Incremental Evaluation: An Approach to the Semantic Interpretation of Noun Phrases

C.S.Mellish,

September 1982

Cognitive Studies Research Paper

Serial no: CSRP 001

The University of Sussex
Cognitive Studies Programme
School of Social Sciences
Falmer
[illegible]

ABSTRACT

A new approach to reference evaluation, called incremental evaluation is presented. This approach is designed to capture the notion that a phrase's referent can only in general be determined by an analysis of the global context of the phrase's use. It is also designed to support the idea that useful semantic information can be extracted from a text before a global analysis is complete. Incremental evaluation can be an effective strategy in a domain where reference evaluation is characterisable as a constraint satisfaction task. It is illustrated with examples from the natural language understanding modules of MECHO [Bundy et al, 1979a], a program which solves mechanics problems stated in English. Several possible representations for partially evaluated noun phrases are proposed and their advantages compared.

KEYWORDS

Natural Language Processing, Reference Evaluation, Incremental Evaluation, Constraint Satisfaction.

CONTENTS

1. Introduction

The research reported here is part of a long term investigation into the timing issues of natural language understanding. That is, we are interested in what constraints the structure of language imposes on the order in which a processor might effectively pursue the various subtasks and the information that a processor might have available to it at any one time. One important question that must be asked is: to what extent is it possible to extract the meaning of a piece of language as it is presented sequentially? This question bears on both the understanding of speech in real time and on the comprehension of text in a strictly left-to-right scan. There is certainly evidence [Marslen-Wilson, 1976] that people are capable of developing some degree of understanding of a sentence before it has been completed. This leads us naturally to ask how much of this is possible and what it would mean to program a computer to do it.

Traditional AI approaches to the understanding of natural language [Woods et al, 1972; Winograd, 1972] have had little to say about these questions. That is, there is no sense in which these programs, half way through a sentence, would have a partial representation of the meaning and would be able to make inferences from it. Part of the problem here has lain in the way in which semantic processing has been seen as depending on syntactic processing, which is not complete until large chunks of input have been consumed. It is in response to this view that Riesbeck [Riesbeck 1975] says:

> Why should consideration of the meaning of a sentence have to depend on the successful syntactic analysis of that sentence? This is certainly not a restriction that applies to people. Why should computer programs be more limited?

The more recent PSI-KLONE system [Bobrow and Webber, 1980] suggests that there are ways around these problems without abandoning the idea of

syntactic analysis. PSI-KLONE determines incrementally the placement of a sentence in a syntactic/semantic taxonomy as information about the syntactic structure arrives, piece by piece. At any point, the system has a partial description of the semantic "shape", on the basis of which inferences could be made.

PSI-KLONE develops semantic representations gradually as it proceeds through a sentence, but these representations are still at the "linguistic", rather than the "discourse" level. That is, PSI-KLONE can disambiguate word senses and classify structure within phrases, but it is not concerned with what those phrases refer to in the non-linguistic world. We feel strongly that understanding at the discourse level must also be investigated as part of the study of timing constraints in natural language understanding, and indeed it is on this level that we will concentrate. To narrow the scope down somewhat, we have restricted our attention to referential noun phrases and the problem of deciding what they refer to. This paper will only consider singular referential noun phrases. An extension to these ideas which covers plural phrases and quantification to some extent is to be found in [Mellish, 1981].

The problem then is: to what extent is it possible to understand a noun phrase (ie determine its referent) at the time that phrase is read (or heard)? One's immediate response to this question, given the large amount of work that has been addressed to the problem of understanding pronouns, is to say "in general, not at all". Indeed, if one insists on an "all or nothing" style of interpretation, this is the case. What we present here, however, is a new style of reference evaluation, incremental evaluation, in which partial knowledge of a phrase's referent is explicitly represented. Given such a style of interpretation, it is indeed possible to have a reliable, admittedly

partial, understanding of a noun phrase when that phrase is encountered. Moreover, that representation can do useful work in allowing other ambiguities to be resolved.

In order to contemplate partial representations of noun phrase referents, it is necessary to have characterised what kinds of information encountered during reading can affect a processor's knowledge of a referent and in what ways. Our particular implementations of incremental evaluation have relied on the conception of reference evaluation as a constraint satisfaction task. This particular characterisation was motivated by the particular domain in which we were operating - understanding mechanics problems. Although we believe that it is applicable in a much wider class of domains, it is in order to make some brief comments about the mechanics domain before we present more details.

1- *Ihl* WEC.HO project

The research described here, and the development of computer programs to investigate incremental evaluation, has taken place within the framework of the MECHO project at Edinburgh University CBundy et al 1979aD. The purpose of this project has been to develop a computer program to solve mechanics problems stated in English, and to use this as a vehicle for studying the control of inference in a problem solving system. For the purposes of this paper, it is only some aspects of the natural language modules of MECHO that will be discussed. For a more general view of MECHO, the interested reader is referred to the above paper.

Here is an example of a mechanics problem that can be successfully analysed by MECHO's natural language modules. It is taken from [Bostock and Chandler 1975]

A stone is dropped from the top of a tower. In the last second of its motion, it falls through a distance which is 1/5 of the height of the tower. Find the height of the tower.

This single example serves to illustrate some of the characteristics of mechanics problems that have been exploited in our programs. The sentences of a mechanics problem serve primarily to describe a particular "micro-world" - what objects and relationships there are within it. Hence we have concentrated our efforts almost entirely on declarative sentences. It is rare for general laws to be stated or for infinite sets of objects to be discussed, and so one can with reasonable safety make the assumption that noun phrases will be simply referential (not, for instance, generic). Time can play an important role, at least in dynamics problems, and so one must be able to handle simple time modifiers. However, we have made the assumption that a single time period or moment will serve to qualify all the time dependent relationships introduced by any one clause. This assumption has been reasonable for all the problems we have considered.

