Six Topics in Search of a Parser: . An Overview of AI Language Research

Eugene Charniak

Department of Computer Science
Brown University
Providence. Rhode Island 02912

1 Introduction

My purpose in this paper is to give an overview of natural language understanding work within artificial intelligence (AI). 1 will concentrate on the problem of parsing - going from natural language input to a semantic representation. Naturally, the form of semantic representation is a factor in such discussions, so it will receive some attention as well. Furthermore. 1 doubt that parsing can be completely isolated from text processing issues, and hence I will touch upon such seemingly non-parsing issues as script application. Nevertheless, the topic is parsing

Unfortunately, to present AI parsing work with any sort of historical accuracy would be to produce a bewildering forest of names (both of people and programs) Rather. I will rather try to extract from the historical record a group of ideas which 1 believe can be molded into a coherent framework Naturally, even within these limitations my remarks will be sketchy - this is an article, not a book

II Syntax

Parsing takes us from natural language(in this paper English) to a semantic representation While we will discuss semantic representation later, for the moment we need simply note that whatever form this representation takes, it must at the very least clearly and unambiguously indicate the logical relations in the sentence. That is, in a sentence like "Jack pushed the button." it must clearly distinguish the pusher from the pushee In English this is typically indicated by word order, but other factors can enter in, as in the case of the passive sentence "The button was pushed by Jack." 1 bring this up. of course, because it is typically our knowledge of the syntax of English which gives us a first cut at this information

A. Rules of Syntax

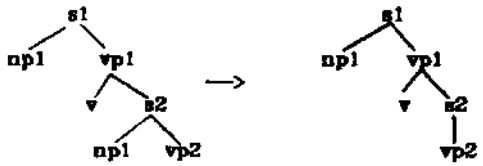
Given that we need to use syntax, the next thing we must decide is if we are to have explicit rules of syntax Actually, if interpreted in the narrowest possible fashion, there is a general consensus on this point After all, if we do not have such explicit rules, then the only alternative is to have the rules implicitly buried in the code of the program (or possibly buried in rules for other things) Given the general popularity of "rule based systems" these days, it seems reasonable to assume that we will make our knowledge of syntax "rule based" as well

'Many of the opinions expressed in this paper were forged during conversations with Graeme Hirst and the selection of topics would have been radically different were it not *lor* his input This research was supported in part by the Office of Naval Research under contract N000M-79-C-0592. und in part by the Notional Science Foundation under contract IS)-WO 13609. apologies to Pirandello

Thus we will have some rule which will catch the distinction between

Jack wants to go to the store.
Jack wants Bill to go to the store.

and state that when the internal clause is missing it's subject then the subject is interpreted as the subject of the higher clause. As with other rule based systems, our rules will consist of two parts, a test, which tells us if the rule is applicable, and an action to be performed if the test allows us to proceed. So, for example, in transformational grammar this fact is captured by a rule called *equi np deletion*. Since rules in transformational grammar start with "deep structure" and derive surface structure, the rule would look something like:



There are two conditions on this rule First npl must equal np2 (that is. we must have equal noun phrases). Second, the verb v must allow for equi np deletion. (As opposed to, say, "believe". We cannot say "Jack believes to go to the store " when we want to say something like "Jack believes that he went to the store".

In parsing, of course, we want to infer the deep structure from the input string. Thus our rule (which we might call *equi np insertion)* will look much different. Nevertheless, it will still consist of a test (see if there is reason to believe that the subject of a2 has been deleted) and an action (assume that the subject of s2 is npl

B. When Syntax, when Semantics

Now up to this point my analysis has not been controversial I have not claimed, for example, that these syntactic rules are applied before semantic rules. Nor have I claimed that there is a separate syntactic parser. It is claims such as these which get the arguments flowing In the classical linguistic view, of course, syntactic rules are applied to the sentence, producing a complete syntactic tree before any semantic analysis is done. Needless to say, such a view *is* controversial. Indeed, very few within AI hold such a position At the very least it is assumed as in Winograd's SHRDLU program [28] that syntax continually calls upon semantics to judge the progress of the parse to date, and hence uses it to guide the parsing process So. for example, in a sentence like

Jack told the boy about the plot.

the phrase "the boy about the plot" could be interpreted as a single noun phrase, as in "the block on the table" That is. syntax cannot rule out this interpretation.

However, semantics could presumably note that "about the plot" docs not make much sense as a modifier of "the boy" (The only possible hold out here might be Woods In [32] he points out that the time needed to do semantic processing in his system is longer than that needed to follow alternative syntactic parses. He may have changed his view on this, but I have not seen it in print However, few people have been convinced by this argument, since most assume that it says more about our lack of knowledge about semantics than anything else.)

