Uniform Parsing and Inferencing for Learning*

Charles E. Martin and Christopher K. Riesbeck

Yale University
New Haven, CT 06520

Abstract

In previous papers we have argued for the complete integration of natural language understanding with the rest of the cognitive system. Given a set of richly indexed memory structures, we have claimed that parsing is a general memory search process guided by predictive patterns of lexical and conceptual items which are a part of those memory structures.

In this paper, we demonstrate that our architecture for language understanding is capable of implementing the memory search processes required to make complex inferences not directly associated with parsing. The uniform format of the knowledge representation and search process provide a foundation for learning research.

1 Introduction

Research at the Yale Economics Learning Project is aimed at modelling knowledge reorganization and learning as a reasoner goes from being novice to expert in its domain. [Riesbeck 1983] has argued for expert reasoning as the result of gradual changes to novice reasoning in response to self-acknowledged *failures* in novice reasoning. The original learning system parsed texts such as "high interest rates limit growth," "low growth raises prices," and "large budget deficits cause higher interest rates" into separate meaning representations which were then pieced together to derive new economic arguments [Riesbeck 1981].

We now believe that a much tighter connection must be made between natural language understanding and the rest of the cognitive system in order to make progress towards our goals for the learning project. The language understanding system must be able to take advantage of the knowledge present in memory to the same degree that any other memory process could, and other memory processes must be able to make full and immediate use of linguistic input without waiting for a final interpretation to be formed.

This is the reflection of a re-orientation of the learning project in a much more promising direction. The system begins with a richly-indexed episodic memory of various arguments, including information such as who gave the argument, which other arguments it supports or contradicts, and so on. Linguistic input is used by the system to recognize relevant prior arguments; differences between the input and prior memory structures give rise to *failures* in the recognition process, which are resolved by recognizing and applying *reconciliation strategies*.

The common threads of this architecture are 1) a uniform representation of domain knowledge, failure structures, and reconciliation strategies in the regular memory format and 2) a uniform view of memory processes, including language understanding, as search through a knowledge base controlled by the prior recognition of structures in that knowledge base.

*This report describes work done in the Department of Computer Science at Yale University. It was supported in part by the Air Force Office of Scientific Research under contract F49620-82-K-0010.

In previous papers ([Riesbeck and Martin 1985]), we have argued for an approach to parsing which conforms to this view. The parsing algorithm is a process of lexically-guided memory search in which predictive patterns of words and concepts guide a general memory search process to recognize relevant memory structures. We call this *direct memory access parsing* (DMAP). Our memory structures are frame-like objects called *Memory Organization Packets* (MOPs), organized by the standard part-whole packaging and class-subclass abstraction hierarchies [Schank 1982].

This approach is the reverse of that taken by past conceptual analyzers ([Riesbeck 1975] [Lebowitz 1980] [Dyer 1982] [Lytinen 1984]) that construct meaning representations from texts which may then be be connected to memory in a separate step; this is the "Build and Store" model of conceptual analysis. The proposed alternative is to find relevant structures in memory and record differences between the input and what exists already. We call this the "Recognize and Modify" model.

We are now turning our attention back to the original goals of the learning project. When failures occur in the understanding process, we wish to trigger inference processes to record those failures and to implement strategies for resolving the anomalies. In this paper, we describe how our previous approach to integrating parsing with memory extends naturally to handle these inference mechanisms: failure episodes and reconciliation strategies are represented in the regular memory format of domain knowledge, and we are excited that a single, uniform memory search process appears capable of handling both parsing and memory-based inference in such a knowledge base.

This paper examines the architecture we have evolved for our system. Section 2 reviews our original work on parsing, detailing the memory structures and the search process used for recognition. Section 3 explains how we have augmented this with failure and strategy structures to build new memory structures where necessary. Section 4 extends the failure and strategy concepts to handle inference which is only indirectly related to the parsing task.