The purpose of natural language processing in MECHO is to receive a problem such as the one above and produce a representation of the micro-world described. This representation must be sufficiently precise to be used by other modules for deciding where to resolve forces, which equations to use, and so on, in order to come to a mathematical characterisation of the relationship betweens the "unknowns" and the "givens" of the problem. The representation that we use is in terms of simple Predicate Calculus assertions (with no quantifiers). The precise repertoire of predicates that we have used is not important here, but these are some of the assertions that would be generated for the above problem (slightly simplified):

```
moment(moment1).
moment(moment2).
period(period1,moment1,moment2).

stone(stone1).

rod(tower1).
tangent(tower1,tangent1).
rightend(tower1,bott1).
body_fixed(bott1,ground,point4,forever).
measure(tangent1,90,degrees).

leftend(tower1,top1).

velocity(stone1,vel1,moment1).
measure(vel1,0,ft/sec).
unsupported(stone1,moment1).

motion(stone1,top1,point5,moment1,moment2).
```

Given this style of representation, one can immediately see possibilities for producing useful semantic structures continuously during a left-right scan of the text. That is, there seems to be no reason why the database of facts should not be built up gradually as the program progresses through the sentence, rather than being generated completely at the end. For instance, after the fragment:

A stone is dropped ...

has been read, there seems to be no reason why at least some of the following could not be added to the database:

```
stone(stone1).
velocity(stone1,vel1,moment1).
measure(vel1,0,ft/sec).
unsupported(stone1,moment1).
```

Unfortunately, significant problems start to appear as soon as we look at definite noun phrases and pronouns in this context. With the phrase "a stone", we were fairly free to assume that a new object was being introduced and to introduce a new token 'stone1' to stand for the referent. If the fragment had been:

It is dropped

then we might have had more trouble identifying the referent before formulating assertions involving it.

3. Early Semantic Processing of NPs

The basic problem with trying to produce an interpretation of a noun phrase as it is read is that, in general, a noun phrase does not in itself provide enough information for this to be done. For instance, in:

> A stone is dropped from a cliff 100 m above the sea.
> ... the speed ...

the noun phrase could refer to the speed of any of the the stone, cliff and sea (at many possible times). Pronouns pose a possibly worse problem than definite NPs - in:

> A particle of mass 5 kg rests on a rough horizontal table.
> ... it ...

"it" could potentially refer to either the particle or the table, and it is impossible to decide without more context. Winograd [Winograd 1972] argued strongly for the early interpretation of definite noun phrases, motivating this by the potential for resolution of syntactic ambiguity, such as prepositional phrase attachment ambiguity. How was Winograd able to do this? The answer is that SHRDLU had no characterisation of the influence of context on definite NP interpretation and was forced to come to an arbitrary decision at the time the phrase was read. Such an approach is in no way guaranteed to come to a correct decision, especially in examples like the above. The inadequacy of Winograd's approach is shown in an excellent paper by Ritchie [Ritchie 1976]. If computer programs are to handle examples such as ours, we must progress beyond the simplistic notion that "reference evaluation" is something that can be "done" to a noun phrase and accept that:

> The notion of "referent" is something shaped by the context of a noun phrase's use, rather than being a simple function of the phrase itself.

Given this conclusion, what rival approaches are to be found? Certainly a great deal of important work has been done, for instance by [Charniak, 1972; Schank et al, 1975] to determine ways in which referents may be determined by context. Unfortunately, all of these have started from the assumption that the form of the meaning of whole sentences is available for use in the computations. They can therefore only provide indirect guidance as to how it might be possible to understand a noun phrase as it is read. In those programs that have attempted to produce more complete understanding whilst a text is being read, such as SCHOLAR [Carbonell, 1978] and IPP [Schank et al, 1980], it has been necessary to make decisions about which referents can be obtained locally and which can only be determined later. SCHOLAR postpones consideration of all but the simplest definite references, whereas IPP postpones all reference decisions until the events described are matched up with an expectation. Both of these systems require reference evaluation to be carried out on an "all or nothing" basis.

None of these existing approaches are flexible enough to account for the fact that one might have a partial understanding of a noun phrase at the time it is read, and our notion of incremental evaluation is designed to provide some of this needed flexibility. But first, we must have some characterisation of how it is that information from the context of a noun phrase's use can affect knowledge of the referent. We will be perhaps less ambitious than Charniak or Schank and restrict ourselves to a certain sort of information - information about what must be true in order for the text to be meaningful. An appropriate characterisation of this information is, we claim, as a network of constraints.

4. Constraints and Reference Evaluation

How is it that information from a text limits what noun phrases in that text can reasonably refer to? We would like to emphasise one particular way. The text is conveying a set of propositions about the world and is doing so in a particular manner. If the text is to be understandable, these propositions must make sense in the world of the reader and the manner of communication must conform with the conventions for English prose. Thus, if the reader can assume that the writer is being cooperative and does not have a wildly differing model of the world, various other propositions may be suggested to him as he reads. These propositions, which we will refer to as constraints, must all be true in order for the communication to make sense. As the reader progresses through the text, more and more constraints accumulate (forming a network). Some of these constraints mention properties and relations that referents of noun phrases must satisfy. Reference evaluation comes about, not through some special-purpose activity centred on noun phrases, but as a "side effect" of the global activity of trying to find a way in which the constraints as a whole can be satisfied.

The power of seeing problems as constraint satisfaction tasks has been demonstrated by much work in Artificial Intelligence [Sussman and Steele, 1980; Burstall, 1969; Waltz, 1972]. We have been able to capitalise on some of these results in our research on incremental evaluation.

What kinds of constraints might actually provide help in the search for referents? One could isolate many categories, and here are a few:

(1) Constraints [illegible]

•given[1] CHaviland and Clark, 1974D. If the writer has indicated that some fact is 'given[1] then the reader should be able to verify it. For example, if the writer talks about "the particle of mass 3 lbs[11] then the referent must be a particle and must have mass 3 lbs. Definite noun phrases generally provide some constraints of this kind, and this is the information that is most commonly associated with reference evaluation.

(2) Constraints that are generated because of what is physically possible in the world. If certain new information provided by the text is to be believed, it must fit in with what is physically possible. For instance, if the phrase "its ends[11] is used, then the referent of "its[11] must be some object that has ends. If somebody writes "The right end of the pole is attached to the left end[11], the referent of "the left end[11] cannot be the left end of the pole, because poles don't work that way (unless we know of a rather strange pole in this context).

(3) Constraints that arise from pragmatic considerations. For instance, one might use Grice's CGrice 1975D maxim of quality to reject interpretations of new information that are trivially true. If somebody wrote "the string is attached to its left end", then it might be reasonable to conclude that the referent of "its" cannot be the string, because the left end of any string is obviously attached to the string itself.

