MAC TR-142

SOME PROBLEMS IN GERMAN TO ENGLISH MACHINE TRANSLATION

24

18

/

Gretchen P. Brown

December 1974

This research was supported by the Advanced Research Projects Agency of the Department of Defense under ARPA Order No. 2095, and was monitored by ONR under Contract No. N00014-70-A-0362-0006

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
PROJECT MAC

CAMBRIDGE MASSACHUSETTS 02139

SOME PROBLEMS IN GERMAN TO ENGLISH MACHINE TRANSLATION

by

Gretchen Purkhiser Brown

SOME PROBLEMS IN GERMAN TO ENGLISH MACHINE TRANSLATION

by

Gretchen Purkhiser Brown

ABSTRACT

This paper discusses some problems in the machine translation of natural language, in particular, for translation from German into English. An implementation of some parts of the translating process has been built. The system consists of a German interpretive grammar, to take in German text and output a set of semantic representations, and a generator, to produce English sentences from single semantic representations. Although based on the assumption that understanding is necessary for correct translation of text, the system does not now contain an understanding component to choose between semantic representations. The representation of knowledge and its use in natural language understanding is a research area that is already under intensive investigation elsewhere. The implementation described here is based on a systemic grammar analysis of German and English, and it applies and extends the work of Winograd. Special attention is paid to questions of semantic representation in a multi-language setting and to stylistic issues in English generation.

This report is a revised version of a thesis submitted to the Department of Electrical Engineering on January 23, 1974 in partial fulfillment of the requirements for the Degrees of Master of Science and Electrical Engineer.

ACKNOWLEDGEMENTS

My thanks --

-- to Terry Winograd, my thesis supervisor, for always asking the right questions, and for all-round help and suggestions.

-- to the members of the Automatic Programming group of Probject MAC and the members of the Artificial Intelligence Laboratory, especially to Bob Baron, Lowell Hawkinson, Dave MacDonald, Mitch Marcus, and Andee Rubin, for helpful discussions along the way.

-- to Eberhard Frey - for his insights into German, and for helping with my more mundane grammatical problems.

-- to Jacques Cohen, for extraordinary dedication to his students.

-- and to Bob, who has been patient, human, and extremely helpful, through this whole project.

TABLE OF CONTENTS

Chapter 1 -- Introduction

1.1 The Problem

The following pages describe a model of the translating process, in particular, a system designed to accept German text and produce an English translation. The model is not in any sense a complete one, especially in the crucial area of language understanding. An implementation of some parts of the model, however, has been written. In view of the considerable history of the mechanical translation problem, I should stress that the objective of this project was not to construct a large-scale working system but rather to see how far some existing programs and techniques could go in handling a group of problems that come up in text.

Does it even make sense to speak of mechanical translation as an independent problem? In the early 60's it was widely recognized that mechanical translation required the full resources of language understanding. If the translating process is strictly a matter of understanding in one language and generating in another, does the mechanical translation problem as such merit attention? I think it is fair to answer "yes" to this question. Translation seems to require the full power of language interpretation but not the full power of generation. The hardest problem of generation, deciding what to say and organizing the message, is generally not at issue in translation. So translation offers a somewhat circumscribed context within which to discuss issues of understanding, language representation, and production.

The translation problem also has the attraction of a readable output, namely the translation. The output is a criterion by which success may be measured, although of course there are a number of pitfalls here: first, it is not hard to recognize, but rather difficult to describe, what a good translation is. Second, a limited system like the one discussed here can be groomed to accept particular sentences gracefully, so that its performance cannot truly be judged either on the text it can process successfully or on the no doubt unlimited amount of text for which it will sputter and die. Once we know the scope of a particular system, however, we can use a given output, or a lack of one, as a basis for comparisons and evaluations. Through the MT problem, then, it is possible to consider some problems common to a number of areas of natural language processing.

1.2 The Evolution of the Problem

If the translating system discussed here is not in any sense a "solution" to the mechanical translation problem, it does represent the evolution that has occurred since Warren Weaver's 1949 memorandum on the subject. In its 25 year history, mechanical translation has not really been involved with a single problem, but rather there have been a series of problems, as each system that was built pointed up other areas that needed attention. Earliest attempts at MT were essentially mechanized dictionaries, doing substitutions on a word-by-word basis. Any more involved processing was left to human pre- and post-editors. When it became clear that word-by-word substitutions did not produce acceptable translations, the problem was refined to include syntactic recognition. Attention in the 50's and early 60's was centered on parsing with, for example, Oettinger's predictive analysis, or top-down, approach and the work of Yngve's group at MIT. The more sophisticated the approach to syntax became, however, the more sharply the nagging problem of syntactic ambiguity came into focus. Mechanical translation began to mean semantics as well as syntax, but it was not exactly clear what semantics meant. Three approaches represented limited answers: probabilities of lexical co-occurence could be used to select the "likeliest" parse, a group of semantic categories such as abstract-concrete could be used to restrict participants in a grammatical relation, and a fixed set of keywords could be used to choose possible interpretations and hence possible parses. However, as Bar-Hillel and others remarked, even if such methods could claim 90% reliability, they would have the disquieting property that it would be impossible to predict where the errors would occur. A systematic approach to semantics was necessary.

