the complex nature of the texts. There are indications that with the proper information in memory most of the abstracts could be processed quite successfully. This indicates to us that our memory-based approach will be a fruitful one.

2 Background: Ambiguity and the need for memory access

Human text processing proceeds at many levels simultaneously [5, 32, 42]. Such processing presumably includes the access of detailed, long-term memory for the purpose of finding information relevant to a new text (as shown by the phenomenon of begin *reminded* [40]). Information from memory

should be useful in assisting low-level processing, but exactly how to use such information is a difficult problem.

Earlier work on the use of memory in text processing has suggested that memory access might help in allocating resources (how and when to apply computational effort) and in identifying the important (or interesting) parts of a text [7, 15, 23, 42]. However, these are imprecise ideas and difficult to apply. For example, IPP [22, 26], a program that read, remembered and generalized from news stories about international terrorism, would know from accessing memory that the destination of a hijacking in a story that began, "A United 727 en route to Miami was commandeered ..." was likely to be Havana. However, it was not clear how to use this information if the story simply continued, "to Havana." It is quite easy to process this text without the expectation, and memory just provided a redundant crosscheck. Of course, had no destination been mentioned, IPP would have used Havana as a default. Providing default information was IPP's main use of memory, along with providing a small amount of help in word sense disambiguation. So while IPP did get somewhat smarter as it read, its improvement was not as great as one would have hoped.

It is our feeling that the best way to use detailed memory information during text understanding in the context of current systems is to identify specific tasks during processing where memory can be applied. We will present here a series of "questions" that arise during text processing that can most easily be answered (and often can *only* be answered) by accessing long-term memory. These questions are particularly important in the resolution of ambiguities that must be resolved to achieve the kind of robust performance that is crucial to RESEARCHER's learning task.

We are proposing using *memory* for understanding, as opposed to general semantic information about words or concepts. By memory we mean an appropriately organized knowledge base of detailed information acquired from previously read texts. While general semantic information is crucial for our conceptually-based understanding methods, in order to resolve many understanding questions it will be necessary to look at very detailed information in memory -- in our case, how the objects described in patent abstracts are constructed and how their pieces relate to each other.

To illustrate why memory is needed, consider the following simple example:

EX1 - A read/write head touching a disk made of XXX ...

In EX1 there is no way to determine whether XXX is the material used for the read/write head or for the disk without knowing something about the objects involved. Indeed, different XXX's would be analyzed differently. In some cases, a single XXX could lead to different analyses depending on the exact state of memory. For example, if XXX was "plastic" our analysis would depend on what we knew about plastic disks and plastic read/write heads. We often get the same effect within noun groups such as EX2:

EX2 - A {floppy | hinged} disk support ...

The resolution of ambiguities such as those in EX1 and EX2 is the area where we believe memory will be most immediately useful in text understanding systems (in addition to supplying default values). While it may also be possible to use memory to resolve word sense ambiguity, this has not proven to be a major issue for RESEARCHER. Note that while the specific methods described here are from RESEARCHER and tailored to reading patent abstracts, ambiguity is a problem for all natural language tasks, and we feel our memory-based resolution techniques will be widely applicable.

The problem of disambiguation is not a new one. Linguists have looked at problems of ambiguity for many years. A standard solution to resolve ambiguities such as prepositional phrase and modifier attachment (both of which we will look at) is to appeal to semantic constraints (for example [19, 21]). To take an example adapted from [19], suppose we are trying to analyze, "red car sale", and determine whether "red" modifies "car" or "sale". If our definition of "red" said that it only modified physical objects, then we would get the "right" answer, assuming "car" was defined as a physical object and "sale" was not.

While the use of semantic characteristics is a useful technique, it has several problems that we have already alluded to. The disambiguation information is static and must be pre-defined for the system. The semantic categories that are defined are often quite *ad hoc*. For example, if we were analyzing "red car seat", only a very specialized set of properties would serve to disambiguate. On the other hand, looking in a dynamic memory for information about red cars and red seats should prove effective.

The application of semantic information, including semantic features of words, to resolve ambiguity has also been widely used in Artificial Intelligence research (e.g., [1, 2, 20, 38, 44, 54, 56, 57]).

Constructions similar to those we will look at have been considered in [10, 33, 55]. All of the Artificial Intelligence work, however, has made use of static semantic information specially designed for the program. Particularly interesting to us is recent work that looks at parallel disambiguation as a central part of understanding [5, 49], but, again, this work addresses static, semantic-oriented knowledge bases. Perhaps closest to our work is that of Riesbeck and Martin [37], which takes a radical approach to the use of episodic information in understanding, limiting the use of bottom-up methods much further than we do here. There has also been some work on using the databases connected to database front ends as information sources (e.g., [18]), but primarily at a lexical level.

We will show here exactly how long-term memory can be applied to resolve ambiguity over a large class of text understanding examples. Specifically, we propose to access memory when object descriptions are completed and when these object descriptions are being combined. Typically such processing will be take place at syntactic phrase boundaries, although from our point of view this is largely coincidental. Our research differs from earlier work both in that we indicate specific points at which to use memory (which turn out to be similar to the locations where semantic information is accessed by systems such as [2, 56, 57]) and that we are using a dynamically changing memory rather than fixed semantic properties.

From a conceptual modelling viewpoint, our proposal for integrating memory access with conceptual analysis fits well with existing psychological results. A large body of research, summarized in [12], indicates that language processing is computation-intensive at clause boundaries. Many of the researchers involved have interpreted these results in syntactic terms. However, since the boundaries of conceptual and syntactic units are usually the same, these results are also compatible with a model such as ours.

More recent experiments that measured reading times across various kinds of ambiguity further indicate that interactions between different levels of processing take place primarily at phrase boundaries [48]. Studies of eye movements during text processing [4] lead to much the same conclusion. Research that seems to show that processing ambiguous words requires more effort than non-ambiguous words (e.g., [45, 46, 47]) is also compatible with a model where memory access is performed only at specified points in an understanding algorithm (although such results involve a different level of processing, lexical access, than those we are concerned with).

Our research is also compatible with much of the other Artificial Intelligence work in integrated language understanding. From early systems that used a variety of different kinds of information, such as Winograd's SHRDLU [56], through systems that concentrated primarily on analyzing text in terms of single high-level knowledge structures, e.g., Cullingford's SAM [6] and Wilensky's PAM [53], to systems that used a multitude of high-level structures, e.g., Dyer's BORIS [9], researchers have struggled with how to relate various levels of processing. Even when the knowledge was represented in a uniform format, such as production rules [52], a close examination of the implementations showed that processing would occur at one level and then hand off information to another level at specified points in the processing. This is perhaps largely due to the sequential computer systems we are working with. We use this form of integration explicitly and show how the use of long-term memory can be incorporated into such a framework.

We are *not* contending that human language processing is a strictly linear sequence of events that shifts among various processing levels. We actually believe that fully parallel models such as those proposed in [5, 14, 24, 49] are closer to the truth. However, we do believe that even if various levels of processing are carried out simultaneously, they probably only interact at well-specified processing points. Perhaps significant results at one level -- such as the completion of a conceptual representation or the discovery of a conceptual impossibility – could cause other levels to be aborted. Even parallel processing can most easily be studied -- and most practically implemented on current computers -- by looking at the various levels of processing separately and identifying the points where they interact. That is the approach we are taking in integrating conceptual analysis and memory access.

3 Basic RESEARCHER understanding techniques

RESEARCHER processes patent abstracts by: 1) using basic syntactic rules to identify "pieces" of the ultimate representation and 2) "putting the pieces together" by adding appropriate relations to its representation. EX3 shows a patent abstract, P58, typical of those read by RESEARCHER. We are concerned primarily with abstracts that describe the physical structures of objects. The goal of the text interpretation phase is to build up descriptions of objects, including physical and functional relations between various sub-parts of the objects, using a canonical, frame-based representation scheme [50, 51]. In Section 5 we show how RESEARCHER does this for the first part of P58.

EX3 - P58; United States Patent #4287445; Mark Lienau (Abstract)

> An electromagnetic linear actuator for positioning a transducer over locations on a rotating magnetic recording disk comprising an actuator housing used as a stationary base for supporting various parts; a coil and cart assembly including a cart, having a rectangular cross section and tubular in construction, adapted at one end to support the transducer, and including a direct current coil with a cross section matching that of the cart mounted at the other end thereof; magnetic means affixed to the actuator housing having an air gap for receiving the coil so that the coil is immersed in a magnetic field; and support means affixed to the actuator housing engaging the surfaces of the cart and disposed about the center of gravity of a moving-mass assembly consisting of the coil and cart assembly and the transducer.

There are several interesting points about P58 for text processing purposes. First of all, the structure of the abstract is not quite what we are used to. For example, the "sentences" in the text have no main verbs. We could probably develop an appropriate syntactic grammar for the abstracts using basically the same principles as in analyzing noun groups from normal text. Quite frequently, though, different syntactic structures function quite similarly in patent abstracts (or any sort of text, for that matter). For example, the phrase from P58, "An electromagnetic linear actuator *for positioning* a transducer ..." and a hypothetical alternate phrasing, "An electromagnetic linear actuator *that positions* a transducer ..." are functionally equivalent, but involve different syntactic structures.

In general, subtle differences in syntactic structure do not seem to be crucial in this domain. Patent abstracts thus provide an excellent source of texts for testing strongly semantic-based understanding methods that build conceptual representations directly from text. RESEARCHER uses only the most basic of syntactic information, unlike language programs that perform a more careful syntactic analysis. In any case, most of the difficult disambiguation problems occur at the semantic, rather than syntactic, level. The advantages of minimizing the use of syntactic information (and, in particular of not doing independent syntactic analysis) for cognitive models are discussed in detail in [41]. [26] describes why this is also advantageous for practical systems.