2 Integrating Parsing with Memory

We integrate parsing knowledge into memory by attaching linguistic templates to memory structures in a manner reminiscent of the Teachable Language Comprehender [Quillian 1969]. These templates, called *concept sequences*, are patterns of words and concepts. For example, attached to the memory structure MILTON-FRIEDMAN is the concept sequence {Milton Friedman}, representing the linguistic phrase "Milton Friedman." Attached to MTRANS-EVENT, our primitive marker for communications events [Schank and Abelson 1977], is the concept sequence {actor says mobject}, representing

1. the identification of another memory structure which is indexed from MTRANS-EVENT through the packaging hierarchy via the actor role,

2. the linguistic item "says," and

3. the identification of another memory structure which is indexed from MTRANS-EVENT through the packaging hierarchy via the mobject role (representing the content of the communicated information).

Any memory structure can have one or more concept sequences; in addition, the abstraction hierarchy provides an inheritence mechanism through which any structure implicitly acquires the sequences attached at a more general level of abstraction.

The dictionary in DMAP, which we call the *concept lexicon*, is simply a set of pointers from words and concepts to the concept sequences they appear in. The concept sequences encode the lexical and syntactic knowledge of the parser. This is a generalization of the "phrasal lexicon" approach to language understanding [Becker 1975] that includes not only actual phrases, but more conceptual combinations as well. The primary task of concept sequences is to quickly connect standardized patterns of language use to general memory structures of the system. To this end, the DMAP model depends on the use of parallel activation and intersection to resolve the basic combinatorial explosion, as is presumed in a number of other recent models [Small et al. 1982] [Hahn and Reimer 1983] [Granger et al. 1984] [Waltz and Pollack 1984] [Charniak unpb].

In the process of recognizing conceptual elements of concept sequences, the parser will identify more specific structures than the general concept sequence refers to. The parser uses these specific structures to recognize episodes in memory which are 1) consistent with the general structures predicted by the concept sequence, and 2) capable of adequately packaging the other structures recognized by the input. Because the parsing process attempts to recognize the most specific memory structures available, exactly which memory structures the parser settles on depends on which ones are already in memory. Figure 1 depicts a simplified portion of the memory structures used to recognize the communicative act of the following text.

> *The New York Times*, August 4, 1983.
>
> Milton Friedman: Interest rates will rise as an inevitable consequence of the monetary explosion we've experienced over the past year.

If this claim of Friedman's has been seen before, then seeing it again, as originally stated, or paraphrased, will guide the parser to the previously built memory structure MF:MTRANS-EVENT.

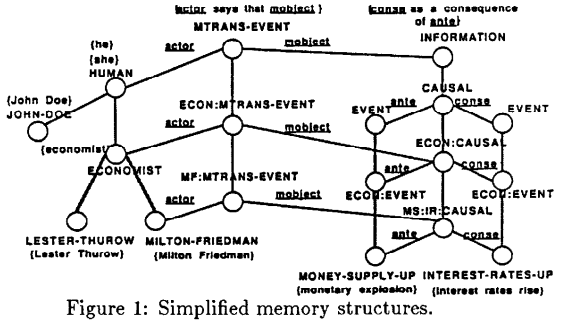

Figure 1: Simplified memory structures.

2.1 Marker passing

The parser uses a marker-passing architecture to identify relevant memory structures from the input text and the expectations in memory. Two kinds of markers are used in the system: *activation* markers, which capture information about the input text and the current selection of relevant memory structures, and *prediction* markers, which indicate which memory structures may be expected to become relevant.

DMAP is definitely not disambiguation with marker passing [Waltz and Pollack 1984]. Rather than using marker passing as an appendage to a standard parser for finding the (shortest, strongest, whatever) path between two nodes in memory, the structures found through marker passing *are* the most relevant ones in memory and comprise themselves the result of the parse.

The connectionist work is also currently focussing on the disambiguation problem [Cottrell 1984], though here it is intended that eventually all aspects of parsing will be included in the same spreading activation framework. The connectionist project is much more difficult, since they are deliberately limiting the allowable set of mechanisms. They do not have access to the kinds of structured markers we are quite willing to invoke.