In recognition of the difficulty of this "new" mechanical translation problem, support for short-term, i.e. practical, projects dried up. The early 60's saw a shift away from attempts to build working systems to an emphasis on more basic research. Limited deduction came into use in systems such as that of Raphael (29) with the implication that language use included not only a static "meaning" but also a deductive ability as well. Then in 1970 with the development of PLANNER and Winograd's question-answering system, the "limited" was deleted, and more general deductive ability was advocated for natural language processing. Implicit in Winograd's system, but not clearly evident because of its question-answering nature, was also the recognition that a sentence is not really an independent semantic unit but instead part of a larger context. Eugene Charniak's detailed analysis (2) of the problems of dealing with context implies that sentence-by-sentence approaches must go the way of word-by-word translating systems. This is not to say that the sentence is not an important basic unit, but it does imply that the only system that has a chance of success at high level language processing is one that can deal with the interrelationships within text.

While we have not yet come full circle, mechanical translation is surfacing again as the name of the problem, or, more properly, as the name of one problem, since language research has diverged considerably since the inception of research on MT. Avowed translation work is underway at the University of Texas at Austin, the University of California at Berkeley (3), at Montreal, and at Stanford (Wilks, 37). At least one company, Logos Development, is marketing an MT system for English to Russian, among other languages. The system relies on posteditors with a knowledge of the source language, but the company claims a high rate of accuracy before posteditting. I could not get any detailed information about semantic processing done in the Logos III system, but from examining the company's literature, I get the impression that some sort of semantic type-checking is used.

1.3 A Sample Text

Assuming that the test of a translating system is not its ability to handle isolated sentences, but rather its ability to deal with connected text, I selected a paragraph from a paper by Hempelmann on octopuses (17). The text was used as a goal for the system, and I have tried to handle, in as general a way as possible, the types of difficulties that the paragraph presents. Choosing texts seemed much more desirable than writing them, since consciously manufactured examples often have an unnatural sound, while the most innocent-looking sample of "found" text will usually contain a number of subtle difficulties. The full paragraph is presented and discussed below, but let me first display the accomplishments of the translating system.

Ein deutlich sichtbares Zeichen für die im Nervensystem verlaufenden Erregungen ist das Spiel der Chromatophoren der Cephalopoden.

A clearly visible indication of the excitations that run through the nervous system is the play of the chromatophores of the cephalopod. The skeptical will say that this is not a very imposing output, but let me remark that a major factor limiting the output of the system is the very constrained scope of the English dictionary. Although many important problems still do remain, it would take only a relatively small amount of routine work to increase the output of the system considerably. I should mention here again that this translation was produced without a component to do understanding, so that disambiguation of word senses and marking instances of coreference was done by the user. This understanding by-pass was not done for the sake of exhibiting translation without understanding, but rather, it was in the interest of getting an output from a partial system.

Let us look at the full text considered, and discuss some of the problems that a mechanical translation system would have to face. Here is the German text, and following it is my own (hand) translation.

(1) Ein deutlich sichtbares Zeichen für die im Nervensystem verlaufenden Erregungen ist das Spiel der Chromatophoren der Cephalopoden, jener unter der Haut liegenden gelb, braun, schwarz, violett oder karminrot gefärbten Zellen,

(5) die sich entweder zusammenziehen oder durch radiär ansetzende Muskeln flächenhaft ausgebreitet werden können. Mit ihrer Hilfe vermögen sich die Tiere bis zu einem gewissen Grade der Farbe des Untergrundes anzupassen. Die das Chromatophorenspiel veranlassenden

(10) Reize werden nicht nur durch die Augen, sondern auch durch diese Farbzellen selbst aufgenommen. So werden

Kraken bei plötzlicher Zunahme der Lichtintensität ganz dunkel, auch wenn sie geblendet sind. Andererseits hängt der Zustand der Ballung oder Ausdehnung der Chromatophoren auch von den Saugnäpfen ab. Wenn diese nicht greifen, so sind im allgemeinen die Chromatophoren in Ruhe; wenn sie aber saugen, so spielen jene. Selbst die Oberflächenbeschaffenheit des Untergrundes übt, je nachdem, ob sie glatt oder rauh ist, eine verschiedene Wirkung auf die Chromatophoren aus. Eine Eledone, deren sämtliche Saugnäpfe entfernt worden sind, bleibt ständig gelbgrau, färbt sich auf Reizung aber noch dunkel (Steinach). Mit dem durch Lichtreiz hervorgerufenen Chromatophorenspiel pflegen Bewegungen der Arme einherzugehen, was sich besonders schön an Sepia beobachten lässt. Auch der Trichter pflegt dabei Wasser auszuspritzen. Ob die Cephalopoden selbst auf Farben reagieren, ist nicht bekannt. Nach von Hess sollen sie sich wie der farbenblinde Mensch verhalten. Da aber manche in der Tiefsee lebenden Tintenfische in verschiedenen bunten Farben erstrahlende Leuchtorgane besitzen, von denen man annimmt, dass sie zum gegenseitigen Sichauffinden der Geschlechter dienen, so scheint das zum mindesten für diese Formen für einen Farbensinn zu sprechen. Eingehende Untersuchungen über einen etwaigen Farbensinn der Cephalopoden sind sehr erwünscht. Neuerdings hat

Fröhlich Unterschiede in den vom Auge abgeleiteten

(40) Aktionsströmen auf verschiedene Farbreize festgestellt,

was ebenfalls sehr dafür spricht, dass diese Tiere

Farben zu unterscheiden vermögen.