But relegating semantics to a subroutine of syntax (let us call this the "semantic subroutine" position) is quite controversial. Schank and his students [22.23] have argued for a much tighter connection - one in which there is but a single system with both syntactic and semantic rules We will call this the "combined semantics" position. While there have been a variety of arguments for such a position, ranging from psychological experiments to efficiency considerations, to me the most cogent is that people understand ungrammatical sentences. The argument goes like this: if we had a syntactic parser in charge of everything then it would be responsible for calling the semantic routines. So, in a sentence like "The boys is hungry." the syntax would have failed to parse it, hence producing nothing which could be handed off to semantics, and hence no understanding

While I know of no written response to this argument, several alternatives seem to come up in discussions. One is to use a "blackboard" model as in HEARSAY [11] In this scheme the syntax and semantic components work in parallel, leaving their conclusions on a common "blackboard" The idea is that even if the syntactic component fails, the semantic component will still succeed, at least if the sentence is comprehensible

That there are troubles with this model is suggested by the fact that HEARSAY itself did not really use it While technically the two systems worked in parallel, in actual design the semantics would only work on the output of the syntactic component, making the model isomorphic to the semantic subroutine model proposed earlier Nor is it hard to see why this should be the case As we have already noted, syntax is the most obvious way to get a first cut at the logical structure of the sentence In a real blackboard model the semantics would be forced to do without the logical structure established by the syntactic component. This would mean that it would have to re-establish the same information by other means If it could do this, then why bother with syntax at all[0] In other words, a blackboard model will ultimately degenerate into either a "no syntax" model, or a combined semantics model.

The other alternative to the combined semantics model is one.in which we have a syntactic parser, but the grammar it uses is not the "correct" grammar for English, but rather a much more permissive grammar The idoa is to make the permissive grammar such that if a sentence is ungrammatical according to these permissive standards, then it will be incomprehensible as well. But there are still problems here We have already noted that people understand sentences in which subject verb agreement is not obeyed, eg "Jack have the ball". Yet some sentences can only be completely understood if we use this rule For example

The fish is hungry.
The fish are hungry.

If our permissive syntax does not check for agreement then we will not have extracted the full measure of meaning from such examples.

Perhaps one final point here We have become so accustomed to the fact that we are concerned with the extraction of meaning from sentences, and *not* accounting for grammatically judgements, that it is, at times, remarkably easy to forget the fact that people *do* have grammatically judgements If we are at all interested in psychologically plausible models of language comprehension then we must account for this fact Permissive syntax simply will not do this for us. short of including two sets of rules, one permissive, one strict - a rather ad hoc solution at best

The way out of this bind is to take a slightly different approach Rather than assuming that the grammar of the language should account for our ability to understand ungrammatical sentences, it seems more plausible to me to assume that one has a standard "correct" grammar for the language, and one tries to apply it to the sentence. If the sentence is ungrammatical, then somehow it should be the parsing mechanism itself which allows for the ungrammaticality. That is, it should note the ungrammaticality, but go on anyway, doing the best it cr*n

However, the most popular of the syntactic parsers, the "augmented transition network" parser (ATN) [32] cannot do this The need to reject ungrammatical sentences is built deep into the workings of ATNs. Ultimately we can trace the problem back to the method ATNs use for handling situations where there is more than one possible interpretation for the next sentence constituent. For example;

Jack gave the ball to the boy
Jack gave the boy the ball

After parsing the "gave", the next noun phrase may be either the direct object "the ball" (in the first sentence), or the indirect object "the boy" (in the second sentence) Suppose for the first of the above cases it guesses that the noun phrase is the indirect object After parsing the noun phrase it will then be immediately expecting a second noun phrase, the direct object, as in the second sentence Of course, it will not be there, a prepositional phrase is there instead The point is that upon finding that there is no way to proceed, an ATN cannot assume the sentence is ungrammatical at that point and try to fix things up. In the case at hand the sentence is not ungrammatical. Rather, the inability to proceed stems from an incorrect guess made at an earlier point Hence the ATN will backtrack to the last decision point and try an alternate guess.

C. Parsifal and Paragram

The solution to this problem is deterministic parsing If we have a parser which need not backtrack then failure at a given point in the sentence *urill* indicate ungrammaticality. and hence wc have the possibility of dealing with the ungrammatical sentence Marcus' Parsifal [10] is such a parser, although it does not handle ungrammatical sentences.

The basic idea behind Parsifal is that we have a *buffer* which allows us to have a limited look-ahead in the sentence. In Parsifal the buffer is of size three, so at any point we may have three constituents (such as individual words, but they may be larger units as well) which have yet to be attached to the overall syntactic tree Once wc have decided how a constituent is to be attached, however, wc cannot change it So, for example, the Parsifal rule for equi np insertion would look something like this:

```
(rule equi-np-insertion
    ;We give the rule the name equi-np-insertion
  [= to] [= verb tnsless] [the higher verb is +equi] -->
    ;This is the test portion of the rule, we will
    ;explain it below.
 Insert a copy of the subject of the next higher sentence
    into the buffer before the 'to'.
    ;This is the action.  It does the insertion of the
    ;understood noun phrase.
 )
```

In a sentence like "Jack wants to go home " equi-np-inscrtion applies after we have parsed the "wants". The test portion of the rule checks that the next thing to come is the word "to", and that it is followed by a "tenseless" verb. That is, by a verb which has the infinitive form, as, in this case, "to go". Finally it checks that the higher verb, in this case "wants" allows equi np insertion to take place

Now, as we already noted, Parsifal will not parse an ungrammatical sentence For example, if we gave it

Jack wants go to the store

the above rule of equi-np-insertion would not apply because the "to" is missing However. I have written a parser. Paragram. which will handle at least some ungrammatical sentences [6] While Paragram differs from Parsifal in several ways, the point which is important here is that rather than have all or nothing tests as in Parsifal, (that is. the test is either true or false) Paragram's tests produce a "goodness of fit" rating Thus, in the case of an ungrammatical sentence, when we reach the point of ungrammatically, all of the tests will have bad fits, yet, baring ties, there will still be one rule which looks marginally better than the others In the case of the above sentence with the missing "to", equi-np-inscrtion is the best of a bad lot for Paragram, and hence Paragram will "hallucinate" a "to" in the right spot, and proceed to apply equi-np-insertion anyway, thus successfully parsing the sentence.