(4) Constraints derived from syntactic or dicourse analysis. Syntactic considerations are known to sometimes yield coreference and disjoint referent information CLangacker, 19693. For instance, the referent of "him" in "John hated him" cannot be John. It may be that

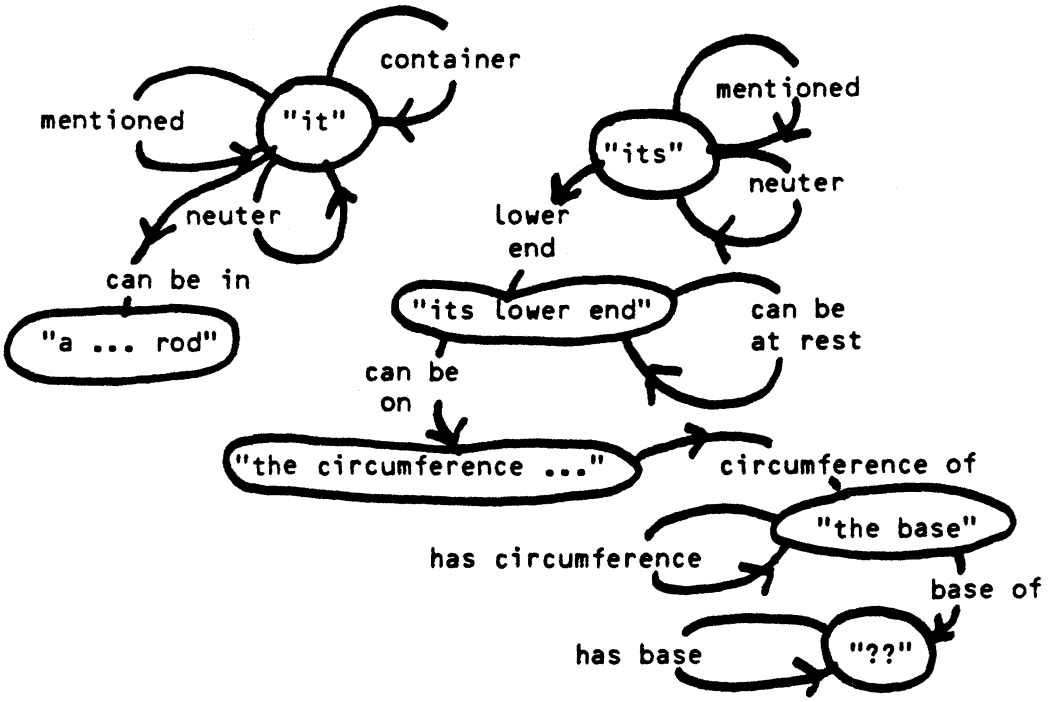
information about focus CGrosz, 1977; Sidner, 19793 could also be formulated as constraints that referents must satisfy.

In MECHO, we have concentrated mainly on the first two of these categories, but there is nothing in principle to prevent us using any of the others. The important thing to note from this brief catalogue is that information of all these kinds should be useable to help reference evaluation, but \/*ery* little of it arises directly from the analysis of a referring phrase itself. If we wish to make use of this information, a "noun phrase centred[11] approach to reference evaluation cannot be adequate.

An example will demonstrate the kinds of constraints that MECHO might generate during the reading of a mechanics problem. The following is from CLoney, 1939D:

> A hollow vertical cylinder, of radius 2a and height 3a, rests on a horizontal table, and a uniform rod is placed within it with its lower end resting on the circumference of the base.

This sentence presents several reference evaluation problems - what are "it[11], "its lower end[11], [M]the circumference[11] and "the base"? The constraints generated will mention these, initially unknown, referents, as well as referents that are known (assumed to be new objects introduced). Here is a diagram of the constraints that might have been generated by the time the end of the sentence is reached. For clarity, referents are represented informally by the corresponding noun phrases and constraints by English phrases.



Looking at part of this network, the phrase "the circumference ..." gives rise to a referent that must be in a particular relationship ("circumference of") with the referent of "the base". This in turn must be the kind of object that has a circumference ("has circumference") and be related to some unspecified object ("??") which it is the base of (there is no reason why we should restrict ourselves to dealing with mentioned "unknown objects"). In MECHO, these constraints are actually represented by Predicate Calculus assertions; that is, the "can be on" constraint would actually expand into something like:

```
not below(End,Circ)
separable(End,Circ)
solid(End)
```

Where do these constraints come from? One could imagine any semantic interpretation rule specifying constraints that must be satisfied for the interpretation to make sense. In MECHO, constraints are generated mainly by lexical entries. That is, the "meaning" of a word explicitly includes indications of what constraints must be satisfied for it to be correctly used. Here is part of our lexical extry for the verb "attach", rephrased in pseudo-English:

(Referent X) "is attached to" (Referent Y):

Basic meaning: fixed_contact(X,Y).
Constraints: solid(X) & solid(Y) & separable(X,Y) & ...

The idea is that the "basic meaning" consists of facts that will normally be added to the database (world model), whereas the "constraints" will be added to the constraint network. That is, the former will be accepted as true, but the system will have at some point to verify that the latter are true. This simple rule is changed if the "attachment" is determined to be marked as 'given' information, in which case both components of the "meaning" generate constraints.

In the "constraints" for this example, the 'separable' constraint embodies a mixture of physical and pragmatic considerations. Two objects are considered "separable" if it is conceivable that they could be moved relative to one another. If this were not the case, it would be anomalous to say that the objects were in contact, because this fact would be either obviously true or obviously false.

Here is not the place to defend the exact form of these assertions (which have in any case been simplified for ease of presentation). The essential point that we are making is that, given a developed representation of some domain, one could frequently think of plausible assertions to put in these places. It might be argued that there are other factors that should be included (such as "preferences" [Wilks, 1975]), but this does not subtract from the fact that a sizeable proportion of one's knowledge of a domain can generally be embodied in constraints. We will consider later how information derived from "expectations" (as used in "script application" [Schank et al, 1975]) can augment this constraint-centred framework.

Before we go on, it is worth making a few superficial remarks on how the

is assumed in all this that an inference system is available which is able to propose values of unknown objects which satisfy some constraint and also check whether particular objects do. This inference system must take into account general laws about the domain as well as the particular information about the world that has been gleaned so far. Given a set of constraints on some referent, the simplest way of guessing what the referent is is just to ask the inference system to find some object which satisfies all these constraints simultaneously. In the case where the only constraints are explicitly marked 'given' facts, this reduces to the paradigm reference evaluation strategy (to evaluate "the big block", find an object which is big and a block).

Such a simplistic mechanism ignores the fact that constraints are accumulating continuously as the text is read. In order to take this into account and nevertheless consider seriously the problem of understanding during reading, we must consider rather more sophisticated strategies.