A clearly visible indication of the excitations that run through the nervous system is the play of the chromatophores of the cephalopod, those cells that lie under the skin and are colored yellow, brown, black, purple, or carmine red. They can contract themselves and, again, via radially fastened muscles, be spread out under the skin surface. With their help, the animal is able to adapt to some degree to the color of its background. The stimuli that trigger the play of the chromatophores are perceived not only through the eyes, but also by the color cells themselves. So it is that cephalopods become quite dark in response to a sudden increase in light intensity, even when they have been blinded. On the other hand, the state of contraction or relaxation of the chromatophores is also dependent on the suckers. When these are not grasping, the chromatophores are generally at rest; when they adhere to something, however, then the chromatophores begin to play. Even the nature of the bottom, whether it is smooth or rough, has a certain influence on the chromatophores. An eledone whose suckers have all been removed remains a yellowish gray, although it will still go dark if given a stimulus (Steinach). Along with the play of chromatophores elicited by light stimulus, there are generally movements of the arms, which are particularly easy to observe in Sepia. At the same time, the ambulatory funnel usually squirts out water. Whether the cephalopods even react to color is not known. According to von Hess, their behavior is like that of a color-blind man. But since many deepsea dwelling cephalopods possess light organs that shine in various bright colors, and since one assumes these serve the purpose of helping the sexes find each other, then for at least these forms one would be inclined to assume a sense of color. Thorough investigations on the existence of a sense of color in cephalopods would be very desirable. Recently, Fröhlich was able to distinguish differences in the neural impulses sent out by the eye when given different colored stimuli, which again very strongly suggests that these animals are able to distinguish colors.

The choices that had to be made in producing the translation will be discussed in more detail in the chapters that follow, but a cursory look at the text reveals the following sorts of problems:

(a) SYNTACTIC AMBIGUITY: Choice of a parse can have rather striking influence on a translation. For example, line 28 and 29 can be parsed in two ways:

Ob (die Cephalopoden) selbst auf Farben reagieren...

Whether (the cephalopods) even react to color ...

Ob (die Cephalopoden selbst) auf Farben reagieren...

Whether (the cepaphalopods themselves) react to color ....

(b) PRONOUN REFERENCES are quite common in the paragraph. In line 7, to what does ihrer refer? The muscles? The chromatophores? It is our understanding of the preceding sentence, rather than some hard and fast rule, that determines a choice here. The phenomenon of pronoun reference is not only limited to personal pronouns, but includes relative pronouns such as denen in line 33, and demonstrative adjectives used as pronouns such as diese and jene in lines 16 and 18. In addition, compounds like dabei and dafür may be used to refer backward (line 27) or to refer forward (line 41).

Going from German to English, we can often get away with translating an ambiguous pronoun reference with an equally ambiguous English pronoun. When a sentence needs considerable rearranging, however, this will not always work. For example, we may have to put in a noun group where the original had a pronoun. Note also that the limited success of this approach is partly a by-product of the language pairs chosen. Going from English into German, where gender agreement is necessary between pronoun and referrent, a definite choice of pronoun referrent would be forced more often than when the direction of translation is from German into English. So while a decision about pronoun reference will not be forced in every case, a translating system must be equipped to make the decision when necessary.

(c) As if pronouns are not enough, we also have a series of NOUN REFERENCES. Line 7 uses "die Tiere" to refer back to the cephalopods of line

3, and it requires an understanding of the paragraph to determine that the two noun groups are coreferential. If the noun groups are coreferential, we can translate "die Tiere" simply as "the animals" (or, as "the animal," since this sounds more natural in English). If, on the other hand, "die Tiere" refers to animals in general, then the English would have to be just "animals."

(d) CHANGES OF TOPIC: Taking the sentence starting with "eine Eledone" on line 21, and reading the main clause of the next sentence, it is not specified whose chromatophores and arms we are talking about - the eledone's or those of any cephalopod. There are clues, but we have to understand the context to find them. Probably the easiest way to decide is to use the information in the subordinate clause: we can reason that sepia is a subclass of cephalopods, but not of eledones, so the discussion must pertain to cephalopods in general. As in (b) and (c) above, a translation can usually get by without keeping track of changes of topic. But situations will come up in which we have to make something explicit that was left implicit in the original, and, for this, knowing the topic could be crucial.

(e) Another very difficult area is WORD CHOICE. The paragraph here is a mixture of technical and common language. In general, the more common the word, the more varied its uses. For example, to translate the word Farbe from line 8 the dictionary (28) says we must choose between:

1. color, tint, hue
2. stain, paint, dye
3. complexion

Therefore, even if we know enough to untangle, say, pronoun and noun reference, we still have to know more: there are fine distinctions between different words, differences of connotation between synonyms, and issues of what words combine best with what other words.

