Taken from Advances in Cognitive Science (1986)

8

From Conceptual Analyzer to Direct Memory Access Parsing: An Overview

Christopher K. Riesbeck

INTRODUCTION—A REVISIONIST HISTORY OF PARSING

There is, I believe, a revolution happening in natural language processing (NLP) system development in Artificial Intelligence (AI). The revolution is a paradigm shift in our view of what mechanical language understanding is all about. This shift is independent of the syntax versus semantics controversy, but I believe it will lead eventually to models of parsing where the question "Who's in charge: syntax or semantics?" will be moot.

What is this new view of parsing? It is this: a parser is a memory search process, pure and simple. It differs from other such processes only in its emphasis on linguistic cues. The purpose of a parser is not to construct an interpretation for a text, but to locate those existing memory structures to which the text is referring. I call this "Direct Memory Access Parsing" or DMAP.

Although there is a constructive aspect to parsing, namely the "remembering" of the text and the references it makes, this construction of memory

structures is neither unique to parsing (we are always remembering uses of memory — that's why memory is dynamic), nor is it limited to those structures that have typically been identified with "the meaning of a sentence". For example, we remember veiled implications, tones of voice, initial misunderstandings, and so on. The only thing that separates these items from more standard ideas of meaning is whether, when we remember them, we include the memory structure for "speaker intended this."

For example, our misunderstanding of a text may seem like it can't possibly be the meaning of that text, but what about jokes? Suppose one of our parsers only remembered what texts were about and it heard this story:

> I just got back from a hunting expedition. The first night I shot two bucks.
> It was all the money I had.

When asked, our parser would paraphrase this as:

> He just got back from a hunting expedition. The first night he spent two dollars. It was all the money he had.

We'd certainly consider a human who did this unintelligent!

In short, in direct memory access parsing, the traditional notion of parsing as "constructing an interpretation" is replaced with the more general, non-parser-specific process of "classifying and remembering an episode," i.e. tracing memory use. What sets parsing apart from other processes is not the construction of interpretations, but the use of peculiarly linguistic items to direct the use of memory structures.

Riesbeck and Martin (1985) describe a particular implementation of a direct memory access parser. In this chapter, I'd like to trace the origins of this view of parsing, and describe the current state of the art. The reader should be warned that I am interpreting modern research in much the same way that a literary critic interprets novels. Do not assume that the researchers involved necessarily agree with the issues and imports I attribute to their systems.

CONCEPTUAL ANALYSIS

Until recently, direct memory access parsing was not possible because there weren't any suitable models of long-term memory to access. Instead, there was "conceptual analysis." Spinoza (Schank *et al.*, 1970) and the MARGIE parser (Riesbeck, 1975) were the first in a series of parsers that attempted to go directly from sentential input to conceptual representations, without constructing an intermediate syntactic description of the sentence.

For example, a key difference between the MARGIE parser and previous systems was the way it treated the following examples:

John gave Mary book.
John gave Mary a kiss.
John kissed Mary.

"John gave Mary a book" was parsed as "John transferred a book from John to Mary." "John gave Mary a kiss," however, despite its syntactic similarity, was parsed as "John pressed his lips against Mary." This result was identical to the parse produced for "John kissed Mary," and was the representation needed by the MARGIE inference module. The theoretical demands of the Conceptual Dependency representation scheme, and the needs of the inference and generation modules, distinguished the development of the MARGIE parser from systems that were being created to process English, but which had no general follow-on system to talk to.

Even at that time, the separation of the MARGIE parser from the MARGIE inference module was a matter of convenience, not theory, since the MARGIE parser often had to make inferences during parsing. The rule of thumb for the division of labor was this: any inference rule that required knowledge of English was the responsibility of the parser. The inference module should be unaffected if the English parser were replaced by a Chinese parser. Thus, for example, it was up to the inference module to determine that "John picked up a stick and hit Mary," implied that John hit Mary with the stick, but it was up to the parser to decide that "John had an apple" probably meant that "John ate an apple".

The MARGIE parser was an example of an expectation-driven parser, using "if–then" rules, called *requests,* to encode conceptual and syntactic parsing knowledge. For example, "give" had requests that said

- The subject of the sentence is transferring something to someone.
- If a physical object follows, then that is the object being transferred.
- If a human follows, then that human is the recipient of the transfer.
- If an action follows, then the subject of the sentence is doing that action to the recipient.

Later conceptual analyzers were ELI (Riesbeck & Schank, 1976), ELI-2 (Gershman, 1979), and CA (Birnbaum & Selfridge, 1981). They extended and changed the basic approach in many ways, including better control structures, standardized request formats, and so on. They maintained the basic ideas however. **The parsers always produced a meaning representation, not a syntactic structure, by applying requests attached to words read left to right in a sentence.** The meaning representation was then passed to inference modules, for language-independent processing.

The goal of the parser, as defined by these systems, was this: to get from the words in a sentence as directly as possible to the meaning of the sentence, with **"meaning" defined as "whatever the inference processes need."** The

request format helped to achieve this goal by allowing processing rules to be written that built and used conceptual structures as easily as other systems built and used syntactic structures. Fig. 1 is a simple block diagram for these conceptual analyzers.

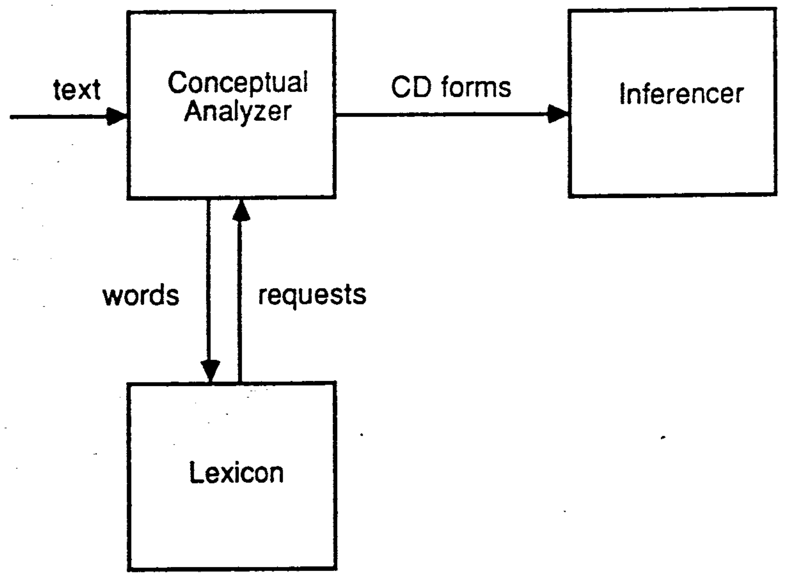

Fig. 1 — Early conceptual analyzers.