III Semantics

Given a possible solution to the problem of ungrammatical sentences, we have cleared the way for a resumption of the semantic subroutine model of parsing Thus we are adopting a model in which the syntactic parser produces some portion of a syntactic tree. and. at predetermined times hands this off to semantics Semantics then helps guide the parse, resolve ambiguity. and builds the final semantic representation

A. Woods Procedural Semantics

A while ago, when I was trying to figure out what to teach my students in my undergraduate AI course, 1 became rather dissatisfied with the state of semantics in AI Idealy what I wanted was some rule based system for turning the output of syntax into a semantic representation. The trouble is. that none of the recent work in semantics is sufficiently formalized to allow for this 1 found mvself forced to turn back the clock about ten years and return to Woods procedural semantics [31J

Wood's system was a classic rule based system, as we can see by examining one of his rules First, we have several general patterns for syntactic trees

Now consider a rule which will handle a sentence like this:

AA-57 departs from Boston.

The rule is:

TEST: In G1: **np** is a flight and **v** = leave or depart
In G3: **prep** = in or at and **np** is a place

OUTPUT: (**depart** G1-**np** G3-**np**)

The output is a pattern in which we substitute the objects found in the input. So G1-**np** would be the **np** found in pattern G1, e.g., AA-57.

Now this is nice because it is fairly easy to give a student this type of work and have him understand it. The situation action type rules are easy to understand, so the student is, with one exception, pretty much able to determine the behavior of the system from an examination of the rules.

The exception, the only part which is not reasonably clear given just the rules, deals with quantification. In order to handle sentences like

Every plane flys to Boston

Woods must engage in some slight bit of complicated machinery. The rule for handling "every" translates it into the following:

(for every X ∇ Δ)

Roughly speaking what will happen is that the translation for "plane", (**plane** X)" , will be substituted in for ∇ , and the translation for "flies to Boston", (**arrive** X **Boston**)" will be substituted in for Δ . The result will be:

(for every X (**plane** X) (**arrive** X **Boston**))

There is a bit more complexity, to insure that the same variable (in this case X) is used in each subformula.

While this is not too bad, it is nevertheless a part of the system which is not adequately rule based, since, as we did here, one must abandon the rules and give a rather lengthly explanation in English We shall return to this point in a moment.

B. Montague Grammar

One possible source for formalizing ideas would be linguistics. Upon turing in that direction however, we find that the major trend in linguistic semantics over the last few years has been Montague grammar, which, as we shall sec, is a rather odd companion for AI language work

Montague grammar is exceedingly complex, and there is no hope that 1 could explain it here. For the reader who wants to learn more about Montague grammar, the most accessible introduction is the textbook by Dowty et al. [10]. As in transformational grammar, a set of grammatical rules analyze the structure of a sentence As opposed to transformational grammar, however, each syntactic rule is associated with a semantic rule which turns the sentonce into a formula of a rather complicated formal logic (The combination of syntactic rule plus translation into logic is essentially just another test/action rule.) The logical formula, in turn, is

Assigned a meaning according to certain rules which define the semantics of the logic. Again, the details here need not concern us. All we really need note is that the logic deals with so called "possible worlds", and many of the expressions in the logic have, as their meanings, quantifications over all possible worlds So a certain expression would be true if in all possible worlds in which X is true, Y is true as well, or some such

Now from an AI point of view there are a lot of problems with Montague grammar The actual syntax Montague used was rather trivial, but this is being attacked by the linguists attracted to the framework More troublesome are the issues we run into if we try to take seriously the idea of using the semantics of the formal logic as a way of inferring facts about the world Quantifying over all possible worlds is, to quote Wilks [27] "no joke" For this and other reasons Wilks rejects Montague grammar as relevant to AI.

By and large, Wilks' view has been the majority one Thus, while Friedman [14] has implemented a Montague grammar parser, it was clearly intended as a toy system. suitable only for learning about Montague grammar It does not seem intended as a useful AI system Hobbs [17] has also considered the use of Montague grammar. but never implemented a program. It would seem that he too had trouble making quantification over possible worlds compatible with AI

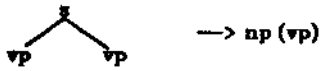
While I basically agree with the above criticism. 1 have recently become more sympathetic to Montague grammar, due to two rather nice features The first is the way in which Montague grammar handles quantification At first glance, quantification in Montague grammar looks a mess So rather than tackle it directly. let us start with a simpler example

Bill sleeps.