EX4 shows the beginning of P58 segmented in a manner that motivates RESEARCHER's text processing algorithm. P58, and most other patent abstracts that provide physical descriptions, can be broken into segments of two types -- those that describe physical objects (whose representations in memory we refer to as *memettes* -- tiny chunks of memory), shown in italics in EX4, and those that relate the various objects to each other. The memette-describing segments are usually (though not always) simple noun phrases, but the relational segments may take many different forms, including verbs and prepositions. The relational segments are largely functionally independent of their syntactic form, so we

can process them solely on the function they serve, ignoring structural complexities.

EX4 - P58; United States Patent #4287445 (abstract) [segmented]

(*An electromagnetic linear actuator*) (for positioning) (*a transducer*) (over) (*locations*) (on) (*a rotating magnetic recording disk*) (comprising) (*an actuator housing*) (used as) (*a stationary base*) (for supporting) (*various parts*); (*a coil and cart assembly*) (including) (*a cart*), (having) (*a rectangular cross section*) (and tubular in construction), (adapted at) (*one end*) (to support) (*the transducer*), (and including) (*a direct current coil*) (with) (*a cross section*) (matching that of) (*the cart*) (mounted at) (*the other end*) (thereof) ...

The analysis in EX4 leads directly to RESEARCHER's text interpretation algorithm of two sub-phases: memette identification and memette relation, or "identifying the pieces" and "putting the pieces together".

3.1 Text processing overview

The text interpretation methods used in RESEARCHER are based on memory-based understanding techniques designed for IPP (which are described in [26]). Processing involves a top-down goal of recognizing conceptual structures integrated with simple, bottom-up techniques. Since patents are not focused on events, as were the news stories IPP processed, the action-based methods of IPP (or other conceptual analyzers, e.g., [1, 38, 54]) must be extensively modified in a manner consistent with the analysis shown in EX4.

RESEARCHER uses a functional classification of words that concentrates on those that refer to physical objects and those that describe physical and functional relations between such objects, including words that indicate assembly/component relations. RESEARCHER does careful processing of object-describing phrases (usually noun phrases) to identify memettes, modifications to memettes, and references to previous mentions of memettes. This processing is interspersed with the application of relational words that create relations among memettes.

In broad terms, the structure of our processing is similar to the cascaded ATN methodology [2, 57], where a syntactic grammar frequently hands off syntactic components to a semantic analyzer that builds semantic structures and eliminates impossible constructs. In all likelihood, the memory check methods suggested here could be applied to such methods, making them dynamic and more robust. Due to the nature of our domain, we are able to use only a small number of different syntactic constructs, eliminating the need for a formal syntactic grammar by focusing on the role of words in the conceptual

representation. Also, the cascaded ATN methodology views the understanding process as a syntactic processor passing off what it finds to the semantic analyzer, we look on the process as being primarily one of a conceptual analysis that makes use of syntactic structures when needed (much as in FRUMP [8]).

3.2 Memette Identification: Finding the pieces

Since the descriptions read by RESEARCHER focus on how objects relate to each other, the identification of objects is obviously crucial. "Finding the pieces" consists primarily of bottom-up recognition of simple noun phrases followed by a reference component that determines whether the object being mentioned has been previously mentioned in the text. No explicit syntactic analysis of complex noun phrases is done. Prepositional phrase attachment occurs as part of relating memettes.

The noun phrase recognition process involves the same "save and skip" strategy described in [26] (which is similar to the processing described in [13]). Using a one-word look-ahead process, RESEARCHER saves noun phrase words in a stack until the head noun is found. Then the words in the stack are popped off and used to modify the memette indicated by the head noun. Noun group recognition could also be done easily with simple bottom-up syntactic processing (e.g., [11]).

Determining how the words within a simple noun group relate to each other is a problem that, as we will see in Section 4, is heavily dependent upon memory access. For example, in a typical phrase such as, "a fixed head disk drive assembly", there is no way of knowing whether "fixed" modifies "head", "disk", "disk drive" or "assembly" without using knowledge about the structure of disk drives.

The final portion of "finding the pieces" involves checking for anaphoric reference to the object described. Here RESEARCHER is able to take advantage of some of the arcane nature of patent abstracts. A very strict formalism is used for reference in patents, involving the word "said" and exact repetition of identifying modifiers. Without such formal language, the reference process would be very complicated, as an abstract can refer to many very similar objects. As it is, we can use an uncomplicated reference procedure that avoids many techniques needed for other sorts of text. In some domains, memory access might also be useful in performing such references.

3.3 Memette relation: Connecting the pieces

The second major sub-phase to RESEARCHER text processing involves putting together identified memettes to build a final representation. This process occurs as soon as the objects involved are found. By and large, there are three different kinds of relations that tie objects together -- assembly/component, physical and functional relations between memettes. The basic RESEARCHER strategy for each is the same -- maintain information from the relational segments of the text in short term memory and then, when the next memette is identified, determine how the appropriate objects relate to each other. The three classes of relations are, however, handled differently in memory update and generalization. This process, which is largely independent of the form of the relational text segments, immediately builds a conceptual representation for later use.

In uncomplicated texts, the relational process is straightforward. Specified memettes are related using a few simple focus techniques including some related to those of Grosz [17] and Sidner [43]. Basically, the system assumes that at any point in the text future relational phrases will refer to a single, "in focus" object. However, as texts get more complicated, the relational situation frequently becomes highly ambiguous, requiring the application of memory for resolution. Memette relation is one of the crucial spots in the processing where memory must be applied. We will see how and why memory is applied to this problem in Section 4.

4 Using memory in RESEARCHER during text processing

In this section, we show specifically how a long-term memory built up from texts can be used to resolve certain classes of ambiguity by having the text understanding process "ask questions" of memory. We must specify exactly what questions should be asked and when. The questions RESEARCHER will ask involve determining which of several possible physical structures is more plausible. Section 4.2 describes how these questions are answered. As well as specifying the specific cases we have identified where we believe memory should be accessed, we also wish to illustrate the level at which we believe memory access should be applied during text processing.

We should reiterate that none of the ambiguities noted here are particularly novel. What is new is the integration of the disambiguation questions into conceptual analysis methods and the use of an automatically generated dynamic memory to answer the questions.

Concrete noun phrases in English, phrases that describe objects, can have very complex structures.

(See [13] for a conceptual analysis approach to noun phrase processing.) RESEARCHER's emphasis on object descriptions requires detailed attention to the internal structure of noun phrases, so that we can identify how the pieces of such phrases fit together. Memory application is crucial in doing this.

EX5 shows the first question requiring memory.

EX5

Form: object-word1 object-word2

Example: An actuator housing ...

Question: What is the relation between object-word1 (actuator) and object-word2 (housing)?

Noun-noun constructions in English can hide a number of different underlying relationships even with no syntactic ambiguity. In the example above, the housing could be part of the actuator, contain the actuator, or be used by the actuator, among many other possibilities. One of the most important tasks for RESEARCHER is to determine the proper relationship in such cases.

Linguistic research provides much insight into this process. There has been considerable effort put into the analysis of complex nominal phrases. Perhaps the most interesting from our point of view is the work of Levi [31]. (Some of Levi's ideas were applied in Finin's Artificial Intelligence work on noun phrases [10].) Levi proposes that all noun-noun constructions can be analyzed in terms of predicate nominalization or predicate deletion. The first case includes examples such as "city planner" or "oil imports" where one noun serves primarily to add specification or to specify a role of the other.[2] While such cases are important, we have found predicate deletion more common in our domain. In the predicate nominalization case, often one noun or the other is not what we would consider to be an object description word, although it may specify a case restriction. Predicate deletion occurs in cases where there is an implicit predicate between the two nouns -- for example, "transducer assembly" (an assembly *that contains* a transducer) or "actuator housing" (a housing *for* an actuator).

Levi not only proposes that a large class of complex nominals can be described in terms of predicate deletion, but also that there are only nine possible implicit conceptual predicates -- *cause, have, make, use, be, in, for, from* and *about*. We will not argue here whether these predicates are sufficient for

[2]This construction can also be expressed with a prepositional phrase, e.g., "planner of cities", but the predicate nominal appears to be the more natural form.

linguistic analysis. As Levi states, for many purposes, including ours, the predicates will have to be made much more specific. Our problem with noun-noun constructions becomes determining the correct relation between the memettes in question. In our representation scheme, this relation can be either one of many physical relations, one of many functional relations, an assembly/component relation or a component/assembly relation.[3] The answer can best be found by looking at examples in memory.

Notice carefully that memory, not just general semantic information, is needed here. (RESEARCHER does use general information if no specific examples are available.) While for most disk drives the housing is likely to be a part of the actuator (as is the case for P58), the converse could be true for another device. Unless we already knew about every possible disk drive, we could not possibly prepare our system for all the examples it might encounter. The same will hold for any complex domain.

Nouns are not the only element in complex nominals. We also have to properly apply modifiers, as in EX6.

EX6

Form: modifier object-word1 object-word2

Example: A metal drive cover ...

Question: Does the modifier (metal) apply better to object-word1 (drive) or object-word2 (cover)?

EX6 shows how a modifier preceding a noun-noun combination can modify either of the objects mentioned. In EX6, either the drive or the cover could be made of metal. In more complex situations, i.e., more nouns or a series of modifiers, syntax can reduce the possible targets of a modifier, but only by asking memory which modification is more plausible in the context of the object being described can the right choice be made.

EX7 illustrates a somewhat similar noun group problem.

[3]It is possible to represent the physical and functional relations as combinations of simpler, canonical fields (see [51]), but the text can still refer to many stereotypical *combinations* of these fields.

EX7

Form: object-word1 object-word2 object-word3

Example: A disk-drive transducer wire ...

Question: Is object-word3 (wire) "related to" (in the sense of EX5) object-word1 (disk-drive) or object-word2 (transducer)?

When multiple nouns appear in succession, it is not always easy to tell how they group together. In EX7, there us no way to tell whether the wire is related to the disk drive or the transducer. The nature of the relation must also be determined as in EX5. Again, an appeal to long-term memory of similar devices is the best way to solve this problem.