2.2 Concept activation

Memory structures are activated by placing activation markers on them. Activation markers are created in two situations.

- *System input:* when an input word is read by the parser, an activation marker is created and placed on the associated lexical item in memory.
- *Concept sequence recognition:* when every element of a concept sequence has been activated, an activation marker is created and placed on the associated memory structure.

Activation markers are passed up the class-subclass abstraction hierarchy from their associated structures. This is a recursive process; all structures which receive an activation marker continue to pass it on to their own abstractions. When a memory structure receives an activation marker, that structure is said to have been *activated*; the activation marker contains a pointer to the originally activated structure.

For example, an activation marker associated with MONEY-SUPPLY-UP will be passed to ECON-EVENT, which in turn passes the marker to EVENT. All of these structures are activated, while the activation marker keeps a pointer to MONEY-SUPPLY-UP.

2.3 Concept prediction

Prediction markers represent concept sequences which are in the process of being recognized. Whenever a memory structure is activated, prediction markers are created for all the concept sequences indexed by that memory structure through the concept lexicon. A prediction marker captures the intuition of the "focus of attention" of the parser. A shift of attention corresponds to passing the prediction marker to a new location in memory; this takes place in response to concept activation. When a memory structure is activated which intersects the current focus of a prediction via some packaging relationship, the prediction is altered by two concurrent processes.

- *Concept refinement.* Since the activation will generally supply more specific information about the current input than the prediction takes into account, the prediction marker can be passed down the abstraction hierarchy to a more specialized memory structure which better packages the activation.
- *Sequence advancement.* Intersection of an activation marker will complete the current element of the prediction marker's concept sequence. If the sequence has not yet been completed, the prediction marker can be passed across the abstraction hierarchy to focus on the next element of the sequence.

3 Simple Memory Modification

Of course, it is not enough to recognize structures in memory; the parser must also be able to record "where it has been." For example, if MS:IR:CAUSAL were not contained in the memory of Figure 1, then

the parser would identify the more general ECON:CAUSAL. In this case, the parser can't find a structure which is specific to the activated memory structures it knows about, yet it has identified some general structures which serve to classify the input. We call this situation a *specialization failure*, and there exist structures in memory which serve to index such situations. In turn, these structures index *reconciliation strategy* memory structures which can reconcile the anomalies.

In this section, we describe how the most general of failure and reconciliation structures are recognized and activated. It is at this most general level that new memory structures are built; Section 4 describes how more specific failures cause the recognition process to search for reconciliations which may result in inference. Ultimately, all such search processes "bottom out" at the most general level of specialization failure, causing new structures to be created.

3.1 Recognizing failures

When the normal recognition process identifies a structure which is not suitably specialized, that process spawns a recognition process which is predictive of a specialization failure structure; although it is handled identically to the normal recognition process, it is initiated internally by the parser and not by a concept sequence. This is the only exception to the general recognition algorithm.

Specialization failure structures, like other memory structures, are organized by part-whole and class-subclass relationships. The most general specialization failure structure is MISSING-SPECIALIZATION. In the above example, if MS:IR:CAUSAL were missing from memory then an instance of MISSING-SPECIALIZATION would be recognized which packaged ECON:CAUSAL and the MONEY-SUPPLY-UP and INTEREST-RATES-UP activations.

3.2 Recognizing strategies

Reconciliation strategies are similar in spirit to both the Exception MOPs proposed in [Riesbeck 1981] and the explanation patterns (XPs) of [Schank unpb]. Reconciliations are also memory structures; they package a failure structure and other memory structures that "explain away" the failure. By "explain away" we mean that if the memory had contained the explanatory structures in the first place, the recognition process would not have arrived at a failure structure. A reconciliation is recognized by the system through the normal process of concept sequence completion; the most general reconciliation structure is ROTE-MEMORY.