"Sleeps" is easily enough. It is translated into some predicate **sleeps**. However "Bill" has a rather bizarre translation. It becomes

(lambda (P) (P **Bill**))

Here I am using the normal lambda calculus, although expressing it in LISP formalism. In this formula, **Bill** is the symbol for the individual referred to by the English word "Bill". Now while this looks pretty odd, note that combining subject and predicate in Montague grammar is now a very simply operation - that of functional application. That is



That is, we apply the function represented by the noun phrase to the verb phrase. In the case at hand this gives us:

((lambda (P) (P **Bill**)) **sleep**)

This still looks pretty odd, but as we all know from normal LISP, this can be simplified because P will now become bound to **sleep** and we can replace it throughout by its value, giving us

(sleep Bill)

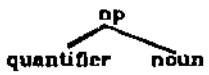
Now this looks like a rather complicated way to produce the obvious. But note what happens when we have quantification. So consider a sentence like:

Every dog sleeps

This will have the following representation (I am changing the Montague formalism a bit here, for reasons which should become obvious)

```
Every ----> (lambda (N)
              (lambda (P)
                (for every (X) (N X) (P X))))
dog   ----> dog
```

Now we add a rule for combining "every" with "dog":



With these rules the noun phrase "every dog" will be analyzed as follows.

```
Every dog ---> ((lambda (N)
                  (lambda (P)
                    (for every (X) (N X) (P X))))
               dog)
           ---> (lambda (P)
                  (for every (X) (dog X) (P X)))
                by lambda substitution
```

When we then combined this with the verb "sleeps", we get:

```
Every dog sleeps ---> ((lambda (P)
                        (for every (X) (dog X) (P X)))
                      sleep)
                  ---> (for every (X) (dog X) (sleep X))
```

Now the point of this exercise is that this last formula is exactly what Woods would have come up with for this sentence. Once we see this, we can then see that the rather complicated expression Montague uses for "every" is really just the same as what Wood's had The only difference is that while Woods had to depart from his rule based system in order to state in English what his system did with all of this. Montague, through the use of the lambda calculus, is able to express everything within his rule formalism. In other words, for this example, anyway, the Montague formalism is the better one if one wants a truly rule based system.

There are other nice features of Montague as well One is his handling of the verb "to be" Typically AI systems handle "to be" as ambiguous between "is of idrntity" and the "is of predication" We have identity m

Mark Twain is Sam Clemens

This is normally represented by something like

(= mark-twain sam-elemens)

The "is of predications" is seen in

Mark Train is a man

This gets a representation like

(man' mark-twain)

Montague is able to have a single sense for the verb "to be" yet end up with the two distinct representations shown above. The differences stem from the difTerences between his representations for "a man" and "Sam Clemens". This is not, in itself, an earth shaking development, but it is a nice touch

Because of these nice touchs, 1 have become interested in Montague grammar as a useful tool for AI Nevertheless. 1 should il make it clear that in most respects my earlier opinions have not changed Il is still the case that quantification over possible worlds is out of the question lor this an other reasons, the actual semantics of Montague grammar does not interest *nw* Nevertheless, Montague's method of developing the

semantic representation using only the lambda calculus and functional application looks very cute to rne. and I would like to see if it can be adopted to more reasonable AI systems The example from Woods suggests that this *is* not an idle possibility

C. Word Sense Ambiguity

But for all of my recent infatuation with Montague grammar, it should be clear that the Linguists are not really going to help us very much The major problem in semantics is not. as discussed above, how to combine predicates into the semantics representation, but rather the disambiguation of predicates ("river bank" vs. "commercial bank") and the determination of noun phrase referents ("Jack put the money in the jar and then put *it* on the self") Since virtually all linguists ignore these problems we are on our own

Needless to say. AI has not been standing still since Wood's paper ten years ago However, the work has not been easily systematrzable To give you some feel for the sorts of things which have been done, and the problems which remain if we are to incorporate them into some easily comprehensible rule based system, let us just look at some of the work which has been done on word sense ambiguity

Probably the most direct attack on word sense disambiguation is that of Rieger and Small [21] Rieger and Small are particularly interested in cases where a single word has a rather large number of uses So, to take a reasonably typical case, the word "call" can. to give just a few, be used in the following ways

[1] Jack picked up the phone and called Fred
[2] Jack called to Fred to come upstairs.
[3] Jack called Fred a dirty name
[4] Fred raised fifty cents and Jack called him
[5] Jack called for an investigation
[6] Jack called the meeting for 10am

Nor is this complete by any means

Confronted with such examples we have two basic problems, the more difficult is deciding upon the factors which are relevant to distinguishing between the senses So, we might note that some of the uses in the above examples take a direct object (call *Frv.d* on the phone), where others use a prepositional phrase (called *to Fred* to come upstairs). We can and should use such information, but we must be very careful So, we can also say

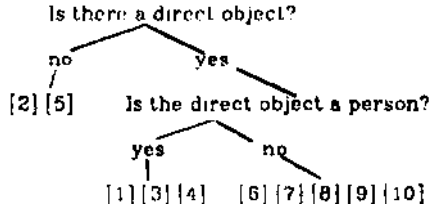
Jack called *Fred* to come upstairs

Furthermore, in some cases the local information within the sentence is not sufficient So, "Jack called Fred" could be phoning, shouting, or poker, to name just a few