The "putting together the pieces" phase of RESEARCHER processing also involves ambiguities that must be resolved with memory. EX8 shows what can happen when more than one relational word is processed.

EX8

Form: object-word1 relation-word1 object-word2 relation-word2 object-word3

Example: A transducer on top of a disk supporting a spindle ...

Question: Does relation-word2 (supported by) connect object-word3 (spindle) with object-word1 (transducer) or object-word2 (disk)?

In one sense, the problem shown in EX8 involves a noun group problem, in that it deals with modifying phrase attachment. However, in our understanding scheme, it falls into a different category. As described in Section 3, relating various objects together is a separate part of processing. This allows RESEARCHER to handle many other syntactic manifestations of this problem with the same mechanism. The mechanism here is to query memory about which of the possible objects most appropriately takes part in the specified relation. In EX8, we need to appeal to memory to determine whether the spindle is more likely supported by the transducer or the disk, either of which is syntactically appropriate.

EX9 illustrates a similar problem, this time with "part indicators" (words that introduce a list of a part's subparts).

EX9

Form: object-word1 part-indicator1 object-word2 part-indicator2 object-word3

Example: A disk drive including a disk with a metal plate (and) ...

Question: Is object-word3 (metal plate) a part of object-word1 (disk drive) or object-word2 (disk)?

In patent abstracts, there are frequently descriptions of an object's parts followed by recursive descriptions of subparts' components. This often creates considerable ambiguity of the sort shown in EX9, which is structurally similar to EX8. We handle this case separately, as the assembly/component relations are represented differently from other relations, since they are so crucial, and are expressed slightly differently in the abstracts. Once again, the only way to determine the correct analysis, in this case whether the metal plate is part of the disk drive or the disk, is to query memory, quite possibly looking at descriptions of specific objects or classes of objects. The same is true for ambiguities involving a combination of physical, functional and assembly/component relations.

4.1 Integrating memory access with text processing

Asking the disambiguation questions described in the previous section fits in nicely with the overall RESEARCHER text processing algorithm. RESEARCHER's "save and skip" noun group strategy naturally accommodates memory-based disambiguation. As the program is processing the items from its short-term memory stack, it keeps track of the identified memettes that can be further modified (the memettes described by the head noun, the most recent noun, and possibly intermediate nouns). Then, if there is more than one distinct object described in the noun group, as RESEARCHER continues to work back through the stack, memory can be queried to determine which object new words modify or relate to. Without performing the memory query, it is necessary to employ a set of complex and rather unsatisfying heuristics to determine how the parts of noun groups fit together.[4] Before there is anything in memory, this is what RESEARCHER has to do, but as memory grows, we can achieve much better results by using its knowledge base. For new domains, people probably initially apply information from other domains, which is not available to our system.

Memory access fits in equally well in the "putting together the pieces" phase of RESEARCHER's

[4]A typical heuristic is: If one word in a noun-noun construction describes an object that normally has no subparts, and the other is a vague assembly word (e.g., "structure") then assume that the first is a part of the second. Another heuristic states that if no other rules apply, assume an unspecified functional relation.

text processing algorithm. RESEARCHER maintains short-term memory buffers with the objects that can be targets of a new relation, usually those specified by the head noun of a noun phrase and the most recent noun (although there are other possibilities). Then, if these objects are different, and a new relation (physical, functional or assembly/component) is being established, memory can be queried about which of the objects more plausibly relates to the new object. As with noun group processing, it is possible to develop heuristics that handle most cases, but they are complex and do not seem to be the right way to go for robust understanding.

4.2 Using memory to answer the questions

We have shown how certain ambiguities in language can be resolved with the answers to specified questions. The questions we have looked at take the form of asking which of a set of objects can more reasonably be modified in a certain way or relate to another object or what is the most plausible relation between two objects. Phrased another way, the needed memory process is to determine which of a set of partial descriptions of a device is most plausible.

Our basic approach to memory in RESEARCHER is to store objects in terms of a hierarchy of automatically generalized prototypes created by noticing similarities among representations [25, 27, 30, 29, 50]. With this sort of memory we can take partial descriptions of objects and determine whether they correspond to objects or generalized objects in memory by searching through the hierarchy of prototypes. We do not currently try to answer questions that require complex inferences. We aim to have as complete as possible a set of examples in memory so that the program can almost always find relevant examples.

Figure 1 schematically shows the structure of the generalization-based memory that RESEARCHER uses. Memory is basically a hierarchy of object-describing frames allowing inheritance in the manner of semantic networks [36], frame systems [34], MOPs [39, 40] or KL/ONE [3]. RESEARCHER allows inheritance of objects' structures, as well as physical properties. The crucial point about generalization-based memory is that the hierarchy of object descriptions is automatically created from examples.

Figure 2 shows how memory might look after a series of disk drive descriptions has been processed. The top level concept is **disk-drive#** (the # is used to distinguish object concepts from words) which contains the information common to all disk drives. It organizes one instance that could not be described by any more specific generalized objects (patent A). There are two more specific versions of

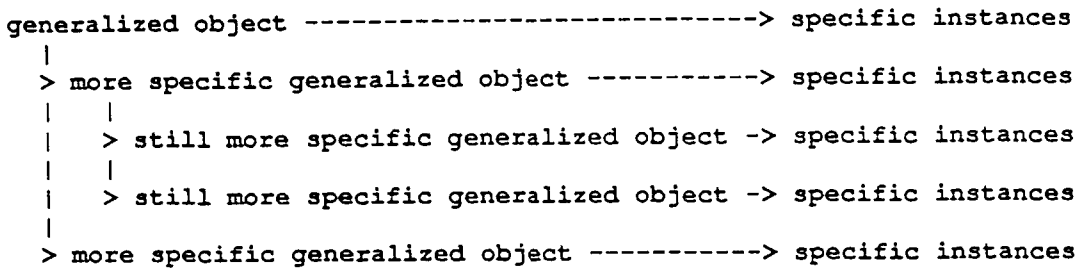

Figure 1: Schematic illustration of generalization-based memory

disk-drive#: **floppy-disk-drive#** and **hard-disk-drive#**. There are also two more specific versions of **floppy-disk-drive#**. Each concept organizes specific instances that it describes.

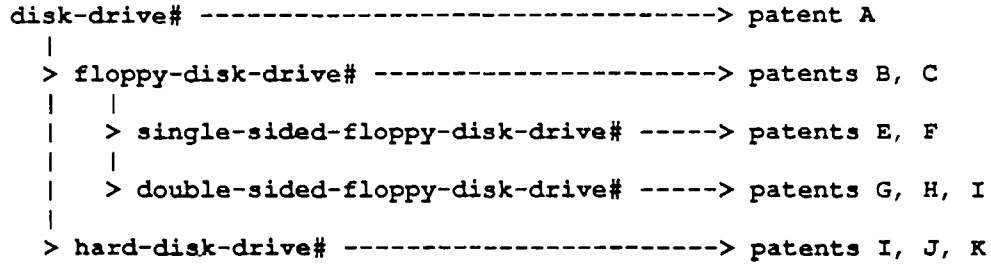

Figure 2: Hypothetical instance of generalization-based memory

There are several factors that make the structure of RESEARCHER's memory more complex than Figure 2. First of all, the generalized concepts consist largely of structural descriptions that specify their parts. Each of these parts (e.g., disks, read/write heads) are in their own generalization hierarchy. Also, the examples used to build up memory rarely lead to a hierarchy as nice as the one in Figure 2. The factors that complicate memory organization include: hierarchies need not take the form of binary trees, the same object can be stored in more than one place in memory (as both a hard disk drive and a high-density disk drive, for example) and the form of memory depends on the order of presentation of examples. There are also many problems in generalizing objects that are themselves hierarchical, many of which are described in [50]. Despite all these caveats, however, we can still make use of memory in text processing. The quality of the help provided will simply depend upon the quality of memory.

Our basic plan for using memory in text processing is to look for examples of the possible cases specified by the text. So, for example, in the situation illustrated in EX8, if we need to know whether object X is more likely to be a part of object Y or object Z (as in "a Y with a Z including an X ..."), we simply look in memory for cases where X is a part of a Y or part of a Z. Similarly, if we have the

noun-noun construction, "an X Y", where we need to know the relation between X and Y, we look for existing relations in memory between X and Y. Notice that this method implies that our system can only find genuinely new constructions from unambiguous text.

Searching for relevant examples in a generalization-based memory is not difficult. Since, as we mentioned briefly above, each object is part of its own generalization hierarchy, and the hierarchies are indexed, we can efficiently find modifiers applied to objects or relations among objects. So, for example, if RESEARCHER is trying to decide whether read/write heads or enclosures are more likely to be magnetic (as in "magnetic read/write head enclosure ..."), it starts out with its most general read/write head and enclosure object descriptions, and searches through the more specific forms of these concepts until it finds a case with one or the other being magnetic. Similar processing is done for relations between objects.

Obviously our scheme for using memory is somewhat oversimplified. It is certainly possible to imagine cases where our questions could be answered through an inference process and yet no specific examples found in memory. However, by and large, most of the important inferences will be captured by the generalized object descriptions in memory. Thus we are quite satisfied to look for examples. Another issue involves what to do when our assumptions taken from memory prove to be wrong. While we have not currently implemented a method for reversing incorrect disambiguation choices, we might at some point consider techniques of the sort described in [28].

4.3 A simple example of memory use

As a simple illustration of RESEARCHER using memory in text understanding that shows how performance depends upon the contents of its memory, consider the noun phrase, EX10:

EX10 - A motor spindle ...

As discussed above (with EX5), the noun-noun construction in EX10 is ambiguous in terms of the relation between the nouns "motor" and "spindle" (actually, the concepts they refer to). We will first show how EX10 is processed with no relevant information in memory. Then we will put another simple example memory and show how RESEARCHER uses it to process EX10.