MEMORY-BASED CONCEPTUAL ANALYSIS

MARGIE, SAM (Schank & Abelson, 1977), PAM (Wilensky, 1978), FRUMP (DeJong, 1979), and POLITICS (Carbonell, 1979) were all knowledge-applying programs. **That is, they understood input texts by applying a fixed set of knowledge structures. SAM knew about standardized event sequences, such as going to a restaurant or a newspaper account of a car accident,** while PAM and POLITICS knew how to understood stories about goals and plans, such as how to get money by asking someone or robbing a bank.

In the early 1980s, our view of inference processes changed. People don't just have a *knowledge base*, they have a *memory* (Schank, 1982). We mean two things by this. First, what people know is the result of experience. Often, what you know about something is very intimately connected to your experiences with it. For example, when I think about hammers, I think about particular hammers I have owned.

Second, what people know is *dynamic,* i.e. it changes with use, whereas a knowledge structure is static, like the information in an encyclopedia. You can use a knowledge structure — or have problems using it — as many times as you want, and it will remain unchanged. Experiential knowledge, however, changes as the set of experiences changes. Barring long-term forgetting, I can't read the same story twice and understand it exactly the same way both times. The second time I read it, I immediately say, "Hey, I've read this before." For example, one restaurant story SAM *didn't* handle was this:

> A guy walks into a restaurant. He asked the waiter if they served crabs there ...

At this point in the processing, I wanted SAM to respond "I've heard that one before," but, unfortunately, SAM had knowledge, rather than memory, structures.

We began encoding knowledge in our programs in memory organization packets (MOPs), which restructured and reorganized knowledge into more "bite-size" chunks, amenable to recording experience and forming generalizations. Several parsing programs were developed using MOPs, namely IPP (Lebowitz, 1980), BORIS (Dyer, 1983), and MOPTRANS (Lytinen, 1984). IPP and BORIS built MOPs and stored them in memory. (MOPTRANS, a machine translation system, had a static memory.)

All three parsers defined the goal of parsing as: find and instantiate (make a copy of) applicable memory structures. Once a basic MOP frame was chosen, requests filled in the slots. These requests were often attached to the MOP structures, rather than to individual words. Since MOPs usually encoded fairly general knowledge, they had to be specialized to apply to the situation at hand. In MOPTRANS, for example, "Police captured ..." would initially be interpreted as a generic GET-CONTROL MOP, but this interpretation would be refined to ARREST because the polic were doing the capturing. In IPP, a similar refinement process would occur in sentences such as "Terrorists holding machine guns sprayed ..." where the generic "spray" would be understood as "shoot" in this context. In BORIS, requests made explicit calls to the memory, e.g. a request might say "If a specific MOP describing event already exists in memory, use it, otherwise add this event to memory."

Fig. 2 shows the basic structure of these memory-based parsers.

IPP, BORIS, and MOPTRANS are examples of what I call "build and store" parsers. There is a separate parsing process in each case that is responsible for building a memory structure and passing it on to a memory-storage facility. Like the earlier conceptual analyzers, these parsers have separately organized lexicons which contain all of the system's language-processing information. This is unlike syntactically based systems, which separate the rule base from the lexicon. The memory-based systems differ from their predecessors in that they construct long-term memory structures, rather than conceptual structures.

For example, given the sentence

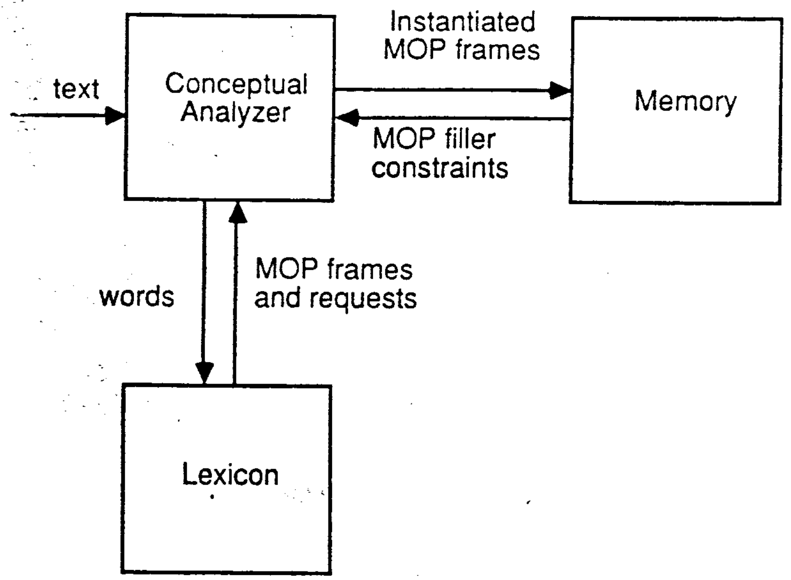

Fig. 2 — Memory-based conceptual analyzers.

> John went into a restaurant.

ELI would have generated something similar to

```
(PTRANS (ACTOR (PERSON NAME (JOHN)))
        (OBJECT (PERSON NAME (JOHN)))
        (TO (INSIDE PART (BUILDING TYPE
        (RESTAURANT)))))
```

which says that John moved himself into a restaurant building. (PTRANS is the conceptual dependency primitive for physical motion.) A reasoning module, such as SAM, would infer from this that John might be initiating the restaurant script by going into the restaurant to get something to eat.

Given the same sentence, one of these memory-based parsers would produce something like this:

```
(ENTER-SCENE (MOP RESTAURANT-DINE)
             (ACTOR PERSON-32)
```

where ENTER-SCENE is the opening event in the memory structure RESTAURANT-DINE, and PERSON-32 is the particular memory token used to keep track of the story character named John. In other words, much of the work that used to be left for the reasoning module had become integrated with the parsing process. The information that going to a restaurant involves physical movement (PTRANS) was still there, but it was stored in memory as part of the definition of the ENTER-SCENE, rather than in the lexicon.

DIRECT MEMORY ACCESS PARSING

The memory-based conceptual analyzers represent a transitional state between memoryless conceptual analyzers and direct memory access parsers. Presented in the manner we have just used, it is clear what the next stage of development is: integrate the separate lexicon into memory, and make parsing purely a memory process. This is what I call "direct memory access parsing."

I would like to survey several different efforts in this direction, most of them very recent (Quillian, 1969; Small *et al.*, 1982; Hahn & Reimer, 1983; Granger *et al.*, 1984; Waltz & Pollack, 1984; Charniak, unpublished.) My view of these systems will be focusing on somewhat different aspects than their authors intended. In particular, I'd like to draw out the particular problem and solutions that those authors have come up with that are most relevant to the goal of totally integrating memory with parsing. I am not claiming that such a goal is the primary interest of any of these other researchers. In fact, many people are more interested in what I think is a side issue, namely, how to speed up parsing with parallelism. While parallelism is indeed a common feature, at least in principle, with these systems, it is not, in my opinion, the important point. The real point is that parsing should be viewed as just a memory search process, and linguistic knowledge, including highly syntactic information, should be stored in memory in exactly the same way that other kinds of information are stored.