But even having decided upon the factors which enter into the decision, we have yet another problem How and when do we apply these facts For example, if, for any given word we have a small number of sense, say. 5 or 10, then it would seems reasonable to just go through each possibility in turn, and perform the various tests which would rule the possibility in or out However, for words such as "call" the number is certainly larger than 10

[7] Jack called the debate a draw
[8] The banker called the loan
[9] Jack called the dance
[10] The game was called due to rain

Rieger and Small suggest that we use a discrimination net for such cases Discrimination nets are a good candidate because if designed properly they reduce the time for a decision from linear in the number of items to the log of the number of items, although whether this can be done for a particular net depends on the kinds of tests available A net for our example might start out:



From the viewpoint of rule based systems, there are several problems with this Firstly, given we have not specified what sorts of tests appear at the nodes, there are no real constraints on such networks Secondly, in such a system we cannot extend our knowledge by simply adding another rule A child (or a programmer) who wanted to add a few new senses for a word would have to modify a rather complicated discrimination net On the other hand, it may well be feasible to automate the construction of such nets, given basic knowledge of how individual senses arc used, in which case the force of such objections would be substantially diminished.

This is. by no means, the only suggestion which has been made concerning word sense disambiguation in the last several years The other idea which I want to discuss, however, is based upon the script idea of story comprehension, so we must break for a discussion of knowledge representation and text comprehension.

IV Knowledge Representation Issues

The last several years have seen a fair amount of work on various knowledge representations. Roughly speaking we can distinguish two trends On one hand there have been various predicate calculus based representations, while on the other we have the script /frame based systems.

In fact. 1 believe that these two traditions can (and should) be merged, and my current knowledge representation language. FRAIL [7], uses both the predicate calculus and frames However, since most of the AI work related to parsing has assume one or another of the frame/script approaches.] shall concentrate on this formalism here

A. Comprehension Based Representations

Even within the frame/script approaches one can distinguish two trends. On one hand we have those systems which take as their starting point the need to provide a knowledge representation to underlie the comprehension of text Certainly the Yale script representations [24,9] fall in this category, as docs my penultimate frame representation [5] A typical example of such a representation might go as follows:

```
(frame: cigarette-lighting
        ;The name of the frame is cigarette-lighting
 slots: (agent (a human))
        (patient (a cigarette))
        (instrument (a match))
        ;There are three objects which play roles in
        ;this activity, an agent who does it, the
        ;thing which is lit (patient), and an
        ;instrument used during the process.
 script: (?agent lights ?instrument)
        ;The sequence of activities, which I will call
        ;a script, consists of lighting the instrument
        ;and bringing it in contact with the cigarette.
        (?agent brings-into-contact ?instrument ?patient)
    )
```

I assume that with the English glosses, the above should be reasonably self explanatory. Basically, a frame such as this is said to be "active" if the activity it describes comes up in a text At that point the program will try matching further actions against those predicted by the script section of the frame If it succeeds, it has understood the action in terms of a higher level goal

B. General Frame Representation

The other major trend in frame representations have been those which arc less directly tied to language comprehension, and have some pretensions to generality. Under this heading 1 would fit KRL [I], NETL [12], KL-ONE [3], and FRAIL [7]. By and large these languages are characterized by

1) More attention to the "semantics" of the representation
2) Greater use of the *isa* hierarchy to eflect a reduction of space by placing required knowledge at the greatest possible level of generality
3) Lack of any explicit script section to the frame, as seen above, with most of the descriptive burden falling on the slot section of the frame

Again, the details need not concern us What is of concern is the degree to which the two frame groups have gone off in incompatible directions Fortunately this seems to be minor. Of the above differences, the only important one is the third However, this can be circumvented by the simple expedient of making the events which are part of the script secion of the frame into slots themselves. Thus, in such a scheme our cigarette-lighting frame would become

```
(frame: cigarette-lighting
   isa: intentional-action
        ;I have added an isa section.
 slots: (agent (a human))
        (patient (a cigarette))
        (instrument (a match))
        ;The three previous slots
        (light-step (?agent lights ?instrument))
        ;The sequence of activities, have now become
        ;a sequence of slots in the frame.
        (contact-step
         (?agent brings-into-contact ?instrument ?patient)
    )
```

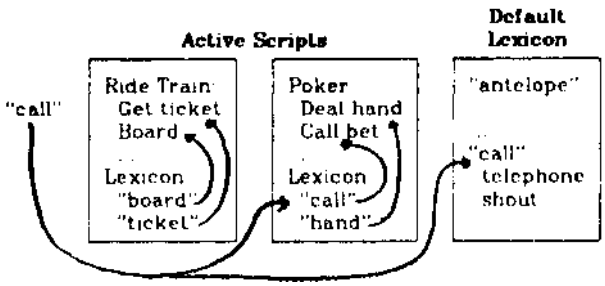
Furthermore, if you do not like the format imposed in which slots and actions are intermixed, it is certainly possible to introduce some syntactic sugar to make the representation look more like the script format shown earlier. FRAIL [7, 30] does exactly this

V Interactions between Parsing and Everything Else

With the discussion of text comprehension out of the way. we can now look at approaches to parsing issues which are dependent on either the form of the representation, the use to which it is made during comprehension, or both