5. Incremental Evaluation

Given this picture of reference evaluation as a side effect of a constraint satisfaction process, there are still important timing strategies to decide. According to this model, as a reader progresses through a text, he accumulates more and more constraints that the referents must satisfy. When does he actually decide to guess what a particular referent is? There are various possible strategies.

A first, extreme, strategy is to make a guess for a referent as soon as the first constraint involving it appears. For instance, if "it" was the first word in a sentence, some neuter object would be chosen as the referent before anything more was read. Such a strategy can be quickly

rejected (in general), because there is liable to be a large number of objects satisfying the constraint, and the wrong one could easily be picked. Such a regime would have to cope with huge amounts of backtracking. The opposite extreme would involve postponing all consideration of constraints until the end of a sentence, or perhaps paragraph. This would be contrary to our goals of early understanding. It would also mean that, during the analysis of a chunk of the chosen length, one would not have the benefit of knowledge of referents to solve syntactic (Winograd's example) or word sense ambiguity. An intermediate position between these two extremes is to make a guess for a referent as soon as the referring phrase has been processed, but we have seen that this too has problems.

Our answer to this question of timing is that there is no arbitrary point (end of noun phrase, sentence, etc) at which one can always unambiguously resolve all references. Moreover, it should be possible to use information about a referent for resolving other ambiguities even before the referent is uniquely known. This answer leads us to a system where reference evaluation proceeds gradually as the text is read. We call this incremental evaluation. Incremental evaluation involves having representations of partially evaluated referents, about which one can reason and which become gradually more defined as the analysis proceeds. There is no longer any need for arbitrary decisions to be made (although there may still be scope for certain defaults to operate).

In the following sections, we will describe two different approaches to incremental evaluation that we have tried and compare their advantages. Before we do so, though, it is useful to summarise some questions that any approach to incremental evaluation must answer:

How are partially evaluated referents represented?

How is a new constraint handled? How is it determined whether it can be satisfied, and how does it affect the state of evaluation of referents it mentions?

What happens to information added to the database if it is expressed in terms of partially evaluated references? What is it taken to mean, and what can be inferred from it?

6. Using Candidate Sets and "Waltz Filtering"

The first approach to incremental evaluation that we have tried amounts to this: we represent a partially evaluated referent by a candidate set, a list of objects that satisfy all the constraints generated so far which mention the referent. The representation of a partially evaluated referent starts off with the set of objects that satisfy the first constraint mentioning it. When a new constraint comes along, all those objects in the list which do not satisfy the new restriction are eliminated. If this means that exactly one object is left, then (in the case of a singular reference) the referent has become fully defined. If it means that all the objects are eliminated, then a contradiction has been found and some drastic measure (in our program, backtracking) is called for. Otherwise the referent still needs further specification - hopefully further constraints will do this.

From this brief description, it can be seen that this choice of representation is at least strong enough to support the syntax/semantics interaction demonstrated by Winograd. That is, if a syntactic process suggests an analysis which leads to a noun phrase with no possible referent, this will be detected and a failure generated. Moreover, this can take place at the time when the noun phrase is being read. The advantage over Winograd's system is, of course, that the system is not forced into a decision if there happen to be several objects that satisfy the description given just in the noun phrase.

It is quite possible for constraints relating together several partially evaluated references to be generated for real sentences. A constraint of this kind might be generated to embody the fact that the referent of some non-reflexive pronoun should not be the same as the referent of some other noun phrase, for instance. Some extra steps are needed to ensure that maximum information is extracted from constraints of this kind. Here, we have been able to use constraint satisfaction algorithms [Mackworth, 1977] previously used in computer vision by Waltz [Waltz, 1972] and others. The essential idea is generally known as "Waltz filtering". The following extra steps have to be taken in order that maximum use can be made of constraints that mention several partially evaluated referents at once.

(1) Whereas a constraint that only mentions one partially evaluated reference can be discarded as soon as its effect on the candidate set has been determined, these other constraints must be stored because they may need to be reapplied later.

(2) If a constraint expresses a relation between two partially evaluated references, say, candidates for one should only be allowed if there exist candidates for the other which stand in the relation to it. That is, one cannot treat the constraint as two unary constraints and consider each reference independently.

(3) Because of (2), whenever a change is made in the candidate set of one partially evaluated reference, there may be ramifications for other references that share constraints with it. Thus all these other references should be reconsidered (in the light of the stored constraints of (1)). That is, changes may propagate around the network.

To show ulcac lucao au NUI* HUM an example, here is an example of constraints being used by MECHO in the understanding of a mechanics problem. The problem is from CLoney, 1939D. In this case, the techniques of constraint propagation allow an awkward reference problem to be solved in a pleasing way.

> A uniform rod ••• is supported ... by a string ... attached to its ends.

The description of this example will have to be confined to a brief treatment of the reference evaluation aspects. The main problem here is in the referent of "its", which could initially be either the rod or the string. However, we would like to allow only the former, since it would be anomalous to speak of an object being attached to one of its ends. In order to explain how this example is treated by the program, we must first indicate how the semantic interpretation of a complex noun phrase like "its ends" takes place. We take a fairly conventional view here - that there is a rule in the grammar which says something like:

HP –> NPC+possD N

Corresponding to this syntactic rule is a semantic rule which states that the referent of the complex *HP* is the result of applying some semantic operation to the referent of the embedded NP. The exact operation is specified by the lexical entry for the noun. Thus, in the interpretation of a phrase like "its ends" there are actually two referents to be obtained - the referent of "its" and the referent of "its ends".