Figure 3 shows how RESEARCHER processes EX10 with no relevant information in memory. In this

example, and others, lines beginning with ">>>" indicate queries of memory. Responses are indicated by "<<<". The letters in the text representation refer to relations listed underneath. In more complex examples, we will see the representation in the form of a tree that specifies assembly/component relations. The &MEM's represent specific instances of abstract memettes.

```
Running RESEARCHER at 14-Jan-86 16:10:56

(A MOTOR SPINDLE)

Processing:

A               : New instance word -- skip
MOTOR           : Memette within NP; save and skip
SPINDLE         : MP word -- memette DRIVE-SHAFT#

New DRIVE-SHAFT# instance (&MEM1)
>>> Looking for relation between MOTOR# and &MEM1 (DRIVE-SHAFT#)
New MOTOR# instance (&MEM2)
Assuming &MEM2 (MOTOR#) and &MEM1 (DRIVE-SHAFT#) are functionally related
Establishing UNKNOWN-PURP-REL relation; SUBJECT: &MEM2 (MOTOR#);
  OBJECT: &MEM1 (DRIVE-SHAFT#) [&REL1]

Text Representation:

----------------------A-1 1 = DRIVE-SHAFT#

----------------------A-2 2 = MOTOR#


A list of relations:

           Subject:              Relation:          Object:

[&REL1/A]  &MEM2 (MOTOR#)        {UNKNOWN-PURP-REL} &MEM1 (DRIVE-SHAFT#)
```

Figure 3: 'Motor spindle' with memory empty

The key processing in Figure 3 takes place at the end, when "spindle" is read. As RESEARCHER works backwards through the words it has "saved and skipped", it must determine the relation between the concepts **drive-shaft#** (the definition of "spindle") and **motor#**. It searches memory for a relation between these concepts. Finding none, it can only postulate an indeterminate functional relation between the objects.

The situation is quite different if there is relevant information in memory. For example, suppose EX11 had been in memory.

EX11 - A drive with a motor that includes a spindle.

RESEARCHER's representation of EX11, with the drive shaft as part of the motor, is shown in Figure 4. Figure 5 shows how the processing of EX10 would take place in this case.

```
Text Representation:

                                                1 = DRIVE#
-------------1|-------------2|-------------3 2 = MOTOR#
                                                3 = DRIVE-SHAFT#


A list of relations:

          Subject:              Relation:          Object:
                                 <none>
```

Figure 4: Setting up memory

```
Running RESEARCHER at 14-Jan-86 16:11:47

(A MOTOR SPINDLE)

Processing:

A              : New instance word -- skip
MOTOR          : Memette within NP; save and skip
SPINDLE        : MP word -- memette DRIVE-SHAFT#

New DRIVE-SHAFT# instance (&MEM4)
>>> Looking for relation between MOTOR# and &MEM4 (DRIVE-SHAFT#)
<<< Found HAS-PART relation(s) between &MEM2 (MOTOR#) and &MEM3 (DRIVE-SHAFT#)
New MOTOR# instance (&MEM5)
Assuming &MEM4 (DRIVE-SHAFT#) is part of &MEM5 (MOTOR#)

Text Representation:

-----------------5|-----------------4 4 = DRIVE-SHAFT#
                                      5 = MOTOR#


A list of relations:

          Subject:              Relation:          Object:
                                <none>
```

Figure 5: 'Motor spindle' with EX11 in memory

The processing of EX10 is now quite different. When RESEARCHER looks for a relation between **drive-shaft#** and **motor#**, it finds the information from EX11 and assumes that the new example follows the same pattern -- that there is an assembly/component relation between the two objects. This shows

clearly how memory can help resolve ambiguity using only information that must be maintained in any case for generalization purposes. RESEARCHER is actually looking at both generalized and real objects in memory for guidance. In a fully developed memory, a simple relation like the one in EX11 would probably occur as part of a generalized object description. An example such as EX11 could also have been used to disambiguate phrases such as "motor spindle cover" or "an assembly with a drive including a motor".

4.4 Problems in memory access

The series of examples in Section 4.3 introduces an important question. What happens if more than one relevant example can be found in memory? Obviously this will occur frequently when memory is complex. At the moment, we have one heuristic for use in this case. When multiple relevant examples are found in memory, those nearest the top of the generalization hierarchy, i.e., in the least specific generalized object descriptions, are used. This seems logical, as a general object description applies to a wide range of cases. However, this heuristic will require the construction of a large memory to be tested. The same is true in considering examples where one of the ambiguous cases is so obvious that Grice-type considerations [16] indicate that the case would not be likely to occur in text.

Several other problems arise in looking for relevant examples for use in text understanding. We will mention two here. First, our scheme would require refinement to handle certain elements of context in a very broad domain. For example, suppose RESEARCHER's memory had descriptions of stereo turntables as well as disk drives. The relation between objects such as motors and drive shafts might be very different for the two devices. A search for a relation between new instances of these objects begun by starting at **motor#** and **drive-shaft#** could easily find the wrong relation. For the moment, we have avoided the problem by only considering a relatively narrow domain.

A second problem is that matching objects in a new example with other examples or generalized object descriptions is not a trivial task. In the disk drive/turntable example, while texts might describe the motors in each device with the same word, they are not truly the same conceptually. RESEARCHER would note the difference by representing the two objects differently in memory. The trick is then to note that we might want to consider the objects as being "the same" for purposes of disambiguation (e.g., in identifying the relation between the motor and the drive shaft). More strikingly in this example, we might wish to place in correspondence the disk drive's read/write head with the turntable's cartridge, since the relations involved are similar.

This matching process is a difficult one. We currently assume that objects with a "close" common ancestor in the generalization hierarchy match for search purposes. It turns out that exactly the same matching process is needed for the generalization process (see [50]), so we can apply solutions found there to our text processing memory search.

5 A RESEARCHER example

We will complete our presentation of RESEARCHER's use of memory in text understanding by showing how the program processes the first part of the real patent abstract we looked at in Section 3 (P58) including how it makes use of other examples. At the risk of making the program look a bit foolish, we will first show how RESEARCHER understands P58 without any other examples in memory. RESEARCHER's representation of P58 in this case is shown in Figure 6. The representation consists of a set of identified memettes, a main part hierarchy and several parts that are not directly part of the actuator. There is also a set of relations between memettes. The relations prefixed with R- are physical and those beginning with P- are functional (purposive).

The representation in Figure 6 is actually not all that bad. Other than one major mistake, which will be discussed shortly, most of the relationships in the representation are correct, or at least plausible. However, and this is the key point, these relationships were determined in the face of extreme ambiguity by relatively *ad hoc* disambiguation rules, or in some cases, just by plain luck. As we will see below, the method by which RESEARCHER resolves ambiguity using information in memory is much more satisfying and potentially much more robust.

The one major mistake made when producing the representation in Figure 6 illustrates this point nicely. RESEARCHER assumed that the list of objects following the word "comprising" were parts of the disk, not the actuator. A close look at P58 reveals that this is syntactically plausible. The mistake could easily be corrected with a heuristic suggesting that an actuator is more complex than a disk and hence more likely to have parts. Similarly, some sort of focus heuristic could also be devised (though focus seems to be used unusually in patent abstracts). However, the robustness of such heuristics is problematic, unlike, we feel, our memory-based methods.

To illustrate the use of memory in processing, there will have to be other examples in memory for RESEARCHER to find. We will show how RESEARCHER processes P58 after having read the following

```
Running RESEARCHER at 19-Jan-86 19:57:10
Patent: P58

(AN ELECTROMAGNETIC LINEAR ACTUATOR FOR POSITIONING A TRANSDUCER OVER
LOCATIONS ON A ROTATING MAGNETIC RECORDING DISK COMPRISING AN ACTUATOR
HOUSING USED AS A STATIONARY BASE FOR SUPPORTING VARIOUS PARTS *SEMI* A COIL
AND CART ASSEMBLY INCLUDING A CART *COMMA* HAVING A RECTANGULAR CROSS
SECTION AND TUBULAR IN CONSTRUCTION *COMMA* ADAPTED AT ONE END TO SUPPORT
THE TRANSDUCER *STOP*)

Text Representation:

----------------------AE-1 1 = TYPE/PURPOSE/ELECTROMAGNETIC
                               CONFIGURATION/LINEAR ACTUATOR#

--------------------ABK-2 2 = TRANSDUCER#

               |-----------B-3                  3 = NUMBER/>1 LOCATION#
               |----------EF-5                  4 = CONFIGURATION/CYLINDRICAL
               |                                    TYPE/PURPOSE/MAGNETIC DISK#
               |---------FGH-6                  5 = ENCLOSURE#
               |----------GH-7                  6 = MOBILITY/NONE BASE#
----------CD-4|               |-------------9  7 = NUMBER/SOME PART#
               |               |------------10  8 = UNKNOWN-ASSEMBLY#
               |-------------8|------------11  9 = CARRIAGE#
                               |--------IJK-12 10 = COIL#
                                                11 = SHAPE/RECTANGULAR
                                                     CROSS-SECTION#
                                                12 = NUMBER/1 END#


A list of relations:

           Subject:              Relation:          Object:

[&REL1/A]  &MEM1 (ACTUATOR#)     (P-GUIDES)         &MEM2 (TRANSDUCER#)
[&REL2/B]  &MEM3 (LOCATION#)     (R-ABOVE)          &MEM2 (TRANSDUCER#)
[&REL3/C]                        (P-WRITES)         &MEM4 (DISK#)
[&REL4/D]                        (P-ROTATES)        &MEM4 (DISK#)
[&REL5/E]  &MEM1 (ACTUATOR#)     (UNKNOWN-PURP-REL) &MEM5 (ENCLOSURE#)
[&REL6/F]  &MEM5 (ENCLOSURE#)    (P-ACTS-AS)        &MEM6 (BASE#)
[&REL7/G]  &MEM6 (BASE#)         (P-SUPPORTS)       &MEM7 (PART#)
[&REL8/H]  &MEM6 (BASE#)         (R-CONNECTED-TO)   &MEM7 (PART#)
[&REL9/I]  &MEM12 (END#)         (P-MODIFIES)
[&REL10/J] &MEM12 (END#)         (R-AT)
[&REL11/K] &MEM12 (END#)         (P-SUPPORTS)       &MEM2 (TRANSDUCER#)
```

Figure 6: P58 without any other examples in memory

three excerpts from other patent abstracts.