ROTE-MEMORY simply adds new memory structures at the appropriate point to resolve a MISSING-SPECIALIZATION. Since MISSING-SPECIALIZATION is packaged by ROTE-MEMORY via the failure relationship, recognition of the failure structure leads to recognition of the strategy, and ROTE-MEMORY builds a new memory structure. In the above example, ROTE-MEMORY would create a new memory structure which packaged INTEREST-RATES-UP and MONEY-SUPPLY-UP underneath ECON:CAUSAL in the abstraction hierarchy.

3.3 Invoking ROTE-MEMORY

ROTE-MEMORY is invoked only in the simple situation where you know things of a certain type can occur, and one of them does. The input matches completely a general pattern and there is no more specific version of the pattern to compare the input with. ROTE-MEMORY creates new specializations of existing structures to package specific items.

It is important to note that ROTE-MEMORY will also be invoked to create specific sub-structures for an identified memory structure. For example, if we have identified a generalized "restaurant" MOP [Schank 1982] from the input, ROTE-MEMORY fills out the unspecified scenes according to the specific information available. The distinction between these two methods of invocation is only one of interpretation; in the implementation, an attempt is made to recognize sub-structures via the normal algorithm, which may or may not end up with the invocation of ROTE-MEMORY to create a new memory structure.

4 Failure-Driven Inferencing

Consider again the memory structures depicted in Figure 1. Given an input such as "John Doe blames the large increase in the money supply for the rise in interest rates," what structures should be recognized? When this is parsed, the parser is unable to specialize from ECON:MTRANS-EVENT to MF:MTRANS-EVENT because the more specific structure only partially matches the input—the actor of MF:MTRANS-EVENT does not match the actor of the input. This state of the parser is similar to that described above, with the exception that some prior knowledge structure has been recognized but deemed over-specialized due to the actor mismatch.

4.1 Failure and reconciliation structures

The additional information in this example serves to locate a more specific failure structure than MISSING-SPECIALIZATION. In this case, the failure structure identified is ACTOR:EXCEPTION. This structure packages: the *new package* that couldn't be specialized (John Doe's argument); the *new part* contained in the new package (John Doe); the *old package* (Milton Friedman's argument); and the *old part* (Milton Friedman).

The general situation of two people saying the same thing can be explained in many ways; since the parser attempts to recognize the most specific relevant structure in memory, it prefers to try domain-specific before more general strategies. A routine domain-specific explanation for why two economists say the same thing is "they belong to the same economic camp." This strategy for ACTOR:EXCEPTION is CREATE-CAMP; it packages

- the ACTOR:EXCEPTION *failure structure*,
- the *economic camp* which the actors belong to, and
- the *camp argument* which both arguments instantiate.

Figure 2 presents the actual definitions of these structures. Note the constraints placed on the sub-structures of CREATE-CAMP which reflect their mutual dependencies: the camp-mtrans structure is a specialization of ECON:MTRANS-EVENT whose actor is the camp of the strategy and which is in turn a generalization (isa-) of the old-package and new-package of the failure of the strategy.

```
(def actor:exception
    (isa: missing-specialization)
    (new-package (econ:mtrans-event))
    (old-package (econ:mtrans-event))
    (new-part (economist))
    (old-part (economist)))

(def create-camp
    (isa: reconciliation)
    (failure
        (actor-exception
            (new-package ?a) (old-package ?b)
            (new-part ?c)    (old-part ?d)))
    (camp (economist (isa- ?c ?d)))
    (camp-mtrans
        (econ:mtrans-event
            (actor (camp)) (isa- ?a ?b))))
```

Figure 2: Failure and strategy memory structures.

4.2 An example of failure-directed inference

With the simplified memory defined so far (the structures of Figures 1 and 2), we can follow the parse of "John Doe says that interest rates rise as a consequence of the monetary explosion." The recognition algorithm described in Section 2 is sufficient to identify ECON:MTRANS-EVENT, which is not specific to the active JOHN-DOE and MS:IR:CAUSAL structures. The specialization failure spawns a recognition process which identifies MISSING-SPECIALIZATION, since no other information is available to locate more specific structures. ROTE-MEMORY constructs JD:MTRANS-EVENT to package the input and be connected to memory as a specialization of ECON:MTRANS-EVENT.