Note that technical language is often quite a bit easier to work with in terms of word choice, since the meanings tend to be carefully circumscribed and usages are limited. This might lead one to think that a highly technical text would be a whiz for a mechanical translator. Compared to poetry, of course, it would be. Nevertheless, even the most technical writing cannot avoid prepositions, and prepositions (or their equivalent syntactic structures) are probably the most ambiguous words in any language. Word choice, therefore, is a problem that is always with us.

What these five problems - (a) through (e) - have in common is that they are unified problems in name only; each particular situation presents its own difficulties and requires its own unique solution. This, in the end, is what makes language processing so difficult. The best that a small scale research project can do is to examine a sampling of problems, with the hope that similar techniques can be used to deal with the various other cases that occur.

1.4 Limitations

Before beginning in earnest, it might be a good idea to sketch out the boundaries of the project. Some areas could have been incorporated into the project given more time, while some are major problems in their own right. So here is a list of what the system is not:

(a) The project was limited to written text and not speech. Thus, problems like intonation, fixing word boundaries, and vocal differences between speakers did not have to be touched.

(b) Also, input was assumed to be grammatically well-formed: no attempt was made to extract a message from a form that was not fully-defined for the system. Readers interested in the problems of handling ungrammatical text are referred to Levine (21), who has developed a program for grading German sentences in a classroom situation.

(c) No understanding of the text is attempted. This is because the problems of representing knowledge and using this knowledge for disambiguation are extremely complex. A considerable amount of research is underway in this area, and whatever results appear in the field of natural language understanding will have direct relevance for translation. The German interpretive grammar in the system is connected to the English generator via an understanding by-pass routine that requires human intercession. This is intended for demonstration purposes only.

(d) The system is a very poor conversationalist. The German grammar, which is the most complete component in the system, is prepared to parse questions and commands, but the rest of the system is geared strictly for declarative text. This limitation is the result of time constraints, but also reflects the focus of the project on connected written text.

The reader by now has an idea of what the system does not do, and the following pages will hopefully make it clear not only what the system does, but how it goes about doing it. I have tried to give English equivalents for German examples, so that a knowledge of German should not be necessary. Familiarity with Winograd's system (39) and LISP, however, would be useful.

1.5 Organization of the System

The implementation was written in MAC-LISP and runs on the PDP-10 Incompatible Time-Sharing System at M.I.T's Project MAC. The translator has six major components, whose functions are outlined below. It currently occupies about 180K of core, which includes the LISP interpreter. The system has only been run interpretively, but could be compiled. The major components are:

The GERMAN DICTIONARY, which contains syntactic and semantic information for the approximately 250 german words currently defined in the system.

The morphology routine called INPUT is the first pass in processing a text. Given a sentence, the input routine analyzes each word into its root and endings, then uses the dictionary to construct possible syntactic feature lists for the combination. This syntactic information, along with semantics picked up from the dictionary definition, is then associated with the sentence word.

The GERMAN GRAMMAR is written in PROGRAMMAR, a LISP-embedded language designed as a grammar-writing tool (Winograd 38). The grammar routines use the result of INPUT to construct a single parse tree of the sentence. To do this, the grammar interacts with the semantic component in an attempt to limit syntactic ambiguities by the limited semantic case checking currently implemented. Where a syntactic ambiguity cannot be eliminated immediately, a choice is made and backup is used if necessary.

When the grammar has parsed a section of the sentence, it calls the SEMANTIC COMPONENT for initial semantic processing. Semantics constructs as many representations as possible. If this number is zero, syntax is asked to reparse the section. If the number is non-null, all possible interpretations are carried forward. These semantic ambiguities will be eliminated later in the sentence either by the semantic component or by user intervention, so that only a single representation is sent to the generator.

Associated with the semantic structure are the routines of the ENGLISH DICTIONARY which, when executed, supply a set of relevant English words. Using these words along with other information available from the semantic representation, the ENGLISH GENERATOR can then construct an English equivalent for the input sentence. Output is not necessarily sentence for sentence, since the translator does have a limited facility for breaking down long German sentences into two or more short English ones.

***

The sections that follow deal with the system in approximately this order, with the exception that information about the dictionaries is distributed throughout as it becomes relevant. Chapters 2, 4, and 6 discuss the sections of the model that have been implemented, and chapters 3 and 5 discuss some issues in representing knowledge and understanding natural language.

Chapter 2 -- The German Interpretive Grammar

2.1 The Underlying Theory

The analysis of German was based on the theory of systemic grammar developed by M.A.K. Halliday (10-16). Some of Halliday's ideas on discourse have also been used, but discussion of this is deferred to section 4.9. In describing systemic grammar, I must necessarily be brief, and the reader who wants an in-depth treatment is referred to Halliday, Hudson (20), and Winograd (39).