When the word "its" is encountered, the program sets up a new referent token, 'flTS[1], to represent the object described. There is an initial constraint on what this object can be - it must be a singular object that has been mentioned. The referent is given an initial candidate list containing all such objects, here just the rod Crod![1]) and the string

(•stringi[1]). The next manoeuvre is to obtain the referent of [M]its ends[11] by applying the meaning of "ends[11] to 'flTS[1]. This provides various constraints on what 'flTS[1] could be, which both candidates pass. That is, both are objects able to have ends. Once this has been done, two new partially evaluated references are set up to represent the two ends of the unknown object "flTS[1]. What candidates are acceptable for these at any time will depend on what candidates are acceptable for [f]£ITS'. This is expressed naturally by two new constraints:

leftend(£ITS,£LEFTEND)
rightend(£ITS,£RIGHTEND)

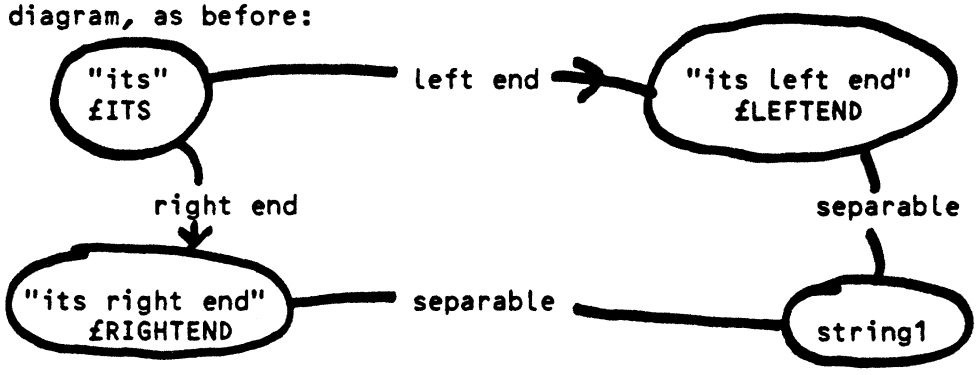
In MECHO, we represent ends in terms of a convention which labels them "left[11] and "right[11]. Here [I]£LEFTEND[I] stands for "its left end" and •£RIGHTEND[f] for "its right end". Each of these has two possible candidates (it is assumed that tokens for the ends of the rod and the string have already been set up).

The program now comes to deal with the attachment predicated between the string and the ends, which involves two attachments, one for each end. The lexical information for "attach" is consulted, firstly with •stringi[1] and [I]£RIGHTEND[I] as the objects related. This corresponds to the relationship that would be derived from "the string is attached to its right end" and will result in assertions in terms of the predicate •fixed_contact[f] being added to the system's database. However, there are various constraints that must be satisfied between two objects said to be attached; in particular, they must be "separable" (see Section 4). Hence a new constraint is generated:

separable(string1,£RIGHTEND)

In order to satisfy this constraint, the system attempts to prove this proposition with the right ends of the string and the rod substituted for '£RIGHTEND'. This succeeds for the right end of the rod, but not for that of the string, because an object is not considered separable from part of itself. Hence the right end of the string can be rejected as a candidate of '£RIGHTEND'. Now that a change has been made in the candidate set, possible repercussions must be followed up. In particular, '£ITS' must be reconsidered because it shares a constraint with '£RIGHTEND'. Filtering '£ITS' using this shared "rightend" constraint now causes 'string1' to be rejected as a candidate, because 'string1' only satisfies the "rightend" constraint if its right end is a candidate of '£RIGHTEND'; the rod now remains as the only valid candidate.

The constraints relevant to this problem are conveniently displayed in an diagram, as before:



In this example, the constraints imposed on the right end have propagated to affect a completely different referent, the referent of "its". This could not be done in conventional reference evaluation systems. Because the options for "its" are held open and later constraints only indirectly concerning the referent are allowed to propagate back, this awkward reference problem can be solved.

However, the story is not yet over. The change in the candidate set of

be considered (both share constraints with it). Consideration of '£RIGHTEND' yields no more changes, but '£LEFTEND' can now lose the left end of the string as a candidate. This change in '£LEFTEND' causes '£ITS' to be reconsidered, but no more changes are forthcoming.

With the addition of the relevant 'new' information to the database, the attachment between 'string1' and '£RIGHTEND' has now been dealt with. When the attachment between 'string1' and '£LEFTEND' is considered, the constraint

    separable(string1,£LEFTEND)

is trivially satisfied, because the only remaining candidate of '£LEFTEND' is the left end of the rod.

This description should give some idea of what adding a new constraint looks like in this system of incremental evaluation. The other important question to be addressed is: how is 'new' information about partially evaluated references to be interpreted? For instance, in the fragment:

    It is smooth ...

the understander may be in a position of wanting to record that "it" is smooth before coming to a decision about what "it" is. That is, it may want to add to its knowledge of the world a fact like:

    coeff_friction(£IT,zero)

How can this fact be used by the inference system before the candidate set for '£IT' has been reduced to one object? The answer that we have come up with is to have an optimistic inference system, which from this fact is capable of inferring that any object is smooth as long as it is a candidate for '£IT'. Hence what will be inferrable from this assertion will change (and decrease) as we gain more and more information about

infer things that are not actually true - it will be able to infer that all sorts of spurious objects are smooth from the above fact. This, however, does not matter if our only use of the inference system is to see if particular candidates satisfy particular constraints - in this case, any error will simply result in candidates failing to be rejected when they should be - a fairly harmless effect. There is no possibility of candidates being erroneously rejected, because this would have to be the result of the system failing to infer something that was true. Hence there is little penalty in allowing the inference system to be "optimistic[11] in this way. Note, however, that when constraints involve negation (and are established using "negation by failure[11] CClark, 1978D), the optimistic mode has to be switched off, to prevent effects of just this kind.

Z." Towards Intensions I Representations

There is a serious problem with the incremental evaluation scheme introduced in the last section. This is that a complete candidate set for a referent must be calculated as soon as the first constraint on that referent is generated. An immediate cause of concern is: how large will these candidate sets be, and will they become unmanageable? If one is making a list of, for instance, all neuter objects that have been mentioned in the past discourse, there may be a large number of objects to consider. An adequate characterisation of focus CGrosz, 1977; Sidner, 1979D might cut this particular problem down to size, but there seem to be other places where no such solution is available. Consider the problem of reference to times. To understand sentences describing time-dependent phenomena (such as appear in dynamics problems), it is essential to identify at what times the various events are taking place. Reference to these times may be explicit (with the use of modifiers like

"10 minutes later" or "in 1982") or may have to be inferred from some default rules (for instance, events are usually described in the order in which they occur). If we are to treat the "current time" as an unknown object, to be linked to other objects by constraints in the network, we must face the possibility that there could be an infinite number of possible "identities" that this object could have. This problem of reference when there are time dependencies has already been pointed out by Ritchie [Ritchie, 1976]. It highlights a major problem with the extensional representation of partially evaluated references developed in the last section. To handle these new situations, we must consider whether it is possible to develop useful intensional representations instead.

An alternative representation of a partially evaluated reference would be just the set of constraints that involve that referent. However, it would only be worth proposing this representation if it had some usefulness (such as the candidate sets had for rejecting false syntactic hypotheses). Moreover, such a representation would only be interesting from the point of view of incremental understanding if it could change gradually during the reading of the text until it was equivalent to the representation of a "fully disambiguated" referent.