EX12 - P82; United States Patent #4305105; Bin Ho and Charles Dong (Abstract)

A linear actuator for a magnetic disc drive has shortened magnetic flux lines totally confined within the actuator housing wnereby the actuator can be placed in closer proximity to a disc. The actuator includes a generally cylindrical housing with magnets attached to the inside surface of the housing.

EX13 - P94; United States Patent #4034613; Martin Halfhill and Russell Brunner (Abstract)

A disc drive memory device is described utilizing a continuously rotating drive shaft and a roller which rides thereon to effect translational motion of a carriage to move a read/write head between address locations on a magnetic recording surface of a data storage disc.

EX14 - P137; United States Patent #4400750; Forestlane Co. Ltd. (Abstract)

A magnetic read/write head carriage assembly for a floppy disk drive is disclosed for use with double sided floppy disks. The head carriage assembly comprises a coil spring, having a central coil portion and first and second ends, which is mounted in a position between the base and the head support arm.

Figures 7, 8 and 9 show the representations created and incorporated into memory by RESEARCHER for EX12 (P82), EX13 (P94) and EX14 (P137). For purposes of understanding P58, the key point in P82 is that it describes an actuator that includes an enclosure ("housing") as a part. The unknown functional relation between these parts will also prove significant. The relevant parts of P94 are the locations on the disk and the carriage as part of an assembly. P137 will contribute the description of an assembly that includes both a coil and a carriage. Note that even if every detail of a representation is not correct, it is still possible for RESEARCHER to make use of the parts that are correct. It is unlikely that any blatantly incorrect relations will prove relevant in processing later texts.

```
Patent: P82

Text Representation:

                                    |-------E-7 1 = CONFIGURATION/LINEAR
                                    |             ACTUATOR#
            |--------CD-1|-------C-6|-------E-8 2 = DRIVE#
            |                                   3 = TYPE/PURPOSE/MAGNETIC
            |                                       DISK#
---------A-2|                                   4 = SIZE/SHORT MAGNETIC
            |                                       NUMBER/>1 LINE#
            |--------AD-3                       6 = CONFIGURATION/CYLINDRICAL
            |                                       ENCLOSURE#
            |---------B-4                       7 = NUMBER/>1 MAGNET#
                                                8 = LOCATION/INTERIOR SURFACE#

----------------------B-5 5 = FLUX#

A list of relations:

           Subject:              Relation:          Object:

[&REL1/A]  &MEM3 (DISK#)         (UNKNOWN-PURP-REL) &MEM2 (DRIVE#)
[&REL2/B]  &MEM5 (FLUX#)         (UNKNOWN-PURP-REL) &MEM4 (LINE#)
[&REL3/C]  &MEM1 (ACTUATOR#)     (UNKNOWN-PURP-REL) &MEM6 (ENCLOSURE#)
[&REL4/D]  &MEM3 (DISK#)         (R-CLOSE-TO)       &MEM1 (ACTUATOR#)
[&REL5/E]  &MEM7 (MAGNET#)       (R-CONNECTED-TO)   &MEM8 (SURFACE#)
```

Figure 7: RESEARCHER representation of EX12

```
Patent: P94

Text Representation:

         |-------10                                          9 = UNKNOWN-ASSEMBLY#
         |-----F-11|----FL-12|-----J-18|-----K-17 10 = MEMORY#
         |                                                 11 = DRIVE#
         |-----G-13                                        12 = DISK#
---------9|-----H-14                                       13 = DRIVE-SHAFT#
         |-----H-15                                        14 = ROLLER#
         |----IK-16                                        15 = CARRIAGE#
                                                           16 = TYPE/PURPOSE/READ-WRITE
                                                                TRANSDUCER#
                                                           17 = NUMBER/>1 LOCATION#
                                                           18 = TYPE/PURPOSE/MAGNETIC
                                                                SURFACE#

---------------------L-19 19 = DATA#

A list of relations:

           Subject:              Relation:          Object:

[&REL6/F]  &MEM12 (DISK#)        {UNKNOWN-PURP-REL} &MEM11 (DRIVE#)
[&REL7/G]                        {P-ROTATES}        &MEM13 (DRIVE-SHAFT#)
[&REL8/H]  &MEM14 (ROLLER#)      {R-ON-TOP-OF}      &MEM15 (CARRIAGE#)
[&REL9/I]  &MEM16 (TRANSDUCER#)  {P-MOVES}
[&REL10/J]                       {P-WRITES}         &MEM18 (SURFACE#)
[&REL11/K] &MEM16 (TRANSDUCER#)  {R-BETWEEN}        &MEM17 (LOCATION#)
  object2: &MEM18 (SURFACE#)
[&REL12/L] &MEM19 (DATA#)        {P-WRITES}         &MEM12 (DISK#)
```

Figure 8: RESEARCHER representation of EX13

Figure 10 shows how the first part of P58 is processed. As before, the lines marked with ">>>" are the points where memory is queried, and "<<<" indicates replies.

Processing of P58 begins with "an electromagnetic linear actuator". We can see in Figure 10 how RESEARCHER processes this noun group by saving and skipping the words "electromagnetic" and "linear" until the head noun, "actuator" is reached. It then works back through the noun group, applying the modifiers to **actuator#**. Even though this case is not ambiguous, RESEARCHER still looks in memory for relevant examples. In doing so it finds the example of the linear actuator from P82. Next to be established is a purposive relation, P-GUIDES, taken from the word "positioning" (after "for", which in this case indicates that a purpose word is to follow).

When processing the phrase "over locations", RESEARCHER runs into its first true ambiguity, one like EX8 in Section 4. In establishing the R-ABOVE relation from "over", RESEARCHER must decide

```
Patent: P137

Text Representation:

                              |------------21 20 = TYPE/PURPOSE/MAGNETIC
                              |                   UNKNOWN-ASSEMBLY#
                              |----------R-22 21 = CARRIAGE#
                              |----------O-25 22 = TYPE/PURPOSE/READ-WRITE
                              |                   TRANSDUCER#
                 |------------20|----------O-26 23 = DRIVE#
---------MN-23|               |----------P-27 24 = TYPE/PURPOSE/TWO-SIDES
                 |               |                   TEXTURE/SOFT DISK#
                 |               |---------PS-28 25 = SPRING#
                 |                                   26 = LOCATION/CENTER COIL#
                 |                                   27 = NUMBER/2 END#
                 |---------MN-24                     28 = LOCATION#

---------------------S-29 29 = BASE#

-------------------QR-30 30 = ARM#


A list of relations:

           Subject:              Relation:          Object:

[&REL13/M] &MEM24 (DISK#)        {UNKNOWN-PURP-REL} &MEM23 (DRIVE#)
[&REL14/N] &MEM23 (DRIVE#)       {P-USED-FOR}       &MEM24 (DISK#)
[&REL15/O] &MEM26 (COIL#)        {UNKNOWN-PURP-REL} &MEM25 (SPRING#)
[&REL16/P] &MEM27 (END#)         {R-INSIDE-OF}      &MEM28 (LOCATION#)
[&REL17/Q] &MEM30 (ARM#)         {P-SUPPORTS}
[&REL18/R] &MEM22 (TRANSDUCER#)  {UNKNOWN-PURP-REL} &MEM30 (ARM#)
[&REL19/S] &MEM28 (LOCATION#)    {R-BETWEEN}        &MEM29 (BASE#)
  object2: &MEM30 (ARM#)
```

Figure 9: RESEARCHER representation of EX14

whether the "transducer" or the "actuator" is over the "locations". In searching memory to resolve this ambiguity, RESEARCHER finds an instance of a transducer and locations in an R-BETWEEN relation from P94. It cannot use this example to resolve the ambiguity involving R-ABOVE, so it uses a simple syntactic heuristic. A more sophisticated version of the program could perhaps consider the connection between R-BETWEEN and R-ABOVE.

A similar ambiguity arises when "a rotating magnetic recording disk" is reached. The internal "identify the pieces" processing is similar to that for the first noun group, saving the modifiers and then applying them to **disk#**. RESEARCHER must decide whether the "part of" word, "on", indicates that the "locations" or the "actuator" is part of **disk#**. (If the second reading is not obvious, imagine the word "and" before "on". This reading is syntactically possible, with or without the word "and" being present.)