The central precept of Halliday's theory is that language is structured to convey meaning. That is, any analysis of language cannot, and should not, divorce form from this single, overriding function. The job of conveying meaning is delegated among different syntactic _units_, of which there are three _ranks_: _clause_, _group_, and _word_. At the group level, there are noun groups, preposition groups, and adjective groups (I follow Hudson here in exiling the verb group; see section 2.6.1). The mechanism of _rankshift_ permits one unit to assume the role of another, for example, a clause may take the place of a noun group. Associated with each unit is a network of _features_, with a set of mutually exclusive features known as a _system_. These networks specify the choices available in the language. We move from one level to another in the network by satisfying the _entry condition_ of a system. Each choice made may set up certain constraints on the surface structure of an utterance by means of _realization rules_ associated with it. For example, the decision to put an adverbial (a preposition group or an adjective group) in the first position of the sentence in German means that subject and finite (i.e. conjugated) verb will have inverted order. An important point about Halliday's theory is that the choices at a given stage are not ordered with respect to each other.

Unlike transformational grammar, we are not deriving one fully-specified structure from another. Instead, as we proceed through the networks we accumulate partial information, until in the end the surface structure of an utterance is fully specified.

A network for the German clauses handled in the system appears in Figure 2.2, along with example sentences. I have tried to keep the presentation close to that of Winograd, so that a reader familiar with his English charts should have no difficulty here. The notation used in the network is shown in Figure 2.1. In Figure 2.1a, the vertical line indicates a system, and the

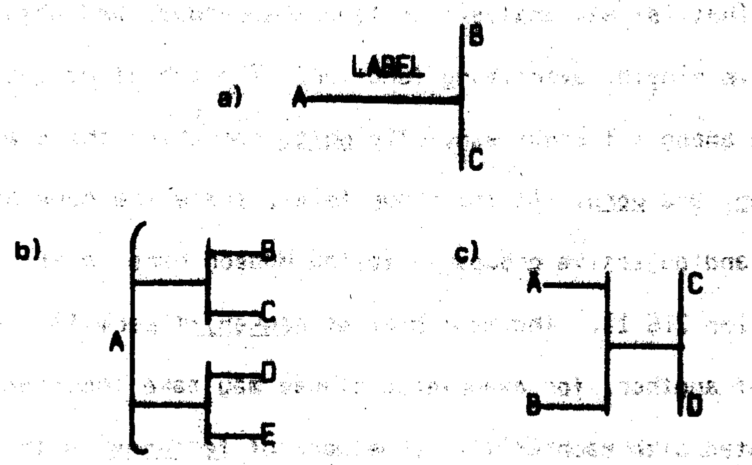
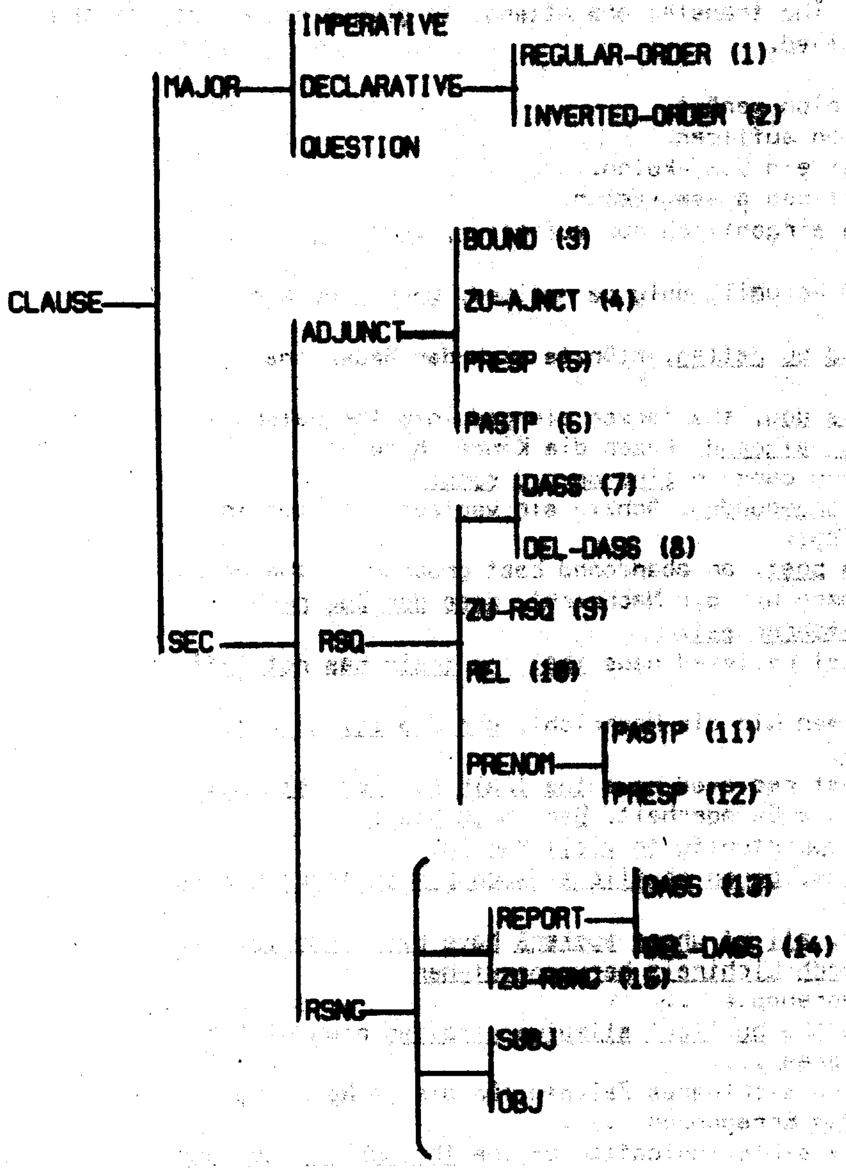