The teachable language comprehender

The first direct memory access parser is older than any system I have discussed so far. It is M. Ross Quillian's *teachable language comprehender*, also known as TLC (Quillian, 1969). TLC's notion of semantic memory (Quillian, 1968) had far-reaching effects in AI and psychology (see, for example, Fahlman, 1979; Collins & Quillian, 1969), but its model of parsing has not been seriously pursued until recently.

In TLC, English words pointed directly to nodes, called *units*, in a semantic memory. A unit had a pointer to a superset, plus zero or more pointers to properties. A *property* had an *attribute*, a *value*, and zero or more sub-properties.

Fig. 3 shows a simple example. The word "client" points to a unit that says that a client is a PERSON, with one property. That property is EMPLOY PROFESSIONAL, with one sub-proprerty, namely that the professional is employed by the client.

When TLC read an English phrase, it placed markers on the units that the words in the phrase referred to. For example, when it read "lawyer's client", it put markers on LAWYER and CLIENT.

Then TLC spread the markers breadth-first from the marked units to

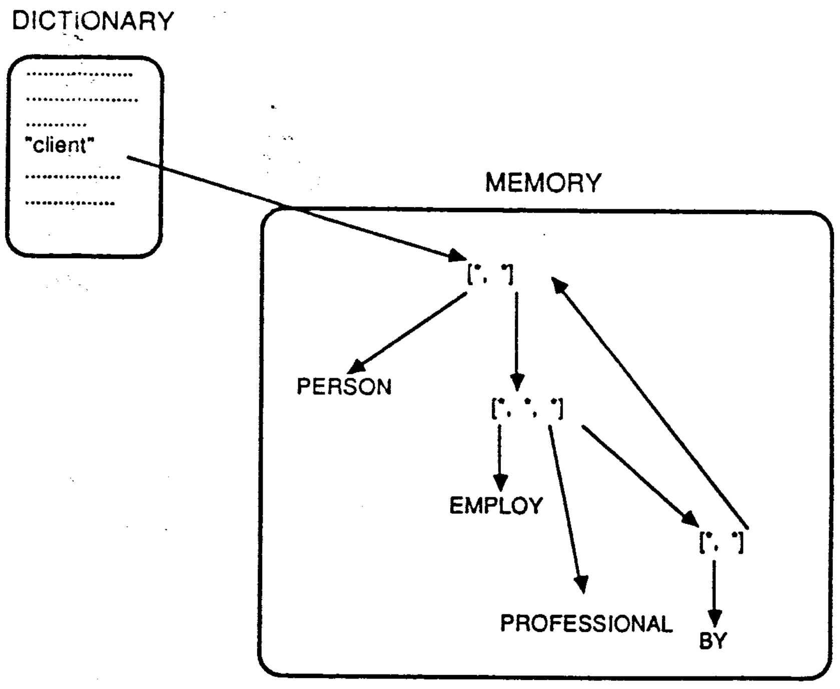

Fig. 3 — The teachable language comprehender (after Quillian 1969, p. 462).

their supersets and properties, then to those items' supersets and properties, and so on, until the markers met somewhere in memory. With "lawyer's client," CLIENT marked the property EMPLOY PROFESSIONAL, and LAWYER marked the superset PROFESSIONAL, and TLC found the intersection at EMPLOY.

An intersection was only a candidate connection. To determine if a connection was actually expressed by the input, TLC used *form tests*. Form tests were attached to memory units and looked for certain features in the input. When a unit was a candidate connection, its form tests were applied to the input. If the input passed any of the tests, then the connection was accepted as a meaning of the input.

For example, EMPLOY had a form test that said that the word referring to the property value must have an " 's" attached and be followed immediately by the word referring to the source word. In this case, "lawyer" referred to PROFESSIONAL and "client" was the source of the marker that went to EMPLOY, so the form test checked for "lawyer's client" in the input.

There are many problems that have to be solved with this kind of approach. One of the first is managing the intersection search process. For example, consider the sentence "The photographer shoots the doctor" and the related memory structure in Fig. 4.

The problem is that there are four possible intersection points visible, using just this bare fragment of memory (more would appear if we filled in more memory):

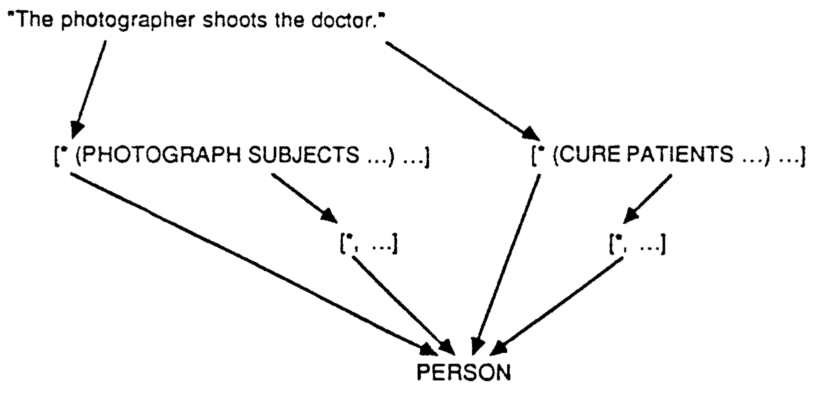
"The photographer shoots the doctor."
[* (PHOTOGRAPH SUBJECTS ...) ...]
[* (CURE PATIENTS ...) ...]
[*, ...]
[*, ...]
PERSON

Fig. 4 — Too many intersections in TLC.

- Photographers and doctors are both people.
- Photographers and doctors both do things to people.
- Photographers do things to people, and doctors are people.
- Doctors do things to people, and photographers are people.

Quillian notes that the first two kinds of intersections are unlikely to be useful in comprehension. TLC avoids them by putting one kind of activation marker on candidate word senses and their supersets, and another kind on properties and their supersets, and then accepting only those intersections where the two types of markers meet.

This leaves TLC then with two potential interpretations of "The photographer shoots the doctor." Either the photographer is photographing the doctor, or the photographer is a patient of the doctor's. Form tests attached to "PHOTOGRAPH" look for phrases, such as "X photographs (takes a picture of) (shoots) ... Y," which accepts the first interpretation. None of the form tests attached to "CURE" ("X cures (heals) ... Y," "Y visits (sees) ... X") work, so the second interpretation is rejected.

Quillian's made the following observations about TLC: TLC draws information from its memory to produce a representation of a piece of input text that is:

1. Encoded in the regular memory format.
2. Much richer and less ambiguous than the input text itself.
3. A highly intraconnected structure with the various concepts mentioned in the text linked in ways supplied from a memory of facts about the world.
4. Linked to the permanent memory by many pointers to established concepts (although its creation in no way changes the permanent memory, except for the addition of temporary tags used by the interaction routines) (Quillian, 1969, p. 467).