A. Contextual Disambiguation

Rieger and Small's discrimination nets attacked the problem of word sense disambiguation in a brute force way. A rather different approach to the problem depends heavily on the use of scripts during language comprehension In this approach rather than listing all possible senses in a large dictionary (or lexicon) we would have a "default" lexicon which would give only the most standard definitions For "call" it might just be restricted to shouting and telephoning However, we would then have associated with each script/frame a lexicon, which gives for each notion used in the script the word associated with it and vice versa [16.9] Thus, in the poker script we would have an entry for "call" pointing directly into the part of the script which deals with "calling a bet"



Thus, in looking for a word sense, we would first look to see if the word appeared in the lexicon of an active script, in which case wc would automatically use the con textually suggested meaning

The advantage of such an approach is that it suggests a way to avoid the problem encountered earlier in organizing large numbers of senses for a particular word In the script/frame approach we will only have a few to consider, and we would naturally consider the contextually relevant ones first On the other hand, this places some rather sever constraints on our scripts/frames If there is only a small number of such frames active then this is fine, but if, as 1 tend to believe, the number might be quite large (they were playing poker with pornographic playing cards on the train on the way to a birthday party while one player was eating a hamburger and another was talking about yesterdays baseball game) then wc have simply replaced a long search through word senses by a long search through scriptal lexicons

Because of such problems 1 have grave doubts about this approach On the other hand. I have some doubts about the discrimination net approach as well, and it is fair to say that 1 know of no approach to word sense disambiguation which 1 find encouraging, much less satisfactory 1 suppose that *my* objections to the discrimination net approach are less severe, but it is a rather dull approach The scriptal lexicon approach certainly has more flair

B. Framcs and Case

One area which seems particularly interesting to me, and which seems comparatively unexplored is the interaction between the processes of language comprehension and the form of semantic representation we use To some degree the scriptal lexicon idea has this flavor, but more interesting to me are examples where properties of language interact with features of the representation which are needed for independent reasons. Let me give one such example, concerning case theory in linguistics [13].

For those unfamiliar with case theory, the basic idea is that there arc certain cases associated with each verb These cases relate the arguments of the verb to the verb itself, and do so in a meaningful manner For example

AA-b7 files to Chicago

In this sentence the noun phrase "Chicago" is said to be in the goal, or destination case of the verb "flies". There are many reasons which have been put forward for wanting such cases One such is the tendency for noun phrases in the same case to take the same prepositions. So the verbs "travel", "go", "walk", "fly", etc , all take "to" with their destinations.

Several people have noticed the similarity between such cases and the slots of frames in Al frame representations This has lead some, for example Winston [29] to assert that cases of a verb are in fact the same thing as slots on a frame. While this comparison seems reasonable, it has not aroused much interest, presumably because nothing seems to follow from it So what if slots and cases are the same?

Interestingly enough, there are some consequences To take but one (see |B| for others), if slots and cases are the same, then if two things can fill the same case, it would seem to imply that they fill the same slot However, this can get us into trouble For example

Jack painted the wall with the brush
Jack painted the wall with the yellow paint

Here both "the brush" and "the yellow paint" are in the instrument case, yet we would normally assume that they should fill two different slots in the painting frame, one for the tool used, and one for the liquid used However, one of the aforementioned frame representations, KL-ONE, has a feature built into it which will handle this problem

In KL-ONE, not only is there an isa hierarchy of frames, but there is also a hierarchy of slots In particular, one can speak of one slot differentiating another slot So, if wr had a mammal which had a head-of slot, then it would be likely that head-of would differentiate the part-of slot in the phys-obj (physical object) frame Naturally, this is useful because wc would automatically know that whatever is true of parts of objects is true of heads of mammals Now, returning to the example at hand, wc might have the following representation for painting

```
(frame deliberate-action
   isa action
 slots (instrument)   )

(frame painting
   isa deliberate-action
 slots (tool differentiates instrument)
       (liquid differentiates instrument)    )
```

Here the painting frame has two slots, tool and liquid so there is no problem in distinguishing what goes on the wall, and what is used to put it there At the same time, since both are differentiations of the instrument slot, we still have the normal case relations

C. Frame Specific Slot Filling

In the examples we looked at in the last section we were able to fill certain slots because of syntactic clues in the input sentence In many cases, however, this is not possible. Typically in such examples the slot to be filled is quite specific to the frame at hand, so it would be unreasonable to expect the English language to make allowances for it in the syntax. For example, Schank and Birnbaum [23] discuss the problem proposed by examples like:

A plane carrying 150 people ...

Presumably in reading this sentence we establish some sort of air-travel frame/script. This frame will have several slots which may be filled, one of which is the passenger slot The problem here is to realize that "150 people" fills that slot, given that there is nothing directly in the sentence, at least when read literally, which suggests this.

After rejecting several alternatives, Schank and Birnbaum decide upon the following. Within the air-travel frame, (or perhaps within some frame higher on the isa hierarchy, such as machine-aided-travel, or some such) we will note that the purpose of this activity is to move passengers (and cargo) to some location That is, we will have something like:

(move ?passengers ?destination)

We then further assume that part of the meaning of "to carry" is that the direct object (in case theory this is often called the patient case) of the carry is moved. Hence we have:

(move ?patient ?destination)

In this example this becomes.