Simultaneous with the above identification of the general ECON:MTRANS-EVENT was the recognition of the inapplicability of MF:MTRANS-EVENT due to over-specialization. With the activation of JD:MTRANS-EVENT, the structure of memory appears as depicted in Figure 3. This figure depicts the failure and strategy structures which will be relevant. (The packaging relationships from the ACTOR:EXCEPTION failure structure have been omitted for clarity.)

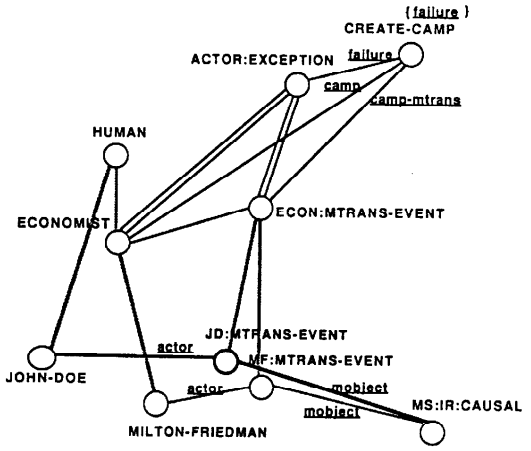

Figure 3: Failure and strategy structures.

Activation of JD:MTRANS-EVENT provides the extra information needed for the failure recognition process to identify ACTOR:EXCEPTION. Since this general failure structure does not directly package the active JD:MTRANS-EVENT and MF:MTRANS-EVENT structures, another recognition failure process is spawned. This identifies MISSING-SPECIALIZATION, and ROTE-MEMORY builds ACTOR:EXCEPTION-1. Note that the normal recognition process works on these failure structures in exactly the way that it works on "domain" memory structures.

The activation of ACTOR:EXCEPTION-1 causes the recognition process to recognize CREATE-CAMP. Once again, a MISSING-SPECIALIZATION is recognized, and CREATE-CAMP-1 is built by ROTE-MEMORY. At this point, the memory appears as depicted in Figure 4. (The packaging links at the general level of Figure 3 have been omitted for clarity.)

4.3 The result of parsing

At the conclusion of this example, the parser has built two memory structures which are not directly related to the input: ECON:CAMP-1 and CAMP-1:MTRANS-EVENT. These were built when the parser recognized two instances of MISSING-SPECIALIZATION while recognizing sub-structures of CREATE-CAMP-1. These new structures serve to better organize memory so that the same text will not create a failure if seen again; prior memory structures such as MF:MTRANS-EVENT have been automatically re-indexed in the correct relationships with the new structures.

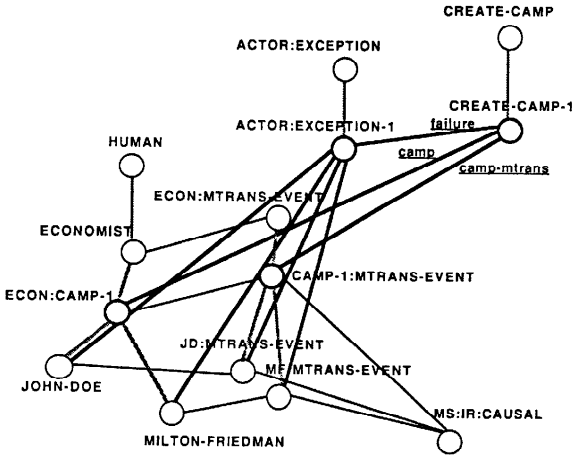

Figure 4: Instantiating a reconciliation structure.

A topic of future research is how the system might learn specific concept sequences to identify ECON:CAMP-1; e.g., that the structure refers to monetarists, with CAMP-1:MTRANS-EVENT referring to arguments commonly held by monetarists.

5 The Economic Learning Project

The previous section outlined an example in which the parser's inference revolves around its knowledge of argumentation and argument advocacy in the economics domain. The goal of the learning project is to model the reorganization and learning of knowledge as a reasoner progresses from novice to expert understanding of its domain. To this end, the system needs to have declarative representations of inference rules used in expert reasoning.