These are all desirable features for any conceptual analyzer (with one caveat), and yet they have not been true of any system since TLC, until recently, because of the separation of the parsing process from memory processes. If memory structures existed at all, they had only an indirect link to the structures being produced by the analyzer, in that, after the parser produced a form, the memory would match it against its stored forms.

The one caveat is that Quillian's fourth comment, that long-term memory is unaffected by parsing, is exactly what we *don't* want in a system that remembers and learns.

DMAP: A DIRECT MEMORY ACCESS PARSER

Quillain's TLC model had many problems, of course, most of them recognized at the time. Its linguistic knowledge was in its form tests, and its form tests were simple string patterns. Its knowledge representation was based on dictionary entries, with a strong English lexical bias. Its method of concept finding by interaction search was not really well matched to the needs of text understanding. It is the spirit of TLC, not the details, that is the inspiration for much of the modern work on direct memory access parsing.

Before DMA parsing can even begin, there must, of course, be a memory to access. A memory consists of one or more types of memory "elements," e.g. frame units, conceptual dependency forms, or predicate calculus formulas, interconnected by some kind of cross-referencing structure. For example, we can fill the slots of frames with pointers to other frames, or store forms in discrimination trees, or index formulas by their constant parts (see (Charniak *et al.* (1980) for a discussion of these options).

Given a particular memory model, designing a DMA parser involves figuring out how to store linguistic knowledge *in the memory*, and developing search processes to access this knowledge during the parsing process.

Aesthetically, it would be pleasing if we could store linguistic knowledge in memory using exactly the same memory elements and interconnection mechanisms used for other kinds of knowledge. For the moment, however, the Yale direct memory access parser (DMAP) has taken a simpler approach and represented linguistic knowledge with specialized data structures, called *concept sequences*, which are attached to MOPs and conceptual dependency action schemata.

For example, events involving transfers of information, such as "John says he'll come," are instances of the MTRANS action schema, which has the form "an *actor* transfers a *mental object* to a recipient," where the items in italics are roles of the schema. One simple concept sequence attached to the MTRANS schema is "*actor* MTRANS-WORD *mobject*." This sequence says that one way in which a transfer of information event is expressed is by saying who the actor is, followed by some word for MTRANS, such as "say" or "tell", followed by what the information (mental object or mobject) was. There are of course other sequences attached to the MTRANS schema, and the notion of MTRANS word is too broad, but this template is quite usable as it stands.

We are not claiming that parsing knowledge necessarily should be different in kind from other kinds of knowledge. We just do not know enough yet to find a good unified form for the two kinds of knowledge.

Attaching parsing knowledge to memory structure

Fig. 5 shows a figurative block diagram of the DMAP system. Since there is only one module, the memory, Fig. 5 goes into that box a little deeper to show language and conceptual hierarchies intertwined. This figure is intended to express the basic idea that linguistic knowledge, such as the lexical items "John" and "city", is attached to the hierarchically organized conceptual nodes in memory. Thus, for example, "post office" is attached to a node that is lower in the hierarchy than the node that "building" is attached to.

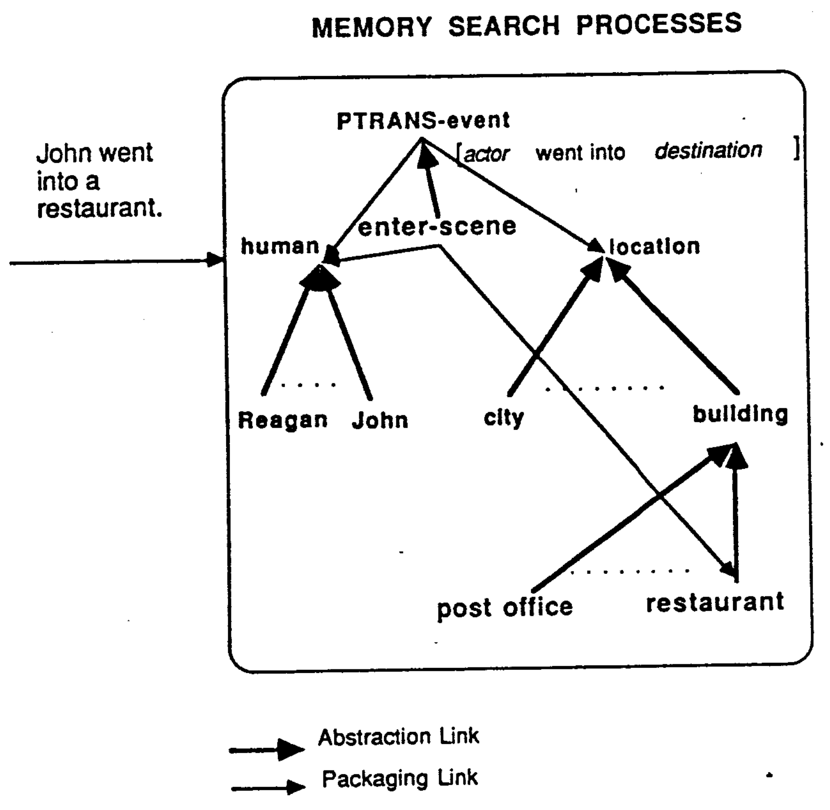

Fig. 5 — Direct memory access parsing.

The concept sequences attached to these memory nodes are phrasal patterns (Becker, 1975). Sequences are attached to objects, events, states, and so on. For example, attached to **PTRANS-event**, i.e. the abstraction of all events involving PTRANS, are phrases used to express motion, such as that the actor "went" to or into some place.

Concept sequences are made up of the following kinds of elements:

- particular lexical items, such as "interest rates" attached to the concept interest rates,
- lexical classes, semantically organized, such as MTRANS-WORD that groups words that mean "transfer of information," and
- role names, such as *actor* and *object*, that allow the concept sequence to refer to the fillers of the slots in the memory element to which the concept sequence is attached.

Our memory has basic frame units for objects, like restaurants and Reagan, and MOPs for events and state changes. **There are two standard kinds of links between MOPs. *Abstraction* links go from specialized nodes to more general ones.** For example, **enter-scene** has the abtraction **PTRANS-event** and **restaurant** has the abstraction **building.** We assume that many concepts have multiple abstractions and all our algorithms deal with that possibility.

2 links
1) Abstr.
2) Packag

The other kind of link is the packaging link. A node packages together other nodes. With objects, packaging links reflect part relationships. For example, the node for **human** packages body parts (not shown in Fig. 5). With MOPs, there are two kinds of packaging links. Fig. 5 shows the *role* packaging link. A MOP event has actors and objects and locations and so on which play different roles in the event. **Enter-scene,** for example, has role-packaging links to the actor and the location entered. The other kind of packaging link is the *subscene* link. A MOP event usually packages together several sub-events into a temporal sequence. Fig. 5 does not show this (Fig. 6 does), but **enter-scene** would be a subscene of the **restaurant** MOP. Normally our figures will label role-packaging links with the role involved, but will leave subscene links unlabelled.