(move 150-people ?destination)

If we then do a match between this line and the corresponding line in the air-travel frame, we get as a side result that 150-people should be in the passenger slot

What is really nice about this is that it does not require any additional machinery beyond what we have already hypothesized for text comprehension While Schank and Birnbaum do not comment on this fact, the mechanism proposed here is our good old script application mechanism That is, the fact that the passengers of an air-travel are moved from place to place will be part of the script describing the actions taken during air-travel If we do normal script application we will find the match described above, and in the course of this match we will be required to fill the passenger slot as indicated

A similar proposal is put forward by Hayes [15]. Hayes is concerned with examples like:

Jack got in the car and touched the steering wheel

The problem here is to recognize that the referent of "the steering wheel" refers to *the steering wheel of the car just mentioned* Or, to put this somewhat differently, assuming that we have some sort of automobile frame, the referent of "the steering wheel" will fill a certain slot in the instance of the automobile frame which wc created to account for the automobile mentioned earlier in the sentence Presumably, the automobile frame will look something like:

```
(frame  automobile
   isa: traveling-instrument
 slots; (door~of (a door))
        (wheel-of (a steering-wheel))

)
```

In the example of an airplane carrying people, we made the connection based upon the actions which take place during airplane-travel. In this last example there is no action, rather we want to make the connection on the basis of the match between the description of the newly mentioned object, namely steering-wheel, and the restriction on the wheel-of slot in the automobile frame, namely that it be a steering-wheel.

While it is not unreasonable to assume that we have two slightly different mechanisms, one for making such associations on the basis of scnptal actions, and one on the basis of slot restrictions, there are reasons to assume that these are really the same thing In particular, when 1 talked about how FRAIL negotiates the differences between script based and general purpose frame representations, I noted that in FRAIL a scnptal action was simply a slot on the frame That is. what typically is represented as

```
(frame: airplane-travel
   is a machine-aided-travel
 slots  (passengers (a person))

script  (move ?passengers ?destination)
        -)
```

would, in FRAIL be expressed exactly the same way, but would, internally be translated into

```
(frame  airplane-travel
   is a machine-aided-travel
 slots, (passengers-of (a person))

       (move-step (move ?passengers-of ?destination)  )
```

Thus, in both cases, filling a slot in a frame is the result of a match bet-ween the incoming statement and the restriction on a slot in an active frame While there are a lot of problems with the details of all of this, I find it encouraging that various strands of work are slowing comming together

VI Conclusion

1 have presented a medley of topics and ideas here, and to conclude let me try to draw them together and sketch what an overall parsing mechanism which included them might look like

Roughly speaking, the parser would come in three parts The first part would be a reasonably standard syntactic parser The only change discussed here from those parsers already on the market is that 1 would like to see the parser press ahead in the face of ungrammatically Syntax would, from time to time, hand off its results to the second part of the parser, the semantics At the moment it seems reasonable to me that this would occur at those points in the syntactic parsing process when we wish to attach a new constituent to the syntactic tree

One advantage to calling semantics when we are about to attach a constituent to the tree is that this fits nicely with the idea that semantics should be responsible for guiding the path of the syntactic parser Hence one possible action that semantics could take is the rejection of the proposed attachment However, the reader should note that 1 have ass dously avoided the issue of how this judgemental aspect fits in with the other processing semantics is doing There is an interesting system by Bobrow and Webber [2] which has some ideas on this problem, but as yet I am not happy with any of the proposals I have seen This is an area which requires a lot of work

Besides deciding on the acceptability of the attachment, semantics must do two other things, translate the English words into semantic symbols, and arrange these symbols into the proper format. One part of the translation process is the disambiguation of word senses Presumably we will use one or more of the methods mentioned earlier If we go with the discrimination net approach, which seems the safest, I assume that the nets will be "compiled" from a more declarative format

The other part of the translation process requires doing something about referent determination, another topic which I have left out from this paper. There has been a good deal of work on referent determination. [4.20.25, 19.26] but how (or even *if)* this work is to be naturally integrated into the framework I am proposing is still an open question

Lastly, semantics will take the translated symbols, and. using the clues provided by the syntax, establish the functional structure for the constituent, and indicate this structure by putting the symbols together in the appropriate semantic format For this process, the use of the lambda calculus ala Montague grammar looks reasonable

However, the output of semantics is not really the end of the parsing process. We might call the output a "surface semantics" in so far as it will get some of the slot assignments, but by no means all As wc have already noted, many of the assignments of slots in frames are best done by processes which most naturally fall under the heading of text comprehension Thus, the last part of the parsing process will be coextensive with the various recognition processes which go under the heading of text processing It is here that we will decide that the people in the airplane fill the passenger slot It may also be at this point that much of the referent determination takes place We should also note that given our decision to have semantics be a subroutine of syntax, it is an interesting question whether we should have semantics immediately call our text processing component That is. do we wait until the end of the sentence (or at least the end of an s node) before doing text processing, or, is this too intermixed with everything else My guess is the latter, but this question is yet another to be added to the list of problems which still require solutions

References

[I] Daniel G Bobrow and Terry Winograd, "An overview of KRL, a knowledge representation language." *Cognitive Science* 1(1) pp 3-46 (1977)

[2] Robert J Bobrow and Bonnie L Webber. "Knowledge representation for syntactic/semantic processing," Proceedings of the First Annual National Conference on Artificial Intelligence (1900)

f3] Ronald J Brachman, "On the epistemological status of semantic networks." pp 3.50 in *Associative Networks Representation and Use of Knowledge, by Computers,* ed N V Findler. Academic Press. New York (1979).