Figure 2.1

horizontal line on the left specifies an entry condition. The system may be labeled, as it is here, by writing the label above the entry condition. If two independent systems share the same entry condition, this is indicated as in Figure 2.1b. If there is more than one entry condition associated with a given system, it may only be necessary to satisfy one of them (Fig. 2.1c). Features indicated by a dashes (---) are unmarked, and are defined merely as the absence of the other features in the system.



Numbers in parentheses refer to example sentences on the following page.

Figure 2.2

The following example sentences correspond to the numbered features on the previous page. The translations attempt to give a rough idea of the structures involved.

(1) Ein Semi-kolon genügt.
A semi-colon suffices.
(2) Oft, genügt ein Semi-kolon.
Often, suffices a semi-colon.
(3) Sie würden eingentlich nur auffallen, wenn sie ausfallen.
They would actually only be noticed, when they are missing.
(4) Um den Hund zu retten, stürzte sich der Bauer ins Wasser.
To save the dog, the farmer plunged into the water.
(5) Alte Lieder singend, kamen die Kinder hinein.
The children came in singing old songs.
(6) Am Pflock angebunden, ächzte ein verlassenes Boot in Wellenschläge.
Bound to a post, an abandoned boat groaned in the waves.
(7) Eben bekommen wir die Nachricht, dass der Zug noch nicht abgefahren sei.
We have just recieved news that the train has not left yet.
(8) Eben bekommen wir die Nachricht, der Zug sei noch nicht abgefahren.
We have just recieved news the train has not left yet.
(9) Ich hatte die Gelegenheit, Berlin zu besuchen.
I had the opportunity to visit Berlin.
(10) Eine Eledone, denen sämtliche Saugnäpfe entfernt worden sind, ...
An eledone, all of whose suckers have been removed, ...
(11) Mit dem durch Lichtreiz hervorgerufenen Chromatophorenspiel ...
Along with the by light stimulus elicited play of the chromatophores ...
(12) Ein deutlich sichtbares Zeichen für die im Nervensystem verlaufenden Erregungen ...
A clearly visible indication of the through the nervous system running excitations ...
(13) Ich nehme an, dass sie schon fort sind.
I assume that they are already gone.
(14) Ich nehme an, sie sind schon fort.
I assume they are already gone.
(15) Ich bitte dich, mich morgen zu besuchen.
I request you to visit me tomorrow.

2.2 A Definition for Syntax

Distinguishing syntax from semantics is a slippery business, especially when I have just claimed to subscribe to the idea that the structure of an utterance is intimately entwined with the meaning it conveys. Still, there seems to be some life left in this very old distinction, so let me make an attempt to delimit a useful boundary. Traditionally, syntax has dealt with the form of language and semantics with its content, or meaning. This definition is a start, but it is too much of a simplification, since it ignores the effects that semantic content can have on form. In German, for example, the default adverbial ordering is time before manner before place. Here, time, manner and place look like semantic categories, but their presence has a direct effect on the surface structure of the utterance.

The whole situation becomes even more complex when one considers that most choices about the form of an utterance have what are, in terms of the traditional definition, semantic implications. It is true that there are choices that seem to be motivated by syntactic rules alone, such as the fact that the preposition ohne takes the accusative case in German, while one like bei takes the dative. (Here, and in what follows, I am using "rule" in the broad sense, to mean a regularity in language.) More common, however, is a situation such as plural formation. In German, there are several ways to form the plural for nouns, with different nouns taking -en, -e, -"e, -"er, er, etc. The choice between endings is a syntactic or morphological one (although many words are fairly idiosyncratic), but at the same time, the addition of a plural ending reflects a distinction between one and more than one that is basically semantic, in the traditional sense.

To firm up our definition, one way to go from here would be to identify the syntactic aspect of language with rules that govern sentence formation and say that semantics is the great morass beyond. This definition begs the question, however, since the content of language may be just as rule-governed as its form. For example, there might be a rule to express the fact that "The blue horse is blue" is redundant, or the fact that "The blue horse is green" is contradictory. To call such phenomena syntactic because they are rule-governed is fine; the only trouble is, however, that then it is not clear what, if anything, is left for semantics.