A representation in terms of raw constraints can have some practical usefulness if we shift our attention from the idea that constraints outline the space of possible solutions and concentrate on the idea that they delimit what is not a solution. Thus we can see the set of constraints on a partially evaluated reference as representing, not a set of possible referents, but a collection of "demons" whose purpose is to reject inappropriate suggestions as to the identity of the referent. These suggestions might come from more speculative parts of the

understanding system which are trying to find interesting patterns and redundancies in the input (such as script appliers [Schank et al, 1975]). However, the question must still be answered: when should the system decide that enough constraints have built up for an actual "guess" for the referent to be made?

In fact, we have been able to subsume an implementation of constraints as "demons" and a way of dynamically deciding when to make a guess for a referent under a single mechanism. The essential idea here is that the system is built on top of a lazy inference system. Constraints are actually given to the inference system (possible solutions are asked for) as soon as they are generated, but the inference system may decide to postpone consideration of them until later. This decision is made on the basis of a meta-level analysis of the form of the constraints. If the form suggests that there can be at most one possible solution, then that solution will be sought and any partially evaluated references mentioned in the constraint will be identified as a result. Otherwise the constraint will be stored away and any partially evaluated references mentioned in it will be left as "variables" whose values are yet to be found. If at some point a script applier (or some similar program) proposes a value for some referent, or a value is found by the successful running of some constraint, then the form of those postponed constraints mentioning this referent will change (the "variable" will change to be the suggested value). As a result of this, the inferencer may now decide to seek the solution to various other constraints.

An example should illustrate how these ideas work in practice. Consider the constraints that would be generated at the start of the second sentence in the following passage:

A particle is thrown vertically upward. The particle's velocity is half of .....

The first sentence introduces one new, concrete referent, 'particle1' say. The second sentence deals with two referents, the referents to "the particle" and "the particle's velocity". Let us represent these with two tokens, '£PARTICLE' and '£VELOCITY'. These tokens can be looked apon as variables, since these should eventually be identified with existing known objects. The first constraint to be generated says that '£PARTICLE' must be a particle:

particle(£PARTICLE)

Should the inferencer try to satisfy this constraint now, or should it leave it for later? In fact, the inference system can work out that it has only been told about one particle so far, and that none of its inference rules would enable it to infer the existence of particles. Therefore the inference goal has at most one solution. The inference system can therefore conclude:

£PARTICLE = particle1

Extra constraints now accumulate to insist that '£PARTICLE' be the kind of object that can have a velocity, for instance:

solid(particle1)

(We can write 'particle1' here, because of the established identity above). This constraint is fully instantiated, and so there is no possibility of multiple solutions. The inference system can and does therefore establish that it is true. The referent '£VELOCITY' is now introduced, together with constraints:

```
velocity(£VELOCITY)
velocity(particle1,£VELOCITY,£TIME)
time(£TIME)
```

Notice that we have to introduce a new object '£TIME', because objects can have different velocities at different times. This is introduced as a variable, because it is an object whose identity must be established (one could regard it as the representation of the referent of an ommitted phrase "then"). We will make the assumption that the same time will apply to all the time-dependent relationships conveyed by this sentence, and that hence this variable represents "the current time", to be inserted in all relevant assertions generated. Now, unfortunately, there are inference rules that enable the system to create new times, and so there are many possibilities for what '£TIME' could be. Likewise, the set of possible velocities is open ended. So none of the above constraints have guaranteed uniqueness properties, and both must be held for later application.

There are several possibilities for what could happen when the rest of the sentence is read. An explicit time modifier (eg. "ten seconds later") might directly provide a value for '£TIME'. Alternatively, the events in the two sentences might be seen as parts of some larger structure (eg. a standard motion of a falling object), in which case the expectations connected with the larger structure might propose a value for '£TIME'. Or there may be no more indications, in which case a default strategy might choose the same time as the "current time" of the last sentence. Whichever happens, at some point an identification of the form:

will probably be made. In the Light of this identity, the second of the above constraints has now changed in form, and is:

velocity(particle1,£VEL0CITY,time27)

Its original import was to ask "find me a velocity and a time at which it is the velocity of particle!11. Its import now is "find me the valocity of particle! at time27M. This now has uniqueness properties, since any object only has one velocity at any given time. The system can therefore infer what the referent is at this point without introducing any search.

We have hinted at some of the criteria by which uniqueness of a constraint is suggested by its form. In fact, uniqueness is just one of a set of meta-level properties that MECHO is able to reason about to direct its inferences CBundy et al, 1979b]. Apart from a few general rules, the main source of information about uniqueness for our natural language modules takes the form of a meta-level database where information about particular predicates is recorded. This was implemented by Lawrence Byrd.

In this second scheme, what interpretation can plausibly be made of information stored in the database when it references partially evaluated references? In our program, a "variable11 like 'fPARTICLE1 is treated for the most part in the same way as any other token representing an object in the world. That is, a fact about I£PARTICLEI is seen in the same way as a fact about 'particle!1 - 'fPARTICLE1 just names a new distinct object. However, the program has an "optimistic" mode of inference just as in the first scheme. In this mode, •fPARTICLE1 can be assumed identical to any other object if this enables some proof to go through. If such an assumption is made, the "variable" will become

instantiated to the given value, and this may, of course cause various postponed inference goals (embodying constraints on possible values) to be tackled. If these inferences succeed, then the identification will be provisionally accepted (although a failure later may cause this to be reconsidered on backtracking); otherwise it will be rejected.

The "optimistic mode[11] of inference could well have been used to implement a standard "script applier[11] in MECHO, using the expectations embodied in scripts to suggest possible referents. In fact, we have conceived of expectations in a rather different way, and have attempted to recognise places where a partial description given in one sentence is then augmented with information given by another. For instance, in the sequence:

```
A train leaves Edinburgh at 11 am.
It arrives in London at 6 pm.
```

both sentences can be seen as describing the same thing - the motion of the train from Edinburgh to London. If we can realise this fact, then as a consequence we can infer the referent of "it" in the second sentence. In MECHO, the two sentences would give rise to assertions of roughly the following form:

```
motion(train1,edinburgh,£DEST,11am,£TIME1)
motion(£IT,£START,london,£TIME2,6pm)
```

where the predicate 'motion[1] relates an object, its starting and finishing places and the starting and finishing times of the motion. (We have used symbols like £START and £TIME2 here to represent "unknown objects" even when they are not referents of actually appearing noun phrases). Given some minimality assumptions about the model we are creating [McCarthy, 1980D, we would like to assume that these two descriptions are of the same motion, ie that:

```
£IT    s traini
fSTART = edinburgh
£DEST  = London
£TIME2 = 11am
£TIME1 = 6pm
```

In MECHO, we can do this by trying to "infer" the second assertion, given the first one already in the database. The optimistic inferencer will allow this, provided that any constraints woken up as a consequence of the identifications can be successfully dealt with.