A common form of inference required to understand economic arguments is the construction of causal chains from individual causal structures. Consider the following expert text.

> Lester C. Thurow, *Newsweek*, September 21, 1983:
>
> With the resulting structure of taxes and expenditures, the President is not going to be balancing the Federal budget in 1984 or any other year. With high growth choked off by high interest rates, budget deficits are going to be bigger, not smaller. The result: more demands for credit and higher interest rates.

This is a rather complex argument, involving an implicit feedback loop and causal chain through interest rates, investment, business growth, tax revenues, and the deficit. Consider the phrase "with high growth choked off by high interest rates." The system recognizes this as an instance of ECON:CAUSAL, but does not recognize this particular example. The inference rule required is familiar:

> IF x_0 causes x_1, x_1 causes x_2, ..., and x_{n-1} causes x_n
> THEN x_0 causes x_n.

This unlimited chaining has been broken down into a two-step chaining structure in the implementation. The failure, strategy, and auxilliary structures are shown in Figure 5.

In this example, "high growth choked off by high interest rates" causes the parser to recognize CAUSAL-1, which invokes an ECON-CAUSAL:CONSEQUENT-EXCEPTION failure. The strategy indexed by this failure is USE:CAUSAL-CHAIN:FORWARD, which supports a causal argument by a causal chain. The binding constraints force the argument to be that presented in the text, while the causal chain begins with CAUSAL-1 and searches for a causal argument connecting this to the goal state. In this case, the system will find CAUSAL-2.

```
(def econ-causal:consequent-exception
    (isa: missing-specialization)
    (new-package (econ:causal))
    (old-package (econ:causal))
    (new-part (econ:event))
    (old-part (econ:event)))

(def use:causal-chain:forward
    (isa: use:causal-chain)
    (failure
        (econ-causal:consequent-exception
            (new-package ?a) (old-package ?b)
            (new-part ?c)    (old-part ?d)))
    (argument ?a)
    (support
        (causal-chain
            (first ?b)
            (second (ante ?d) (cnsq ?c)))))

(def causal-chain
    (first (econ:causal))
    (second (econ:causal)))

(def causal-1                    (def causal-2
    (isa: econ:causal)               (isa: econ:causal)
    (ante (high-interest-rates))     (ante (low-investment))
    (cnsq (low-investment)))         (cnsq (low-growth)))
```

Figure 5: Using the USE:CAUSAL-CHAIN heuristic.

At the conclusion of this example, the parser has built a new causal structure which is supported by a causal chain. This support structure records the use of USE:CAUSAL-CHAIN:FORWARD strategy, and the constructed chain can now be recognized in subsequent parsing without repeating the original memory search.

The identification of specific repair strategies and indexing them under the specific circumstances of their recognition is reminiscent of the work on learning to anticipate and avoid planning failures in [Hammond 1986]. Further research in the Economic Learning Project will require representing complex domain objects, including goals, plans, events, and argument structures ([Flowers et al. 1982], [Alvarado 1985]), and the inference rules used to understand and connect objects in memory. The inference rules of our system are represented declaratively and are processed identically to other memory objects; we think this is an essential step in building a learning system.

6 Conclusions

At this point we have demonstrated that our representation and process model is capable of handling 1) phrase-oriented parsing, and 2) automatic instantiation of inferential structures. What is particularly pleasing about this architecture is that these explanatory inference mechanisms are triggered when required by the current state of memory, and not through the artificial intercession of specific control procedures.

For example, the CREATE-CAMP reconciliation may or may not be applicable to a given ACTOR:EXCEPTION failure depending upon what the system already knows, i.e., what other memory structures it has that package the two arguments. Suppose that the concept of a monetarist is already represented in memory and the parser reads "Lester Thurow blames the rise in interest rates on the increased money supply." If we know enough about Lester Thurow to know that he has often made arguments against the monetarist position, then the failure is more specific than ACTOR:EXCEPTION and CREATE-CAMP will not be the appropriate resolution.