RESEARCHER's first choice of how to resolve this ambiguity is with a memory check. It looks for cases in memory where either the concept **location#** or **actuator#** is part of **disk#**, and finds the example of locations on disks from P94. This is used to resolve the ambiguity and RESEARCHER creates a

```
Running RESEARCHER at 16-Jan-86 01:04:15
Patent: P58

(AN ELECTROMAGNETIC LINEAR ACTUATOR FOR POSITIONING A TRANSDUCER OVER LOCATIONS
ON A ROTATING MAGNETIC RECORDING DISK COMPRISING AN ACTUATOR HOUSING USED AS A
STATIONARY BASE FOR SUPPORTING VARIOUS PARTS *SEMI* A COIL AND CART ASSEMBLY
INCLUDING A CART *COMMA* HAVING A RECTANGULAR CROSS SECTION AND TUBULAR IN
CONSTRUCTION *COMMA* ADAPTED AT ONE END TO SUPPORT THE TRANSDUCER *STOP*)

Processing:

AN              : New instance word -- skip
ELECTROMAGNETIC : Mamette modifier; save and skip
LINEAR          : Mamette modifier; save and skip
ACTUATOR        : MP word -- mamette ACTUATOR#

New ACTUATOR# instance (&MEM31)
>>> Looking for mamette modified by CONFIGURATION/LINEAR from &MEM31 (ACTUATOR#)
<<< Found use of property CONFIGURATION = LINEAR [&MEM1 (ACTUATOR#)] (from P82)
Augmenting &MEM31 (ACTUATOR#) with feature: CONFIGURATION = LINEAR
>>> Looking for mamette modified by TYPE//PURPOSE/ELECTROMAGNETIC from &MEM31 (ACTUATOR#)
Augmenting &MEM31 (ACTUATOR#) with feature: TYPE//PURPOSE = ELECTROMAGNETIC

FOR (FOR1)      : Purpose indicator -- skip
POSITIONING     : Purpose word -- save and skip
A               : New instance word -- skip
TRANSDUCER      : MP word -- mamette TRANSDUCER#

New TRANSDUCER# instance (&MEM32)
Establishing P-GUIDES relation; SUBJECT: &MEM31 (ACTUATOR#);
  OBJECT: &MEM32 (TRANSDUCER#) [&REL20]

OVER            : Relation word -- save and skip
LOCATIONS       : MP word -- mamette LOCATION#

New LOCATION# instance (&MEM33)
>>> Refining R-ABOVE OBJECT from &MEM32 (TRANSDUCER#) &MEM31 (ACTUATOR#)
<<< Found R-BETWEEN relation(s) between &MEM17 (LOCATION#)
      and &MEM16 (TRANSDUCER#) (from P94)
Unable to select OBJECT -- using most recent
Establishing R-ABOVE relation; OBJECT: &MEM32 (TRANSDUCER#);
  SUBJECT: &MEM33 (LOCATION#) [&REL21]

ON (ON2)        : Part of indicator

Assuming &MEM33 (LOCATION#) or &MEM32 (TRANSDUCER#) or &MEM31 (ACTUATOR#)
 is part of the following

A               : New instance word -- skip
ROTATING        : Purpose word within NP; save and skip
MAGNETIC        : Mamette modifier; save and skip
RECORDING       : Purpose word within NP; save and skip
DISK            : MP word -- mamette DISK#

New DISK# instance (&MEM34)
Establishing P-WRITES relation; OBJECT: &MEM34 (DISK#) [&REL22]
>>> Looking for mamette modified by TYPE//PURPOSE/MAGNETIC from &MEM34 (DISK#)
<<< Found use of property TYPE//PURPOSE = MAGNETIC [&MEM3 (DISK#)] (from P82)
Augmenting &MEM34 (DISK#) with feature: TYPE//PURPOSE = MAGNETIC
Establishing P-ROTATES relation; OBJECT: &MEM34 (DISK#) [&REL23]
>>> Selecting comp for &MEM34 (DISK#) from among &MEM33 (LOCATION#)
      &MEM32 (TRANSDUCER#) &MEM31 (ACTUATOR#)
<<< Found HAS-PART relation(s) between &MEM12 (DISK#) and &MEM17 (LOCATION#) (from P94)
<<< Found R-CLOSE-TO relation(s) between &MEM3 (DISK#) and &MEM1 (ACTUATOR#) (from P82)
Assuming &MEM33 (LOCATION#) is part of &MEM34 (DISK#)
```

Figure 10: RESEARCHER processing the first part of P58

relation similar to the one in P94 in its representation of this example. RESEARCHER also finds a

physical relation between **actuator#** and **disk#** (from P82), but that example was not relevant to the ambiguity being resolved.

Had RESEARCHER not found a relevant example, it would have resorted to its set of heuristics and gotten the same result. The relevant rule states that "virtual" objects, such as **location#**, which refer to implicit parts of objects, are more likely to be parts of solid objects (such as **disk#**) than are complex objects (such as **actuator#**).[5] This sort of processing is related to the use of semantic properties of words for disambiguation, and is integrated nicely with memory search. However, we wish to avoid adding too many *ad hoc* rules of this sort and feel that the memory-based result is more pleasing.

Further use of memory by RESEARCHER occurs in the processing of the next section of P58, shown in Figure 11.

The first noun group processed in Figure 11, "an actuator housing", includes a noun-noun construction requiring memory access. RESEARCHER must determine the relationship between the two objects described, **actuator#** and **enclosure#** ("housing"). There are no syntactic clues or semantic clues as to the relation. So, RESEARCHER goes to memory and finds the assembly/component and unknown purpose relations that existed in memory for another example of these concepts (P82). It assumes that these relations hold for the new example. Had there been a more complex construction, say noun-noun-noun, RESEARCHER would have used this same information in memory to determine which objects were related to each other.

The assembly/component relation from P82 is used to resolve another textual ambiguity. RESEARCHER must determine whether **actuator#** or **disk#** is comprised of the "actuator housing" (and other parts). Again, the existing relation in memory resolves this ambiguity, and determines that the housing is part of the actuator. Since this relationship has already been established (while analyzing the noun group) processing simply moves on. As mentioned when we showed how RESEARCHER processed P58 with no help from memory, this turns out to be the key ambiguity to be resolved in understanding P58.

The remainder of Figure 11 shows more examples of "identifying pieces", "putting the pieces

[5]Other such virtual objects are **side#**, **top#**, etc.

```
COMPRISING      : Parts of &MEM34 (DISK#) or &MEM33 (LOCATION#)
                    or &MEM32 (TRANSDUCER#) or &MEM31 (ACTUATOR#) to follow
AN              : New instance word -- skip
ACTUATOR        : Memette within NP; save and skip
HOUSING         : MP word -- memette ENCLOSURE#

New ENCLOSURE# instance (&MEM35)
>>> Looking for relation between ACTUATOR# and &MEM35 (ENCLOSURE#)
<<< Found HAS-PART UNKNOWN-PURP-REL relation(s) between &MEM1 (ACTUATOR#)
      and &MEM6 (ENCLOSURE#) {from P82}
Assuming &MEM35 (ENCLOSURE#) is part of &MEM31 (ACTUATOR#)
Establishing UNKNOWN-PURP-REL relation; SUBJECT: &MEM31 (ACTUATOR#);
  OBJECT: &MEM35 (ENCLOSURE#) [&REL24]
>>> Selecting assy for &MEM35 (ENCLOSURE#) from among &MEM34 (DISK#)
      &MEM33 (LOCATION#) &MEM32 (TRANSDUCER#) &MEM31 (ACTUATOR#)
<<< Found IS-PART-OF UNKNOWN-PURP-REL relation(s) between &MEM6 (ENCLOSURE#)
      and &MEM1 (ACTUATOR#) {from P82}
&MEM35 (ENCLOSURE#) is already known to be a part of &MEM31 (ACTUATOR#)

USED AS         : Phrase
-> USED-AS      : Purpose word -- save and skip
A               : New instance word -- skip
STATIONARY      : Memette modifier; save and skip
BASE            : MP word -- memette BASE#

New BASE# instance (&MEM36)
>>> Looking for memette modified by MOBILITY/NONE from &MEM36 (BASE#)
Augmenting &MEM36 (BASE#) with feature: MOBILITY = NONE
Assuming &MEM36 (BASE#) is part of &MEM31 (ACTUATOR#)
>>> Refining P-ACTS-AS SUBJECT from &MEM35 (ENCLOSURE#) &MEM31 (ACTUATOR#)
<<< Positive heuristic result ENCLOSURE# ACTUATOR# on NOT-IMPLICIT-PURPOSE
Establishing P-ACTS-AS relation; SUBJECT: &MEM35 (ENCLOSURE#);
  OBJECT: &MEM36 (BASE#) [&REL25]

FOR (FOR1)      : Purpose indicator -- skip
SUPPORTING      : Purpose word -- save and skip
VARIOUS         : Memette modifier; save and skip
PARTS           : MP word -- memette PART#

New PART# instance (&MEM37)
>>> Looking for memette modified by NUMBER/SOME from &MEM37 (PART#)
Augmenting &MEM37 (PART#) with feature: NUMBER = SOME
Assuming &MEM37 (PART#) is part of &MEM31 (ACTUATOR#)
>>> Refining P-SUPPORTS SUBJECT from &MEM36 (BASE#) &MEM35 (ENCLOSURE#)
<<< Positive heuristic result &MEM29 (BASE#) on IMPLICIT-PART-OF-UNITARY
<<< Positive heuristic result &MEM6 (ENCLOSURE#) on NOT-IMPLICIT-PURPOSE
Unable to select SUBJECT -- using most recent
Establishing P-SUPPORTS relation; SUBJECT: &MEM36 (BASE#);
  OBJECT: &MEM37 (PART#) [&REL26]
Establishing R-CONNECTED-TO relation; SUBJECT: &MEM36 (BASE#);
  OBJECT: &MEM37 (PART#) [&REL27]

*SEMI*          : Skip (SKIP)
```

Figure 11: RESEARCHER processing the second part of P58

together" and accessing memory to resolve ambiguity. No further relevant examples from memory are found, so RESEARCHER makes use of heuristics like the one mentioned above. In the case of determining whether the "actuator" or the "housing" is "used as a base" both satisfy the same heuristic, so the most recent object mentioned is used.

Figures 12 and 13 show further examples of RESEARCHER's processing as it completes the first

fragment of P58. Examples from both P94 and P137 are found to indicate the various relations within the "coil and cart assembly". (Actually, to do this example in a fully general fashion, we would also have to apply memory-based techniques to determine the scope of "and" as a connective.) This example illustrates clearly how RESEARCHER's memory search is semantic rather than lexical, since P94 referred to a "carriage" in a "device" and yet RESEARCHER can use this example to determine the relation between a "cart" and an "assembly". Figures 12 and 13 include cases of RESEARCHER finding examples from both P94 and P137 that serve to resolve the same ambiguity. A topic of further research is to determine what to do if relevant examples from memory contradict each other (see Section 4.4).

When RESEARCHER reaches the point in processing shown in Figure 13, it has built up the representation shown in Figure 14. (RESEARCHER handles the entire patent abstract, but the representation gets rather messy.) This representation captures all the information from P58 that is needed for the learning aspects of RESEARCHER.