In building the system, my decision whether a given aspect of language was syntactic or semantic was based on the following definition for syntax. Syntactic rules give:

(i) a minimal specification of word order

(ii) a specification of the morphemic and lexical tags that explicitly mark the relationships between words and the relations within and between syntactic structures

(iii) a specification of punctuation

All the rest goes into semantics. (I am ignoring the semantic/pragmatic distinction, since I find it even more difficult to draw than the syntactic/semantic one.) Note that English syntactic rules rely very heavily on word order, while German strikes more of a balance between word order and morphology (i.e. case distinctions). With more reliance on explicit tags for its syntactic distinctions, German has a concomitantly higher degree of word order flexibility than does English. With respect to (i), the definition of minimal can really only be given in the form of a list of syntactic rules. The closest thing I have to this now is the set of word order constraints built into the interpretive and generative grammars, but of course these are not complete. To give you an idea of what I have in mind for (i), I would consider something like verb ordering in end-order a syntactic rule. On the other hand, such things as adverbial ordering (mentioned above) and adjective ordering before a noun ("the big red block" as opposed to "the red big block") are orderings beyond the minimal, and are based on semantic criteria. Relating this definition of syntax to the networks of systemic grammar, I would call a feature syntactic if its associated realization rules all fall into the three categories given above.

I realize that the definition of syntax given here is a sketchy one, but it should be enough to give a sense of the criteria that were used for deciding which regularities should be reflected in the syntactic components (both interpretive and generative) and which in the semantic component. I should add here that these criteria were ignored occasionally for the sake of efficiency, so that the interpretive grammar program is far from a pure linguistic statement.

2.3 Word Classes

While I will not discuss the treatment of clauses and groups in detail, I would like to look at classification at the word level. A description of the different parts of speech used for German words follows, while a summary of the features used in definitions can be found in Appendix A.

2.3.1 Adjectives and Adverbs

A distinction between adjectives and adverbs is especially difficult to draw in German. Where in English we have "Fido sings _well_" and "Fido is _good_," the equivalent in German is "Fido singt _gut_" and "Fido ist _gut_." To get an idea of where the distinction between adjectives and adverbs is to be drawn - or whether such a distinction even should be drawn - we need to know the functions that these units will have. Five likely-looking functions would be:

(1) attribute of a noun (declined), as in "der blaue Engel" = "the blue angel"

(2) complement, as in "Die Eledone ist klug" = "The eledone is smart"

(3) postnominal modifier, as in "die Strasse links" = "the street on the left

(4) modifier of a verb, as in "Er fährt schnell" = "He drives fast"

(5) modifier of the modifier types (1) through (5), as in "eine sehr gute Gelegenheit" = "a very good opportunity"

Especially when dealing with German, it may be better to have a single adjective-adverb category called "modifiers" and avoid the problem of where to draw the line. A highly conservative line, however, was drawn for the system. Basically, an adverb is a member of the class that performs function (5) above, but there are several conditions that further constrain adverb membership.

One criterion for an adverb that will be used is that it may never take adjective endings, that is, perform function (1). Further, an adverb may never have a comparative or superlative form. Both criteria must be present at once, since some perfectly good words that take attributive endings do not have comparative or superlative forms, and there are words that can perform both functions (1) and (5). An example of the latter is deutlich in "ein deutlich sichtbares Zeichen" ("a clearly visible indication") and "ein deutlicher Satz" ("a meaningful sentence"). Candidates for adverb, then, are words like so, sehr, zu, and a few others.

It must be obvious at this point that adverbs are a rather select group, and that the class "adjective" covers a lot of ground. To alleviate this problem, the system uses features that correspond to the functions given above. Adjectives are either ATTR (attributive, function 1), COMPL (complement, function 2), POSTNOM (postnominal, function 3), or RELMOD (relation modifiers, functions 4 and 5). Some RELMOD adjectives may take an object as a verb does, for example, "Das ist nicht der Mühe wert" ("That's not worth the effort"). If this is possible, then the case of the object must be specified in the feature list of the RELMOD adjective.

ATTRibutive adjectives are DECLined, which means that they take endings, which are specified for case, gender and number of the main noun in the noun group. Adjectives are also said to have a STRONG, WEAK, or MIXED declension, depending on the determiner in the noun group. STRONG adjective endings are used when an adjective is in first position in the noun group, for example, "guter Wein" ("good wine") or when the adjective parallels another strongly declined adjective ("feiner, lebhafter Wein" / "fine, lively wine"). WEAK endings appear on adjectives following any determiner with strong endings ("der gute Wein" / "the good wine"), and MIXED endings follow indefinite determiners, since these may or may not carry case and gender information: "einen guten Wein" (accusative), but "ein guter Wein" (nominative).

I should note that calling words that fulfill fuctions (1), (2), and (4) adjectives is in agreement with the analysis of Glinz (8), although my adverb category is more tightly constrained than his.

2.3.2 Binders

Examples of the class BINDER are dass, nachdem, seit, wenn, etc. Binders appear in the first place of a subordinate clause. Wenn/dann combinations (the equivalent of if/then) are not handled in the grammar, but they would need a special tag in the dictionary.

2.3.3 Cardinals

Cardinals (NUM), like zwei, drei, etc., occur in noun groups, and right now the parser assumes that they will not be declined.

2.3.4 Conjunctions

The coordinating conjunctions - und, aber, oder, sondern and denn - are not given syntactic features, but instead are defined using a special function. In the system, the parsing of conjunctions is done using a program taken from Winograd's system with only slight modification. See Winograd (39, p. 98 ff.).