When a concept sequence attached to a MOP contains a role name, that name refers to the role in the corresponding MOP. Thus, something like "*actor* went to *destination*," attached to **PTRANS-event** would mean that the MOP could be expressed as "the actor of the PTRANS went to the destination of the PTRANS," where the actor and destination would be filled in by phrases attached to the MOPs filling the actor and destination roles of **PTRANS-event.**

A concept sequence might be stored in many places in the memory, but in general it will be stored at the most abstract node possible, and be inherited implicitly by all specializations of that node. Thus, because "*actor* went into *destination*" is attached to **PTRANS-event**, it is inherited by **enter-scene.** This inheritance is done by the particular memory search process used in parsing, described in the next section.

Using parsing knowledge in DMAP

We store parsing knowledge in DMAP in much the same way that TLC did, by attaching simple patterns to memory nodes. DMAP uses these patterns very differently, however.

In TLC, markers were passed outward from referenced concepts until

they intersected at some common point, whereupon form tests at that point would be applied to see if a reasonable interpretation had been found. Marker passing was a general search mechanism and the linguistic patterns filtered out the desired results.

In DMAP, the patterns control where markers are passed in the first place. DMAP has two kinds of markers. Activation markers (A-markers) are placed on memory nodes that have been seen or are abstractions of nodes that have been seen. Prediction markers (P-markers) are placed on memory nodes that have been predicted or appear in concept sequences of nodes that have been predicted. That is, putting an A-marker on a node puts A-markers on its abstractions, while putting a P-marker on a node puts P-markers on certain of its parts and role fillers.

As in TLC, things happen only when markers meet markers. In DMAP, when an A-marker is passed to a concept that already has a P-marker, DMAP traces the P-marker back to its source, which will be some concept sequence that referred to this concept. Let's call the concept that received the A-marker the *part-concept* and call the source of the P-marker the *whole-concept.* Two things now happen. First, the concept sequence and P-marker on whole-concept are passed down the abstraction hierarchy to the most specialized version of whole-concept that packages some abstraction hierarchy to the most specialized version of whole-concept that packages some abstraction of part-concept. This process is called "concept refinement." Then the concept sequence is "bumped," i.e. a P-marker is passed to the next element in the concept sequence. If everything in the concept sequence has received an A-marker, then the concept to which the sequence is attached gets an A-marker, which is then passed up the abstraction hierarchy, and so on.

To see briefly how this works, assume we have the highly simplified hierarchy shown in Fig. 6, which says that one kind of **PTRANS-event** is **travel,** below which are the specializations **vehicular travel, other agent vehicular travel** and **air travel.** Though this memory fragment is just a toy for pedagogical purposes, already we can see various complexities creeping in. **Other agent vehicular travel,** for example, has a second abstraction, **other agent service,** which represents events where someone else does something for you. This means that **other agent vehicular travel** inherits two subscenes and two sets of role links, one set for the traveller and another set for what the other agent does for the traveller. (We ignore — and in fact do not handle well in the current implementation — the problem of conflicting role names, such as the presence of *actor* in both inherited scenes.) These scenes in **air travel** describe the pilot flying the plane and the passenger riding in the plane.

Every specialization inherits the packaging links of its abstractions. All the specializations of **travel,** for example, inherit the *dest* role with the filler **geographical location.** These roles are accessible to the subscenes as well, so that *air-trip scene* can refer to the destination of **air travel.**

Of particular interest are the concept sequences attached to the nodes in this memory fragment. **PTRANS-event** has the generic sequence "*actor*

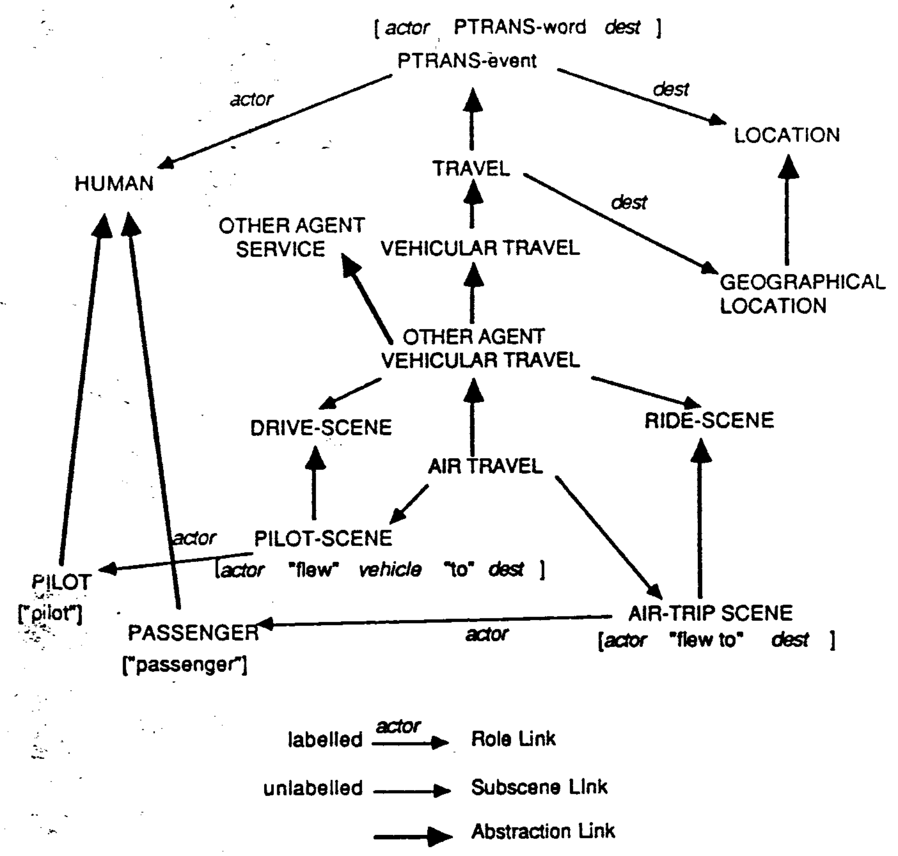

Fig. 6 — Air travel hierarchy with concept sequences.