6 Conclusion

As we have seen in this paper, memory application is an absolute necessity if we wish for our understanding systems to take advantage of the all information they possess to improve performance. However, we must delineate exactly how memory should be used, as illustrated in this paper. In the RESEARCHER framework, simple syntactic rules, driven by generic memory structures ("semantics"), limit the possible ways that a representation can be constructed, and searching detailed memory resolves ambiguities. This allows each phase of the processing to be relatively simple and lets the redundant nature of language help us obtain robust performance. While a different conceptual understanding scheme or a different domain might require other points of memory access, this general framework will still be appropriate.

Although we do feel that many levels of language comprehension can occur in parallel, as is probably the case in the human brain, we must resist the temptation to assume that such processing is hopelessly unstructured. The observation that the levels of the parallel processing only interact at specified points allows us to consider different levels separately. However, it is equally crucial that we not consider the different levels of processing in total isolation, as strong interaction is certainly necessary for optimal performance. Furthermore, access of long-term memory must be part of language processing to

```
A              : New instance word -- skip
COIL           : Memette within NP; save and skip
AND (AND2)     : Skip (SKIP)
CART           : Memette within NP; save and skip
ASSEMBLY       : MP word -- memette UNKNOWN-ASSEMBLY#

New UNKNOWN-ASSEMBLY# instance (&MEM38)
>>> Looking for relation between CARRIAGE# and &MEM38 (UNKNOWN-ASSEMBLY# -- 'ASSEMBLY')
<<< Found IS-PART-OF relation(s) between &MEM21 (CARRIAGE#)
      and &MEM20 (UNKNOWN-ASSEMBLY# -- 'ASSEMBLY') (from P137)
<<< Found IS-PART-OF relation(s) between &MEM15 (CARRIAGE#)
      and &MEM9 (UNKNOWN-ASSEMBLY# -- 'DEVICE') (from P94)
New CARRIAGE# instance (&MEM39)
Assuming &MEM39 (CARRIAGE#) is part of &MEM38 (UNKNOWN-ASSEMBLY# -- 'ASSEMBLY')
>>> Looking for relation between COIL# and one of &MEM39 (CARRIAGE#)
      &MEM38 (UNKNOWN-ASSEMBLY# -- 'ASSEMBLY')
<<< Found IS-PART-OF relation(s) between &MEM26 (COIL#)
      and &MEM20 (UNKNOWN-ASSEMBLY# -- 'ASSEMBLY') (from P137)
New COIL# instance (&MEM40)
Assuming &MEM40 (COIL#) is part of &MEM38 (UNKNOWN-ASSEMBLY# --'ASSEMBLY')
Assuming &MEM38 (UNKNOWN-ASSEMBLY# -- 'ASSEMBLY') is part of &MEM31 (ACTUATOR#)

INCLUDING      : Parts of &MEM38 (UNKNOWN-ASSEMBLY# -- 'ASSEMBLY')
                   or &MEM31 (ACTUATOR#) to follow
A              : New instance word -- skip
CART           : MP word -- memette CARRIAGE#

Reference for CARRIAGE#: &MEM39
>>> Selecting assy for &MEM39 (CARRIAGE#) from among
      &MEM38 (UNKNOWN-ASSEMBLY# -- 'ASSEMBLY') &MEM31 (ACTUATOR#)
<<< Found IS-PART-OF relation(s) between &MEM21 (CARRIAGE#)
      and &MEM20 (UNKNOWN-ASSEMBLY# -- 'ASSEMBLY') (from P137)
<<< Found IS-PART-OF relation(s) between &MEM15 (CARRIAGE#)
      and &MEM9 (UNKNOWN-ASSEMBLY# -- 'DEVICE') (from P94)
&MEM39 (CARRIAGE#) is already known to be a part of &MEM38
(UNKNOWN-ASSEMBLY# -- 'ASSEMBLY')

*COMMA*        : Skip (SKIP)
HAVING         : Parts of &MEM39 (CARRIAGE#) or &MEM38 (UNKNOWN-ASSEMBLY# -- 'ASSEMBLY')
                   or &MEM31 (ACTUATOR#) to follow
A              : New instance word -- skip
RECTANGULAR    : Memette modifier; save and skip
CROSS SECTION  : Phrase
-> CROSS-SECTION : MP word -- memette CROSS-SECTION#

New CROSS-SECTION# instance (&MEM41)
>>> Looking for memette modified by SHAPE/RECTANGULAR from &MEM41 (CROSS-SECTION#)
Augmenting &MEM41 (CROSS-SECTION# --'CROSS-SECTION') with feature: SHAPE = RECTANGULAR
>>> Selecting assy for &MEM41 (CROSS-SECTION# -- 'CROSS-SECTION')
      from among &MEM39 (CARRIAGE#) &MEM38 (UNKNOWN-ASSEMBLY# -- 'ASSEMBLY') &MEM31 (ACTUATOR#)
<<< Positive heuristic result UNKNOWN-ASSEMBLY# on GENERIC-ASSEMBLY
Assuming &MEM41 (CROSS-SECTION# -- 'CROSS-SECTION') is part of
      &MEM38 (UNKNOWN-ASSEMBLY# -- 'ASSEMBLY')
```

Figure 12: RESEARCHER processing more of P58

explain the robustness of human processing and to achieve robustness in computer understanding systems. We feel that our efforts to integrate memory access with conceptual analysis techniques in RESEARCHER are important step in the direction of truly robust understanding and in developing systems that get smarter as they read.

```
AND (AND2)     : Skip (SKIP)
TUBULAR        : Memette modifier; save and skip
IN CONSTRUCTION : Phrase
-> IN-CONSTRUCTION : Collecting modifiers

>>> Looking for memette modified by CONFIGURATION/CYLINDRICAL from &MEM31 (ACTUATOR#)
&MEM38 (UNKNOWN-ASSEMBLY# -- 'ASSEMBLY') &MEM41 (CROSS-SECTION# -- 'CROSS-SECTION')
Unable to determine -- using closest
Augmenting &MEM38 (UNKNOWN-ASSEMBLY# -- 'ASSEMBLY') with feature: CONFIGURATION = CYLINDRICAL

*COMMA*        : Skip (SKIP)
ADAPTED        : Purpose word -- save and skip
AT             : Relation word -- save and skip
ONE            : Memette modifier; save and skip
END            : MP word -- memette END#

New END# instance (&MEM42)
>>> Looking for memette modified by NUMBER/1 from &MEM42 (END#)
Augmenting &MEM42 (END#) with feature: NUMBER = 1
Assuming &MEM42 (END#) is part of &MEM38 (UNKNOWN-ASSEMBLY# -- 'ASSEMBLY')
Establishing P-MODIFIES relation; SUBJECT: &MEM42 (END#) [&REL28]
Establishing R-AT relation; SUBJECT: &MEM42 (END#) [&REL29]

TO SUPPORT     : Phrase
-> SUPPORTS    : Purpose word -- save and skip
THE            : Antecedent word -- skip
TRANSDUCER     : MP word -- memette TRANSDUCER#

Reference for TRANSDUCER#: &MEM32
>>> Refining P-SUPPORTS SUBJECT from &MEM42 (END#)
      &MEM38 (UNKNOWN-ASSEMBLY# -- 'ASSEMBLY') &MEM31 (ACTUATOR#)
<<< Found IS-PART-OF relation(s) between &MEM22 (TRANSDUCER#)
      and &MEM20 (UNKNOWN-ASSEMBLY# -- 'ASSEMBLY') (from P137)
<<< Found IS-PART-OF relation(s) between &MEM16 (TRANSDUCER#)
      and &MEM9 (UNKNOWN-ASSEMBLY# -- 'DEVICE') (from P94)
Unable to select SUBJECT -- using most recent
Establishing P-SUPPORTS relation; SUBJECT: &MEM42 (END#); OBJECT: &MEM32 (TRANSDUCER#) [&REL30]

*STOP*         : Break word -- skip
```

Figure 13: RESEARCHER processing the final part of P58

Text Representation:

```
              |---------XY-35                31 = TYPE/PURPOSE/ELECTROMAGNETIC
              |                                   CONFIGURATION/LINEAR
              |                                   ACTUATOR#
              |-------YZA1-36                35 = ENCLOSURE#
              |--------ZA1-37                36 = MOBILITY/NONE BASE#
              |               |------------39 37 = NUMBER/SOME PART#
---------TX-31|               |------------40 38 = UNKNOWN-ASSEMBLY#
              |------------38|------------41 39 = CARRIAGE#
                              |-----B1C1D1-42 40 = COIL#
                                             41 = SHAPE/RECTANGULAR
                                                  CROSS-SECTION#
                                             42 = NUMBER/1 END#

------------------TUD1-32 32 = TRANSDUCER#

-------------VW-34|--------------U-33 33 = NUMBER/>1 LOCATION#
                                    34 = TYPE/PURPOSE/MAGNETIC DISK#
```

A list of relations:

	Subject:	Relation:	Object:
[&REL20/T]	&MEM31 (ACTUATOR#)	{P-GUIDES}	&MEM32 (TRANSDUCER#)
[&REL21/U]	&MEM33 (LOCATION#)	{R-ABOVE}	&MEM32 (TRANSDUCER#)
[&REL22/V]		{P-WRITES}	&MEM34 (DISK#)
[&REL23/W]		{P-ROTATES}	&MEM34 (DISK#)
[&REL24/X]	&MEM31 (ACTUATOR#)	{UNKNOWN-PURP-REL}	&MEM35 (ENCLOSURE#)
[&REL25/Y]	&MEM35 (ENCLOSURE#)	{P-ACTS-AS}	&MEM36 (BASE#)
[&REL26/Z]	&MEM36 (BASE#)	{P-SUPPORTS}	&MEM37 (PART#)
[&REL27/A1]	&MEM36 (BASE#)	{R-CONNECTED-TO}	&MEM37 (PART#)
[&REL28/B1]	&MEM42 (END#)	{P-MODIFIES}	
[&REL29/C1]	&MEM42 (END#)	{R-AT}	
[&REL30/D1]	&MEM42 (END#)	{P-SUPPORTS}	&MEM32 (TRANSDUCER#)

Figure 14: RESEARCHER representation of P58 at this point

References

[1] Birnbaum, L. and Selfridge, M. Conceptual analysis of natural language. In R. C. Schank and C. K. Riesbeck, Ed., *Inside Computer Understanding*, Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1981, pp. 318 - 353.