8. Detecting inconsistencies

In the incremental evaluation scheme of the last section, there are some serious problems that arise from the fact that constraints are only considered individually for uniqueness properties. It is quite possible for a set of constraints on a single referent to allow only one solution, whereas the individual constraints allow many. For instance, if '£X' has to be a segment of some piece of string, then

```
leftend(£X,point34)
rightend(£X,point56)
```

only allow at most one possible value for ^{f}£X^{f}, whereas individually they allow the possibility of several. The system would postpone finding a solution for ^{f}£X^{f} whereas it should not.

A partial solution to this problem is to consider "in parallel" multiple representations of the same facts. For instance, if we use a predicate •ends[1] which relates an object to its two ends in addition to the above, we have:

ends(£x,point34,point36)

which is a single constraint with clear uniqueness properties. This method has been used to some limited extent in MECHO. It is similar to what Sussman and Steele call "shifting perspective".

A second severe problem is that one can no longer guarantee that examples like Winograd's will work in this scheme. This is because it is quite possible for the set of postponed constraints to be contradictory, and yet for this not to be noticed until a good while later. We have introduced a new way of handling a certain subclass of constraints which overcomes this problem. Unfortunately, the generalisation to all constraints is problematic.

The class of constraints that we can handle with this mechanism is <u>type constraints</u>. Assuming that the organisation of types within a domain is static and can be structured in a particular way (to be discussed), it is possible to associate with each type a fixed logical term. Given an "unknown object", such as a pronoun referent, the current knowledge of how that object fits into the various type hierarchies can also be represented by a logical term. When a constraint on the type of the object appears, this can be reflected in the object's special term by replacing the current value with the most general unifier of the current value with the term associated with the type constraint (in fact, a simplified version of unification will suffice). If the two terms are not unifiable, then the new constraint is contradictory to the information already known about the object. Hence inconsistencies (of this limited form) in postponed constraints can be detected without the necessity to actually decide on referents.

To illustrate the structure that is expected of the type hierarchies and the kinds of logical terms that are used to represent types, here are

some of the rules defining types in MECHO (simplified):

```
(1) entity(X) <--> oneof {zero_d(X),one_d(X),two_d(X),shapeless(X)}
(2) entity(X) <--> oneof {neuter(X),non_neuter(X)}
(3) entity(X) <--> oneof {whole(X),part(X)}
(4) entity(X) <--> oneof {spacial(X),abstract(X)}

(5)       non_neuter(X) <--> oneof {male(X),female(X)}
(6)       spacial(X) <--> oneof {solid(X),non_solid(X)}

(7) body(X) <--> whole(X) & solid(X)
(8) particle(X) <--> zero_d(X) & body(X)
(9) man(X) <--> male(X) & particle(X)
```

Two kinds of rules are supported by the scheme. Rules (1) to (6) specify how particular types decompose. These are all decompositions into disjoint subtypes ('oneof' is a variadic varient of "exclusive or"). A type may decompose in several independent ways (eg. rules (1) to (4) for 'entity'). The second kind of rule defines new types in terms of those already mentioned (rules (7) to (9) are of this form). Here are the logical terms representing some of these types, assuming that the above is a complete description. An isolated "_" symbol represents a variable with a different name from any other.

```
entity          -       entity(_,_,_,_)
zero_d          -       entity(zero_d,_,_,_)
one_d           -       entity(one_d,_,_,_)
neuter          -       entity(_,neuter,_,_)
non_neuter      -       entity(_,non_neuter(_),_,_)
whole           -       entity(_,_,whole,_)
spacial         -       entity(_,_,_,spacial(_))
male            -       entity(_,non_neuter(male),_,_)
solid           -       entity(_,_,_,spacial(solid))
body            -       entity(_,_,whole,spacial(solid))
particle        -       entity(zero_d,_,whole,spacial(solid))
man             -       entity(zero_d,non_neuter(male),whole,
                                      spacial(solid))
```

The essential idea here is that variables represent non-commitment about certain subclassifications, whereas non-variable values represent commitment to particular values. However, non-variable values may themselves have structure, corresponding to subtrees in the type hierarchies. A subtype may be subclassified in several ways, and the

independent subclassifications give rise to values appearing in separate components of a term. A type rule of the second type gives rise to a term which is formed by unifying the terms corresponding to the defin***. For example, the term for 'body' is the unifier of the terms for 'whole' and 'solid'. On the other hand, there is no term corresponding to the intersection of 'neuter' and 'man', because the corresponding terms do not match.

In representing types as terms of logic, we are extending some of the ideas used by Dahl [Dahl, 1981] in her database question answering programs. Our mechanism represents an extension, inasmuch as alternative subclassifications can be handled and the terms are generated automatically from the rules.

Given this underlying framework, this is what might happen to an unknown object (such as a definite noun phrase referent) encountered in a text. The object, '£REF1', say, is initially assigned a type term corresponding to the special type 'entity', ie., given the above classification, a version of:

entity(_,_,_,_)

Whenever a new constraint appears which has a bearing on the objects's type, this term will become further instantiated to reflect this. For instance, if the object was the referent of the phrase "it" then the constraint

neuter(£REF1)

would appear. As a result of this, the term would become further instantiated to:

(the result of unifying the previous term with the special term for 'neuter') This term represents our partial knowledge of how the object fits into our type classifications. It would be sufficient to reject an additional constraint such as:

man(£REF1)

because the term for 'man' does not unify with what has been built so far. Because of the constraints associated with relationships that '£REF1' is predicated as taking part in, the type term is likely to become further instantiated as the text is read. For instance, if the phrase appears in the fragment:

It has a mass of 5 lbs and ...

then '£REF1' must be a body (since only bodies have masses). As a result of the constraint:

body(£REF1)

being generated, the type term for '£REF1' will become further instantiated to:

entity(_,neuter,whole,spacial(solid))

This would be sufficient to reject extra constraints specifying that '£REF1' be the end of a rod (a 'part' rather than a 'whole') or a time period (an 'abstract' rather than a 'spacial'), for instance. It is still not completely explicit about all aspects of the object (eg whether it is one-dimensional), but the representation suffices for contradictory constraints to be detected without the necessity of actually deciding what the referent is.