PTRANS-WORD *dest.*" Suppose previous processing has predicted (passed a P-marker to) **PTRANS-event.** Because of the attached concept sequence, **PTRANS-event** passes a P-marker through *actor* to **human.**

Now DMAP reads a sentence starting with "The passenger ..." A-markers spread up the activation hierarchy from **passenger,** eventually reaching **human.** Most of the A-markers hit unmarked nodes, but **human** has the P-marker from the concept sequence on **PTRANS-event.** Concept refinement passes this sequence to the most specialized version of **PTRANS-event** that packages some abstraction of **passenger.** In this case, there is a specialization that packages **passenger** directly, namely **air-trip scene.**

Now the concept sequence is bumped, which means P-markers are passed to PTRANS-WORDs. Furthermore, since a P-marker has been passed to **air-trip scene,** which has its own concept sequence, a P-marker is passed to the lexical node "flew to." (The *actor* of **air-trip scene** is already active and therefore the concept sequence is immediately bumped.)

Concept refinement has refined the predictions being made in two ways.

First a more specific concept sequence has been added, looking for "flew to" where before the system was predicting only generic **PTRANS-words**. Second, the role filler predictions from the generic concept sequence are now more specific. The *dest* of **air-trip scene** is **geographical location,** not any location at all. The system is expecting a country or city name, not something like "The passenger flew to the store."

If the sentence was "The pilot flew ...," then "pilot" and **PTRANS-event** would have led to predictions on **pilot-scene,** and, instead of predicting "to", the system would be looking for **plane,** the filler of *vehicle* of **air travel** (not shown in Fig. 6).

Further details of this parsing algorithm can be found in Riesbeck and Martin (1985). The actual domain of application has been understanding economic arguments. DMAP's memory processes search through a crude but complex model of novice economic knowledge, identifying not only what abstract causalities are being referred to, but also whether or not these arguments are already present in memory. Currently, DMAP recognizes several dozen texts. Some typical examples are the ones referred to in the section called "Real life versus AI," and

> Milton Friedman: Interest rates will rise as an inevitable consequence of the monetary explosion.

> The President says that if Americans are given a 25% across the board tax cut they will suddenly become large savers.

What we think is very exciting is that concept refinement is not limited to generalized world knowledge. If we know about a particular trip our friend John took to Detroit, then "John flew..." will refine from **PTRANS-event** to the memory node we have for that trip, and the P-marker will pass through *dest* to **Detroit.** This smooth integration of particular episodic knowledge with parsing predictions is exactly what we want.

Ambiguity

Ambiguity, as in TLC, is a constant fact of life. Words have multiple senses, and the concepts referred to by those senses have multiple abstractions, and the abstractions appear in multiple packages. Linguistic choices, such as where a prepositional phrase attaches, appear in the system in the form of multiple concept activation, where one concept would be described by one choice, and the other concept would be described by the other choice. What decides which packages and concepts are correct?

In the conceptual analyzers, the decision was made in one of two ways. In ELI and CA, whatever came first in the dictionary took priority. In BORIS and IPP, the parser asked memory for a decision.

The first method was used most often, because the second method was too difficult. Because the inference engine didn't know exactly what was happening in the parser, it was hard to design good questions for the parser to ask the inferencer that would give the inferencer enough information to

make a reasonable decision. We explicitly rejected the alternative of having the parser produce structures for all the ambiguous interpretations, and pass the set of them to the inferencer for selection. It was, and remains, totally implausible to us that a good model of parsing would have no better idea at the end of a sentence of what the sentence was about than it had at the beginning of the sentence.

With direct memory access parsing, the whole nature of the problem has changed. There is no longer an issue of when the parser should talk to memory, and what information it should pass, because there is no separation between the two. In effect, the parser *is* "passing" all the interpretations to memory, but this occurs not at sentence or clause boundary, but continuously throughout the parse.

Furthermore, the disambiguation problem in this situation is purely a memory processing issue: given several packagings of some currently active concepts, choose which packages are relevant. It is not really a language issue at all.

In fact, DMAP, being strongly top-down, has more of a problem with too few packages, rather than too many. As we said in our algorithm description, a concept sequence attached to a package is not pursued unless it has been predicted (passed a P-marker). These initial high-level predictions are just what we need, but where do they come from?

Real life versus AI

We have now come to, I think, the crucial point, and also to the point where many of us will part company. As long as the memory that a DMA parser accesses contains the standard knowledge structures of AI and computational linguistics, I do not believe that it can do any better than any other parsing system. The secret is not in the algorithm a system uses, it is in what the system knows is going on at the moment of the parse. "Who is talking (writing) this?" "What do I know about them from before?" "What activities are they engaged in of which this speech (text) is a subpart?" "Why should I care about what this person is saying to me?" **Answering these questions is, I claim, the *primary* goal of human understanders.** Getting the meaning is *secondary,* often instrumental to the primary goal, but not always essential.

Most people in AI would probably grant that the ultimate understanding system should be capable of answering the questions asked in the previous paragraph, but that to get there, we have to take things one step at a time. Grosz (1977) studies these questions by setting up the problem of a tutor telling an assistant in another room how to put a complex object together. The goals are well defined and the tutor can try matching the assistant's questions and statements against a library of known scripts and plans.

I believe, however, that this still abstracts the language-processing task too far from its everyday foundation. Most of the people I talk to are well known to me. Well known means not only that I have met them before, but that I know what they're up to when they start talking to me. One may be arranging a time for a meeting that we've previously agreed to have, another may be checking to see if I've done what I said I would, another may be complaining about some advice I gave the week before. All of these are *on-going* long-term activities, which I quickly *recognize* as being relevant to the current conversation. Of the people I talk to who are less familiar, many of them are playing roles in very specific scripts, such as the waitresses in the restaurant I go to several times a week for lunch, the tellers in my bank, the receptionist at the doctor's office, and so on. Again, within a few words of conversation, I have usually recognized some very particular fragment of a script as being relevant.

Here is my rating of various language-processing tasks, from most common/typical/concrete to most unusual/atypical/abstract. I hear spoken language from family, friends, coworkers, secretaries, waitresses, shop-keepers, and other actors in social roles, television and radio announcers, usually giving updates of on-going events, such as arms talks or hostage situations, and acquaintances, old and new, at parties and other gatherings. The texts I read, in order of decreasing frequency, are signs and signals, such as stop signs and gas gauges, magazine and billboard advertisements, mail (mostly junk), articles in newspapers, articles in magazines, fiction, and technical articles in journals.

There is something in common between the speech I hear most often from family and friends, and the texts I read most often in signs and advertisements: both use language that is highly elliptical and presumes a great deal of background information. This reliance of "you know what's going on" is just as important in texts with a greater distance between speaker/writer and hearer/reader. Consider, for example, one of the texts that DMAP handles:

Q: Why are you so unimpressed with the recent improvements in the inflation numbers?
A: You have to look at inflation numbers in a cyclical perspective.

DMAP understands that the "you" in the question is referring to the interviewee, but that the "you" in the answer is referring to neither the interviewer nor the interviewee, but to the "audience". How does it do this, particularly given the fact that the only inferencing capability DMAP has is simple memory search?