[2] Bobrow, R. J. and Webber, B. L. PSI-KLONE - Parsing and semantic interpretation in the BBN Natural Language Understanding System. Proceedings of the CSCSI/CSEIO Annual Conference, 1980.

[3] Brachman, R. J. and Schmolze, J. G. "An overview of the KL-ONE representation system." *Cognitive Science 9*, 2, 1985, pp. 171 - 216.

[4] Carrithers, C. and Bever, T. G. "Eye-fixation patterns during reading confirm theories of language comprehension." *Cognitive Science 8*, 2, 1984, pp. 157 - 172.

[5] Charniak, E. "Passing markers: A theory of contextual influence in language comprehension." *Cognitive Science 7*, 3, 1983, pp. 171 - 190.

[6] Cullingford, R. Script application: Computer understanding of newspaper stories. Technical Report 116, Yale University Department of Computer Science, 1978.

[7] Davis, M. S. "That's interesting! Towards a phenomenology of sociology and a sociology of phenomenology." *Philosophy of the Social Sciences 1*, 1971, pp. 309 - 344.

[8] DeJong, G. F. "Prediction and substantiation: A new approach to natural language processing." *Cognitive Science 3*, 1979, pp. 251 - 273.

[9] Dyer, M. G. *In-depth understanding: A computer model of integrated processing for narrative comprehension.* MIT Press, Cambridge, MA, 1983.

[10] Finin, T. W. The interpretation of nominal compounds in discourse. Technical Report MS-CIS-82-3, Moore School of Engineering, University of Pennsylvania, 1982.

[11] Finin, T. W. and Webber, B. L. BUP - A bottom up parser. Technical Report MS-CIS-83-16, Moore School of Engineering, University of Pennsylvania, 1983.

[12] Fodor, J. A., Bever, T. G. and Garrett, M. F. *The Psychology of Language.* McGraw Hill, New York, 1974.

[13] Gershman, A. V. Analyzing English noun groups for their conceptual content. Technical Report 110, Yale University Department of Computer Science, 1977.

[14] Granger, R. H., Eislet, K. P. and Holbrook, J. K. The parallel organization of lexical, syntactic and pragmatic inference. Proceedings of the First Annual Workshop on Theoretical Issues in Conceptual Information Processing, Atlanta, Georgia, 1984.

[15] Greiner, R. and Genesereth, M. R. What's new? A semantic definition of novelty. Proceedings of the Eighth International Joint Conference on Artificial Intelligence, Karlsruhe, West Germany, 1983, pp. 450 - 454.

[16] Grice, H. P. Logic and conversation. In P. Cole and J. L. Morgan, Ed., *Syntax and Semantics: Speech Acts, Volume 3*, Academic Press, New York, 1975.

[17] Grosz, B. J. Representation and use of focus in a system for understanding dialogs. Proceedings of the Fifth International Joint Conference on Artificial Intelligence, International Joint Conference on Artificial Intelligence, Cambridge, MA, 1977.

[18] Harris, L. R. Natural language processing applied to data base query. Proceedings of the 1978 ACM Annual Conference, Association for Computer Machinery, Washington, D. C., 1978.

[19] Hayes, P. J. Semantic markers and selectional restrictions. In E. Charniak and Y. Wilks, Ed., *Computational Semantics*, North-Holland Publishing Company, Amsterdam, 1976, pp. 41 - 54.

[20] Hirst, G. *Semantic interpretation against ambiguity.* Ph.D. Thesis, Department of Computer Science, Brown University, 1983.

[21] Katz, J. and Fodor, J. A. "The structure of a semantic theory." *Language 39*, 1963, pp. 170 - 210.

[22] Lebowitz, M. Generalization and memory in an integrated understanding system. Technical Report 186, Yale University Department of Computer Science, New Haven, CT, 1980. PhD Thesis.

[23] Lebowitz, M. Cancelled due to lack of interest. Proceedings of the Seventh International Joint Conference on Artificial Intelligence, Vancouver, Canada, 1981, pp. 13 - 15.

[24] Lebowitz, M. Limited parallel parsing. Columbia University Department of Computer Science, 1982.

[25] Lebowitz, M. RESEARCHER: An overview. Proceedings of the Third National Conference on Artificial Intelligence, Washington, DC, 1983, pp. 232 - 235.

[26] Lebowitz, M. "Memory-based parsing." *Artificial Intelligence 21*, 4, 1983, pp. 363 - 404.

[27] Lebowitz, M. "Generalization from natural language text." *Cognitive Science 7*, 1, 1983, pp. 1 - 40.

[28] Lebowitz, M. "Conceptual processing when things go wrong." *Cognition and Brain Theory 7*, 3 & 4, 1984, pp. 375 - 398.

[29] Lebowitz, M. RESEARCHER: An experimental intelligent information system. Proceedings of the Ninth International Joint Conference on Artificial Intelligence, Los Angeles, 1985, pp. 858 - 862.

[30] Lebowitz, M. Concept learning in a rich input domain: Generalization-Based Memory. In Michalski, R. S., Carbonell, J. G. and Mitchell, T. M., Ed., *Machine Learning: An Artificial Intelligence Approach, Volume II*, Morgan Kaufmann, Los Altos, CA, 1986.

[31] Levi, J. N. *The Syntax and Semantics of Complex Nominals.* McGraw Hill, New York, 1978.

[32] Marslen-Wilson, W. D. "Sentence perception as an interactive parallel process." *Science 189*, 1975, pp. 226 - 228.

[33] McDonald, D. B. Compound: A program that understands noun compounds. Proceedings of the Seventh International Joint Conference on Artificial Intelligence, Vancouver, Canada, 1981, pp. 1061.

[34] Minsky, M. A framework for representing knowledge. In P. H. Winston, Ed., *The Psychology of Computer Vision*, McGraw-Hill, New York, 1975.

[35] Paris, C. L. Description strategies for naive and expert users. Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics, Chicago, 1985.

[36] Quillian, M. R. Semantic memory. In M. Minsky, Ed., *Semantic Information Processing*, MIT Press, Cambridge, MA, 1978.

[37] Riesbeck, C. K. and Martin, C. E. Direct memory access parsing. Technical Report 354, Yale University Department of Computer Science, 1985.

[38] Riesbeck, C. K. and Schank, R. C. Comprehension by computer: Expectation-based analysis of sentences in context. In W. J. M. Levelt and G. B. Flores d'Arcais, Ed., *Studies in the Perception of Language*, John Wiley and Sons, Chichester, England, 1976.

[39] Schank, R. C. "Language and memory." *Cognitive Science 4*, 3, 1980, pp. 243 - 284.

[40] Schank, R. C. *Dynamic Memory: A Theory of Reminding and Learning in Computers and People.* Cambridge University Press, New York, 1982.

[41] Schank, R. C. and Birnbaum L. Memory, meaning and syntax. In T. Bever, J. Carroll and L. Miller, Ed., *Talking Minds: The Study of Language in Cognitive Sciences*, MIT Press, Cambridge, MA, 1982. Also Yale Computer Science Technical Report 189.

[42] Schank, R. C., Lebowitz, M., and Birnbaum, L. "An integrated understander." *American Journal of Computational Linguistics 6*, 1, 1980, pp. 13 - 30.

[43] Sidner, C. L. *A Computational model of co-reference comprehension in English.* Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, 1979.

[44] Small, S. Word expert parsing: A theory of distributed word-based natural language understanding. Technical Report TR-954, University of Maryland, Department of Computer Science, 1980.

[45] Swinney, D. "Lexical access during sentence comprehension: (Re)consideration of context effects." *Journal of Verbal Learning and Verbal Behavior 18*, 1979, pp. 645 - 660.

[46] Swinney, D. "The process of language comprehension; an approach to examining issues in cognition and language." *Cognition 10*, 1981, pp. 307 - 312.

[47] Swinney, D. and Cutler, A. "The access and processing of idiomatic expressions." *Journal of Verbal Learning and Verbal Behavior 18*, 1979, pp. 523 - 534.

[48] Townsend, D. J. and Bever, T. G. "Natural units of representation interact during sentence comprehension." *Journal of Verbal Learning and Verbal Behavior 21*, 1982, pp. 688 - 703.

[49] Waltz, D. L. and Pollack, J. B. "Massively parallel parsing: A strongly interactive model of natural language interpretation." *Cognitive Science 9*, 1, 1985, pp. 51 - 74.

[50] Wasserman, K. *Unifying representation and generalization: Understanding hierarchically structured objects.* Ph.D. Thesis, Columbia University Department of Computer Science, 1985.

[51] Wasserman, K. and Lebowitz, M. "Representing complex physical objects." *Cognition and Brain Theory 6*, 3, 1983, pp. 333 - 352.

[52] Waterman, D. A. and Hayes-Roth, F. *Pattern-Directed Inference.* Academic Press, New York, 1978.

[53] Wilensky, R. Understanding goal-based stories. Technical Report 140, Yale University Department of Computer Science, 1978.

[54] Wilks, Y. "Making preferences more active." *Artificial Intelligence 11*, 1978, pp. 197 - 223.

[55] Wilks, Y., Huang, X. and Fass, D. Syntax, preference and right attachment. Proceedings of the Ninth International Joint Conference on Artificial Intelligence, Los Angeles, 1985, pp. 779 - 784.

[56] Winograd, T. *Understanding Natural Language.* Academic Press, New York, 1972.

[57] Woods, W. A. "Cascaded ATN grammars." *American Journal of Computational Linguistics 6*, 1, 1980.