[4] Eugene Charniuk. "Toward a model of childrens story understanding," TR-266 MIT Artificial Intelligence Laboratory (1972)

[5] Eugene Charniak, "A framed PAINTING The representation of a common sense knowledge fragment," *Cognitive Science* 1(4) pp 3bb-39l (1977)

[6J Eugene Charniak, "A parser with something for everyone," TK 70 Department of Computer Science, Brown University (1981)

[7] Eugene Charniak, "A common representation for problem solving and language comprehension information," *Artificial Intelligence,* (forthcoming)

[8] Eugene Charniak, *"The* case-slot identity theory," *Cognitive Science,* (forthcoming).

[9] Richard E Cullingford, "Script application Computer understanding of newspaper stones," Report 11G Yale Department of Computer Science (1978).

[10] David R Dowty, Robert E. Wall, and Stanley Peters, *Introduction to Montague Semantics,* D. Reidel Publishing Company. Dordrecht. Holland (1901).

[II] Lee D Erman, Eredenck Hayes-Roth, Victor R Lesser, and D Raj Reddy, "The Hearsay-II speech-understanding system integrating knowledge to resolve uncertainty." *Computing Surveys* 12(2) pp. 213-253 (1980).

[12] Scott E. Fahlman. *NFTL: A system for Representing and Using Real-World Knowledge,* MIT Press, Cambridge. Mass (1979)

[13] Charles J Fillmore. "The case for case." pp 1-88 in *Universale in Linguistic Theory,* ed. E. Bach and RT. Harmes. Holt. Rinehart, and Winston, New York (1968)

[14] Joyce Friedman and David S Warren. "A parsing method for Montague grammars." *Linguistics and Philosophy* 2 pp 34 7-372 (1 976).

[I5] Philip J Hayes, "On semantic nets, frames and associations," pp 99-107 in *Proceedings of the Fifth International Joint Conference on Artificial Intelligence.* (1977)

[16] Philip J. Hayes, "Mapping input onto schemas." TR 29 Department of Computer Science University of Rochester (1978)

[17] Jerry R Hobbs and Stanley J Rosensehein. "Making computational sense of Montague's mtensional logic," Courant Computer Science Report //11 New York University (1977)

[18] Mitchell P Marcus. *A Theory of Syntactic Recognition for Natural Language,* M.I.T Press, Cambridge, Mass. (1980)

[19] Bonny Nash-Webber and Raymond Reiter, "Anaphora and Logical Form On Formal Meaning Representations for Natural Language," pp 121-131 in Pro*ceedings of the f\fth International Joint Conference on Artificial Intelligence,* (1977)

[20] Chuck Rieger. "Conceptual memory." in *Conceptual Information Processing,* ed R C. Schank, North-Holland Press, Amsterdam (197b).

[21] Chuck Rieger and Steve Small. "Word expert parsing." pp 723-728 in *Proceedings of the Sixth International Joint Conference on Artificial Intelligence,* (1979)

[22] Christopher K Ricsbeck and Roger C Schank. "Comprehension by computer Expectation-based analysis of sentences in context," Research Report *//78* Yale University Department of Computer Science (1976)

[23J Roger Schank and Lawrence Birnbaum. "Memory, meaning, and syntax," Yale Department of Computer Science (1960).

[24] Roger C Schank and Robert P. Abclson. *Scripts, Plans, Goals, and Understanding,* Lawrence Erlbaum Associates. Hillsdale, New Jersey (1977).

[25] Candace L. Sidner, "Towards a Computational Theory of Definite Anaphora Comprehension in English Discourse." Al-TR 537 Massachusetts Institute of Technology Artificial Intelligence Laboratory (1979).

[26] Yorick Wilks, "An intelligent analyzer and understander of English," *Communications of the ACM* 18(b) pp. 264-274 (1975).

[27] Yorick Wilks, "Philosophy of language." pp. 205-233 in *Computational Semantics: An Introduction to Artificial Intelligence and Natural Language Comprehension,* ed E Charniak and Y. Wilks. (1976).

[28] Terry Winograd. *Understanding Natural Language,* Academic Press. New York (1972).

[29] Patrick H Winston, *Artificial Intelligence,* Addison Wesley. Reading. Massachusetts (1977).

[30] Douglas Wong, "Language Comprehension in a Problem Solver." in *Proceedings of the Seventh International Joint Conference on Artificial Intelligence,* (1981).

[31] William A Woods. "Procedural semantics for a question-answering machine," *Proceedings of the Fall Joint Computer Conference,* pp 457-471 (1968).

[32] William A Woods. "An experimental parsing system for transition network grammars," pp 113-154 in *Natural Language Processing,* ed R Rustin. Algonthmics Press. New York (1972).