The answer is that DMAP has a **magazine interview** MOP, which is a specialization of **interview,** which is in turn a specialization of **conversation.** A *magazine interview* involves one person asking questions of another for the purpose of giving the other person a chance to present views to a general audience. Lexical items, such as "Q:" and "A:", are part of the concept sequences for the two sides of the interview, and the content of the statements are specializations of, respectively, "question to speaker" and "statement to audience." Seeing strings like "Q:" and "A:" activate the **interview** MOP, thereby bringing in the concept sequences for interviews, which include different uses of "you," depending on who is talking. Most TV-viewing American adults have become familiar with (i.e. formed MOPs

for) many different kinds of interviews, such as the "promote your latest movie" and the "defend your election platform." Context and lexical cues guide the memory search down to these highly specialized subMOPS, which often have particular phrases and concept sequences attached.

Consider the difference between the use of "some people" when my wife says "Some people came by to fix the TV," and when a politician says "Some people feel that we can survive with a growing deficit." My wife means "a group of people that I'm not going to specify further," but the politician means "that group of people who have recently publicly disagreed with me." That is, "some people" is not at all an indefinite reference when a politician uses it. The DMAP model of understanding would handle this by attaching the phrase "some people" to the appropriate filler in the memory structure describing how a politician normally responds to opposition arguments without naming names. Obviously, "some people" would appear in many places in memory. This is why the memory search processes must keep active the most specific memory structures it can, "lighting up", in effect, the appropriate specialized uses of different phrases.

There's a good reason, of course, why AI has not dealt with understanding texts with complex, multi-goaled, highly familiar contexts. It's very hard to do. There are three major problems:

- knowing how to represent the necessary information,
- dealing efficiently with large amounts of inferentially interrelated information, and
- integrating parsing knowledge with everything else.

We believe that the DMA approach answers the third problem in a way that allows the parser to take full advantage of the information that an intelligent system needs. As long as the answer to the first problem involves using a packaging and abstraction hierarchy, markers can be passed to and from a DMA parser in a well-defined manner. And the use of marker passing offers the potential advantages of parallel processing to manage the large quantities of information involved.

OTHER APPROACHES

The connectionist/word expert system

The original word expert parser (Small & Rieger, 1982) was a conceptual analyzer, similar in some ways to the ELI system, but with a greater emphasis on the disambiguation process. Word experts were (sometimes large) programs attached to each word that communicated with other experts during the parsing process in order to reach an agreement on the meaning of the sentence as a whole. Small *et al.* (1982) recast the WEP approach into the University of Rochester's connectionist paradigm (Feldman & Ballard, 1982).

The connectionist framework does not allow structures like markers to be passed between nodes. The basic mechanism is spreading activation through a graph with fixed connections. The meaning of a text is represented by the cluster of nodes activated. Mutual inhibition as well as activation plays an important role in the parsing process, and critical is the use of nodes to represent concepts such as "the agent of the action PROPEL (or MOVE or whatever)."

For example, in parsing "A man threw up a ball," the eventual resolution of "threw up" as meaning "propel upwards" rather than "vomit" is arrived at because "a ball" is more closely connected to "object of PROPEL" than it is to "object of VOMIT," and activating the former inhibits the activation of the latter, which in turn makes the "PROPEL" node connected to "threw up" more active than the "VOMIT" node.

Major problems yet to be solved with this and similar connectionist systems include recognizing word order, keeping straight multiple occurrences of the same word or concept, and storing what has been read. It is also the case that most of these systems have been used only in simple semantic network systems, not in episodically based memories.

TOPIC

The TOPIC system (Hahn & Reimer, 1983) is also related to the word expert model of parsing. Effectively, they have replaced Quillian's form tests with word experts. They call their approach *text parsing*, and emphasize the interaction between world knowledge and linguistic knowledge during the parsing process. The following quote is indicative of their spiritual ties to direct memory access parsing:

> TOPIC parses text strings directly into knowledge structures (*semantic parsing*) by skipping any kind of intermediate structure, as might be provided by phrase structure trees whose nodes are labelled with syntactic categories (NP, VP, PP, etc.). [p. 3]

The TOPIC system has a hierarchically organized knowledge base of information about microcomputers. The parsing process activates concepts in this memory. A concept is activated when referred to by the text. That activation is increased with subsequent references.

One problem to address is that raised by the text fragment "... provided by *micros*. Nevertheless, these *machines* ..." The word "machines" would initially activate the general machine concept, but this needs to be corrected to be a second activation of the micros concept. This is done by the word expert for "this/these" which reassigns activation from more abstract to more concrete nodes in the appropriate circumstances.

ATLAST

ATLAST (Granger *et al.* 1984) uses spreading activation in a memory containing a blend of lexical, semantic, and pragmatic information. Three processes are in control: the lexical capsulizer, the proposer, and the filter. The capusulizer initiates concept activation as words are read and posts

syntactic information. The proposer spreads activation from concept to concept, effectively pursuing all possible inference paths. The filter prunes these paths, using in part the syntactic information posted by the capsulizer. An example text is:

> The CIA called in an inspector to check for bugs. The secretaries had reported seeing roaches.

In accordance with the lexical access data of Swinney and Hakes (1976) (which has inspired a fair amount of recent AI research into lexical disambiguation), the ambiguity of "bugs" is first available, then resolved, but incorrectly, then re-solved. The mechanism for doing this in ATLAST is called *conditional retention*. Preceding text will select one meaning of an ambiguous word, but, as long as there is potentially relevant text following, the other meanings are not actively suppressed.

Massively parallel parsing

Waltz and Pollack present a spreading activation, lateral inhibition model of lexical disambiguation (Waltz & Pollack, 1984). Syntactic, semantic and pragmatic information is represented in the same network, so that in parsing "John shot some bucks," the "throw off" meaning of "bucks" is connected to the verb usage of the word, and hence inhibited by "some" which favors the noun usage in "some bucks."

The single-semantic-process theory of parsing

Finally, there is the very recent single-semantic-process model (Charniak, unpublished), which explicitly pays homage to TLC. Charniak's model uses the "dumb" marker passing technique explored by Hirst and Charniak (1982) to propose limited chains of possible inferences to an abductive understanding system. That is, the primary problem Charniak is trying to solve is the unification of forms as "John picked up a menu" with the relevant line in the "go to restaurant" frame. This involves making abductive assumptions, such as that the menu John picked up is the one belonging to the restaurant he just went into, and that the event itself is unifiable with the event predicted by the restaurant frame.

Charniak's memory model, based on FRAIL, a frame-based representation language (Charniak *et al.*, 1983), has the standard abstraction and packaging hierarchies. The key idea is that finding potential forms to unify with an input can be done using a cross between spreading activation and marker passing. Of central interest to Charniak is the potential this model has to carry on syntactic and semantic processing autonomously but interactively.