This mechanism gives us a simple way of detecting inconsistent combinations of unary constraints. The main overhead is the "type term"

kept in association wixn eacn ODjecx xoKen. IT we wisnca LU e*ucnu tms to n-ary constraints, it would be necessary to keep a term for each possible n-tuple of objects! This would be quite unwieldy in practice. In addition, the above rule formats would be inadequate to define many n-ary predicates. As a result, we must conclude that our initial candidate set approach still stands as the most powerful method, as regards the detection of inconsistencies.

9. Conclusions

What conclusions can we draw from this work about the possibilities for understanding text as it is read? Initially it seemed that any general approach to deriving noun phrase referents early was doomed to failure, given the fact that information from all over the text should be taken into account in making decisions. We hope that this paper has shown that

> In domains such as the mechanics domain, representations of partially-known referents can be computed early and refined incrementally as the text is read. These partial representations can do useful work, for instance by
>
> (1) Policing syntactic decisions which have ramifications for referents (eg Winograd's example).
>
> (2) Policing suggestions from ''script applier[11] type programs which are trying to recognise higher level structures in the text.

We have presented various approaches to this incremental evaluation and compared their advantages. All of the approaches rely on a particular characterisation of how information from the text ?»ff*ets knowledge of referents. We hope that we have demonstrated that, in a class of domains, it is appropriate to see this information as taking the form of constraints. Moreover, reference evaluation should not be seen as a

special-purpose activity associated with interpreting noun phrases, b
as a side effect of a global process of establishing the reasonablene
of the message of the text.

Acknowledgements

We would like to thank Alan Bundy and the other members of the MECHO group for many ideas and fruitful discussions. This research made extensive use of the DEC System-10 Prolog system [Pereira et al, 1979], and the MECHO project was funded by grant GR/A 57954 from the British Science Research Council.

References

Bobrow, R.J. and Webber, B.L. (1980), Parsing and Semantic Interpretation in the BBN Natural Language Understanding ul System, Proceedings of the CSCSI/SCEIO Conference, May 1980.

Bostock and Chandler (1975) Applied Mathematics (Vol 2), S. Thornes, 1975.

Bundy, A., Byrd, L., Luger, G., Mellish, C., Milne, R. and Palmer, M. (1979a) MECHO: A Program to Solve Mechanics Problems Working Paper 50, Department of Artificial Intelligence, University of Edinburgh.

Bundy, A., Byrd, L., Luger, G., Mellish, C. and Palmer, M. (1979b) Solving Mechanics Problems using Meta-level Inference, Proceedings of IJCAI-6, 1979.

Burstall, R.M. (1969) A Program for Solving Word Sum Puzzles Computer Journal 12:48-51, 1969.

Carbonell, J. G. (1978) POLITICS: Automated Ideological Reasoning Cognitive Science 2, 27-51 (1978).

Charniak, E. (1972) Towards a Model of Children's Story Comprehension PhD Thesis, MIT AI Laboratory, 1972.

Clark, K.L. (1978) Negation as Failure in Gallaire, H. and Minker, J. (eds) Logic and Databases, Plenum Press, New York.

Dahl, V. (1981) Translating Spanish into Logic through Logic AJCL 7(3), 149-164 (1981)

Grice, H.P., (1975) Logic and Conversation In Cole, P. and Morgan, J. (eds) Syntax and Semantics, Academic Press, 1975.

Grosz, B.J. (1977) The Representation and Use of Focus in Dialogue Understanding Technical Note 151, SRI International, 1977.

Haviland, S. and Clark, H. (1974) What's New? Acquiring New Information as a Process in Comprehension JVLVB 13:512-521, 1974.

Langacker, R. (1969) On Pronominalisation and the Chain of Command Prentice Hall, 1969.

Loney, S.L. (1939) The Elements of Statics and Dynamics Cambridge University Press, 1969.

Mackworth, A. (1977) Consistency in Networks of Relations Artificial Intelligence 8:99-118, 1977.

Marslen-Wilson, W. (1976) Linguistic Descriptions and Psychological Assumptions in the Study of Sentence Perception, in Wales, R.J. and Walker, E. (eds) New Approaches to Language Mechanisms North Holland, 1976.

McCarthy, J. (1980) Circumscription - a form of Non-Monotonic Reasoning Artificial Intelligence, 1980.

Mellish, C.S. (1981) Coping with Uncertainty: Noun Phrase Interpretation and Early Semantic Analysis PhD thesis, Department of Artificial Intelligence, University of Edinburgh, 1981.

Pereira, L., Pereira, F. and Warren, D., (1979) User's Guide to DEC System-10 Prolog, Occasional Paper 15, Department of Artificial Intelligence, University of Edinburgh.

Riesbeck, C.K. (1975) Computational Understanding in Schank, R. and Nash-Webber, B. (eds), Theoretical Issues in Natural Language Processing, Association of Computational Linguistics, 1975.

Ritchie, G.D. (1976) Problems in Local Semantic Processing in Brady, M. (ed), Proceedings of the AISB Conference, Edinburgh, 1976.

Schank, R.C. and the Yale AI Project (1975) SAM - a Story Understander Research Report 43, Yale University Department of Computer Science, 1975.

Schank, R.C, Lebowitz, M. and Birnbaum, L. (1980) An Integrated Understander AJCL Vol 6, No 1, 1980.

Sidner, C.L. (1979) Towards a Computational theory of Definite Anaphora Comprehension in English Discourse, PhD Thesis, Dept of Electrical Engineering and Computer Science, MIT, 1979.

Sussman. G.J. and Steele, G.L. Jr (1980) CONSTRAINTS - a Language for Expressing Almost-Hierarchical Descriptions Artificial Intelligence 14, 1-39 (1980).

Waltz, D. (1972) Generating Semantic Descriptions from Drawings of Scenes with Shadows Technical Report, MIT, 1972.

Wilks, Y.A. (1975) A Preferential, Pattern-Seeking, Semantics for Natural Language Inference, Artificial Intelligence 6:53-74, 1975.

Winograd, T. (1972) Understanding Natural Language Academic Press, 1972.

Woods, W.A. et al (1972) The Lunar Sciences Natural Language Information System: final Report, Report No. 2378, BBN, 1972.