Instead, we want to locate reconciliations which capture the explanation that "Thurow is leaning towards monetarism," "current economic conditions make the monetarist position generally acceptable," and (at the general level of MTRANS-EVENT specialization failures) "Thurow is lying," among others. The point is that more specific information available in memory guides the search process to appropriate failure and reconciliation structures.

Acknowledgements

The work described here is based on the joint efforts of the Direct Memory Access Parsing project, which consists of Charles E. Martin, Monique Barbançon, and Michael Factor.

References

Alvarado, S.J., Dyer, M.G, and Flowers, M. (1985). Memory Representation and Retrieval for Editorial Comprehension. In *Proceedings of the Seventh Annual Conference of the Cognitive Science Society.* Irvine, CA.

Becker, J.D. (1975). The phrasal lexicon. In *Theoretical Issues in Natural Language Processing.* Cambridge, MA.

Charniak, E. (Unpublished). *A Single-Semantic-Process Theory of Parsing.*

Cottrell, G.W. (1984). A model of lexical access of ambiguous words. In *Proceedings of the AAAI-84.* Austin, Texas.

Dyer, M.G. (1982, May). *In-Depth Understanding: A Computer Model of Integrated Processing for Narrative Comprehension.* Technical Report 219 Yale University Department of Computer Science.

Flowers, M., McGuire, R., and Birnbaum, L. (1982). Adversary Arguments and the Logic of Personal Attacks. In W.G. Lehnert and M.G. Ringle (Eds.), *Strategies for Natural Language Understanding.* Lawrence Erlbaum Associates.

Granger, R.H., Eisclt, K.P., and Holbrook, J.K. (1984). The parallel organization of lexical, syntactic, and pragmatic inference processes. In *Proceedings of the First Annual Workshop on Theoretical Issues in Conceptual Information Processing.* Atlanta, GA.

Hahn, U. and Reimer U. (1983, November). *Word expert parsing: An approach to text parsing with a distributed lexical grammar.* Bericht TOPIC 6/83. Universitat Konstanz, Konstanz, West Germany.

Hammond, K.J. (1986). *Case-based Planning: An integrated theory of planning, learning and memory.* Ph.D. Thesis, Yale University. Forthcoming.

Lebowitz, M. (1980, October). *Generalization and Memory in an Integrated Understanding System.* Ph.D. Thesis, Yale University. Research Report #186.

Lytinen, L. (1984, November). *The Organization of Knowledge in a Multilingual, Integrated Parser.* Ph.D. Thesis, Yale University. Research Report #340.

Quillian, M.R. (1969). The Teachable Language Comprehender: A Simulation Program and Theory of Language. *Communications of the ACM, 12* (8).

Riesbeck, C.K. and Martin, C.E. (1985). *Direct Memory Access Parsing.* YALEU/DCS/RR 354. Yale University,

Riesbeck, C.K. (1975). Conceptual Analysis. In Schank, R.C. (Ed.), *Conceptual Information Processing.* Amsterdam: North Holland/American Elsevier.

Riesbeck, C.K. (1981). Failure-driven Reminding for Incremental Learning. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence.* Vancouver, B.C..

Riesbeck, C.K. (1983). Some problems for conceptual analyzers. In Sparck Jones, K. and Wilks, Y. (Eds.), *Automatic Natural Language Parsing.* Chichester: Ellis Horwood Limited.

Schank, R.C. and Abelson, R.P. (1977). *Scripts, plans, goals, and understanding.* Lawrence Erlbaum Associates.

Schank, R.C. (1982). *Dynamic Memory: A Theory of Learning in Computers and People.* Cambridge University Press.

Schank, R.C. (Unpublished). *Explanation Patterns.*

Small, S., Cottrell, G. and Shastri, L. (1982). Toward Connectionist Parsing. In *Proceedings of the AAAI-82.* Pittsburgh, PA..

Waltz, D.L. and Pollack, J.B. (1984). Phenomenologically plausible parsing. In *Proceedings of the AAAI-84.* Austin, Texas.