Discrete markers, as in DMAP, rather than levels of activations, are passed up and down the two hierarchies, but how far they are passed is controlled by a "zorch" level. Each marker starts with a full amount of zorch. When a marker passes through a particular node, its zorch is reduced by the number of links from that node. When a marker's zorch level falls below a certain threshold, the marker is passed no further.

One of the main effects of zorch is to stop markers from passing through very general concepts. For example, the **animal** concept has so many links to kinds of animals that any marker reaching **animal** goes no further, because its zorch is reduced well below threshold. This is Charniak's solution to the **people** problem in TLC's "the photographer shot the doctor," discussed earlier.

Comparison

DMAP, the connectionist WEP, and ATLAST are close in terms of the content and form of the memories being searched, since they have a common origin in conceptual dependency representation theory (Schank, 1975). Otherwise, DMAP, TLC, and Charniak's parser have the most in common, since they both use marker-passing, while the connectionist systems use spreading activation. A node in a marker-passing parser is either active or not, whereas the spreading activation parsers have nodes with degrees of activation. Furthermore, DMAP and Charniak's parser both pass "structured" markers. More than just distinct tags, structured markers contain pointers to their origins. When a marker from one node reaches another node, it is easy to return to the source node. Many other marker-passing models and all the spreading activation connectionist models explicitly forbid this kind of structure passing, as being neuroscientifically unsound. Hence, both DMAP and Charniak's parser must be viewed as algorithmic descriptions that are several levels of abstraction above the brain hardware level.

SUMMARY

None of the systems that we have looked at is complete, nor are they all consistent with each other. But they do share the following basic theme: understanding of language is a kind of memory search process. They each offer some method for going as directly as possible from lexical input to concepts in memory. Once initial concepts have been found, they each offer some method for connecting these concepts to larger ones, still in memory. Highly language-specific information is still used, but it is stored in memory, rather than in a separate lexicon.

In several cases, the development of these systems is motivated by a desire to be neuroscientifically plausible, or to take advantage of low-level parallel hardware. But the real pay-off, I believe, is that these systems have broken down the boundaries between parsing and memory-based inferencing, opening the door to language-understanding systems with greater flexibility and power than ever before.

ACKNOWLEDGEMENT

This work was funded in part by the Air Force Office of Scientific Research under contract F49620-82-K0010.

REFERENCES

Becker, J. D. (1975). The phrasal lexicon. In R. C. Schank, & B. N. Nash-Webber (Eds.), *Theoretical issues in natural language processing*. Proceedings of the workshop of the association of computational linguistics, June.

Birnbaum, L., & Selfridge, M. (1981). Conceptual analysis of natural language. In R. C. Schank, & C. K. Riesbeck, (eds.), *Inside computer understanding*. Hillsdale, NJ: Lawrence Erlbaum.

Carbonell, J. G. (1979). *Subjective understanding: Computer models of belief systems*. PhD Thesis, Yale University. Research Report #150.

Charniak, E., Riesbeck, C. K., & McDermott, D. (1980). *Artificial intelligence programming techniques*. Hillsdale, NJ: Lawrence Erlbaum.

Charniak, E., Gavin, M. K., & Hendler, J. A. (1983). *The FRAIL/NASL reference manual*. CS-83-06. Brown University, Providence, RI.

Charniak, E. (unpublished). *A single-semantic-process theory of parsing.*

Collins, A., & Quillian, M. R. (1969). Retrieval time from semantic memory. *Journal of Verbal Learning and Verbal Behavior*, **9**, 432–438.

DeJong, G. F. (1979). Prediction and substantiation: A new approach to natural language processing. *Cognitive Science*, **3** (3), 251–273.

Dyer, M. G. (1983). *In-depth understanding*. Cambridge, MA: MIT Press.

Fahlman, S. E. (1979). *NETL: A system for representing and using real-world knowledge*. Cambridge, MA: MIT Press.

Feldman, J. A., & Ballard, D. (1982). Connectionist models and their properties. *Cognitive Science*, **6** (3), 205–254.

Gershman, A. V. (1979). *Knowledge-based parsing*. Technical Report 156, Yale University Department of Computer Science.

Granger, R. H., Eiselt, K. P., & 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*.

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

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

Hirst, G., & Charniak, E. (1982). Word sense and case slot disambiguation. In *Proceedings of the AAAI-82*. Pittsburgh, PA.

Lebowitz, M. (1980). *Generalization and memory in an integrated understanding system*. PhD Thesis, Yale University. Research Report #186.

Lytinen, S. (1984). *The organization of knowledge in a multi-lingual, integrated parser*. PhD Thesis, Yale University. Research Report #340.

Quillian, M. R. (1968). Semantic memory. In M. Minsky, (Ed.), *Semantic information processing*. Cambridge, MA: MIT Press.

Quillian, M. R. (1969). *The teachable language comprehender*. BBN Scientific Report 10. Bolt Beranek and Newman, Boston, MA.

Riesbeck, C. K., & Martin, C. E. (1985). *Direct memory access parsing*. YALEU/DCS/RR 354. Yale University.

Riesbeck, C. K., & Schank, R. C. (1976). Comprehension by computer: Expectation-based analysis of sentences in context. In W. J. M. Levelt, & G. B. Flores d'Arcais, (Eds.), *Studies in the perception of language*. Chichester, UK: John Wiley.

Riesbeck, C. K. (1975). Conceptual analysis. In R. C. Schank, (Ed.), *Conceptual information processing*. Amsterdam: North Holland.

Schank, R. C. (1971). *Intention, memory, and computer understanding*. AI Memo 140. Stanford University, CA.

Schank, R. C. (1975). *Conceptual information processing*. Amsterdam: North Holland.

Schank, R. C. (1982). *Dynamic memory: A theory of learning in computers and people*. Cambridge University Press.

Schank, R. C., & Abelson, R. P. (1977). *Scripts, plans, goals, and understanding*. Hillsdale, NJ: Lawrence Erlbaum.

Schank, R. C., Tesler, L., & Weber, S. (1970). Spinoza II: Conceptual case-based natural language analysis. Stanford Artificial Intelligence Project Memo No. AIM–109, Computer Science Dept., Stanford University, CA.

Small, S., & Reiger, C. (1982). Parsing and comprehending with word experts (a theory and its realization). In W. G. Lehnert, & M. Ringle, (Eds.), *Strategies for natural language processing*. Hillsdale, NJ: Lawrence Erlbaum.

Small, S., Cottrell, G., & Shastri, L. (1982). Toward connectionist parsing. In *Proceedings of the AAAI-82*. Pittsburgh, PA.

Swinney, D. A., & Hakes, D. T. (1976). Effects of prior context upon lexical access during sentence comprehension. *Journal of Verbal Learning and Verbal Behavior*, **15**, 681–689.

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

Wilensky, R. (1978). *Understanding goal-based stories*. PhD Thesis, Yale University. Research Report #140.