2.3.5 Determiners

Determiners (DET) are a fairly diverse class, with the common property that they can, and usually do, occupy the first position in a noun group. Those with the feature DEF - the definites der, die, das, des, etc. - carry specification of gender, number, and case. The indefinites (INDEF) ein and kein are distinguished with respect to number (SING or PLUR) and take endings that indicate gender and case. Possessive determiners (POSS) like mein, dein and sein, specify person, gender, and number from their associated pronouns, then take endings for gender, number, and case of the main noun in the noun group. Determiners that take "der" endings are called demonstrative adjectives here (DEMADJ) and include dies-, jen-, etc. There are also the interrogative determiners (INTER) welch- and wessen. Welch- is declined for gender, number and case, while wessen takes no endings and carries no such specification.

2.3.6 Interjections

These words, like aber and ja, may appear between the subject and verb in regular or inverted order clauses.

2.3.7 Nouns

Nouns may be either MASS, COUNT, or proper (PROPN), which are relevant to whether a determiner may be used. They may be STRONG or WEAK depending on what endings are taken. For STRONG nouns, the plural and genitive endings are given in the definition. The genitive ending is necessary here because there is not enough morphological information in the input routine to derive it. To do this reliably, INPUT would have to take into consideration the number of syllables in the word and the nature of its terminal letters.

2.3.8 Participles

The PARTiciple class consists of past participles (PASTP) - like geschwommen (swum) - and present participles (PRESP) , like schwimmend (swimming). Participles are not entered in the dictionary explicitly, but the underlying verbs are. It is the responsibility of the morphology program, discussed below, to transform a verb definition into one for a participle. If a participle is DECLined, then it takes the same endings as an adjective.

2.3.9 Prepositions

Happily, this is a simple part of speech syntactically. Prepositions are either pre-fixed (PRE) as in "zu Hause," or post-fixed (POST) as in "dem Haus gegenüber." The cases they govern are either dative (DAT), accusative (ACC), genitive (GEN) or mixed (MIXED), i.e. either dative or accusative.

2.3.10 Pronouns

As in the case of determiners, there are a number of varieties of pronouns (PRON). Most common are the personal pronouns (PERS), which is actually a poor choice of terminology here, since in German they are frequently used to refer to inanimate objects. Personal pronouns are specified for person, case, number and gender (if third person). There is also a group of personal pronouns distinguished by the feature RELMOD. RELMOD pronouns like das, was, and da (when da is compounded with a preposition) may refer to whole relations or statements rather than just to other noun groups. The interrogative pronouns (INTER) like wer and was are specified for case, while the possessive pronouns (POSS) - like meine - are marked with gender, number, and person of the pronoun part, then endings indicate gender, case, and number in the pronoun's role in the noun group. Abstract pronouns (ABSTRACT) are those like man and jemand. The relative pronouns (REL) - der, die, das, deren, dessen, welche, etc. - look a lot like definite determiners and carry gender, number, and case information. Finally, DEFinite and INDEFinite determiners may also be used as pronouns, so they carry the features PRON DEM (for demonstrative pronoun).

2.3.11 Quantifiers

This class is used for words that appear in first position of the noun group but can coexist with determiners, e.g. "all die Menschen" ("all the people") and "selbst ein Cephalopod" ("even a cephalopod").

2.3.12 Separable Prefixes

In general, these coincide with the class of prepositions, and their usage corresponds to the English particle, as in "I'll call them up." Separable prefixes (SEPPR) appear in the dictionary as separate words in terms of syntactic features, but they are not given semantic definitions independent of their associated verb.

2.3.13 Verbs

Verbs come in the following varieties: First, they are either main verbs (MVB), auxiliaries (AUX), or modals (MODAL). The auxiliaries are haben, sein, and werden, and modals are those that require a main verb infinitive. Modals include können, sollen, müssen, etc.

If a verb is a MVB, it may be PLAIN, -SEPPR or +SEPPR. +SEPPR means that the verb form happens to have its separable prefix attached (abhängen), while -SEPPR means that the separable prefix is somewhere else right now (hängt davon ab). PLAIN verbs are all the rest. A verb definition also specifies whether the verb is regular, irregular, mixed, or takes an umlaut in the second and third person singular (REG, IRR, MIXED, and UML, respectively). Verbs with inseparable prefixes, like besitzen, are marked INSEPPR. This information is for the benefit of the morphology routine.

Another sort of information is the type of objects that a verb may take, which will determine the transitivity of the clause. Instead of using features like TRANSitive, I have specified transitivity in terms of syntactic case for noun groups and other abbreviations for adverbials. For example, A+D is the feature of a verb that takes an accusative and a dative object (not necessarily in this order). If a preposition is required by the verb, then its transitivity is given as "P", while if any adverbial is required, "E" is used (for no particular reason; "A" was already used for accusative). An intransitive verb is still marked "I," and "W" (for "Wemfall") is used for dative pronouns used reflexively. "Z" is used if a rankshifted "zu" clause is required, as in:

Auch der Trichter pflegt dabei Wasser auszuspritzen.

At the same time, the ambulatory funnel usually squirts out water.

I have not used categories like "required location," since this seems to be part of the broader phenomenon of semantic relations between arguments of the verb, which is handled by the semantic component. (In this case, a selection restriction would be used.) A verb with a required location, then, is marked with E, P, A+P, etc. as relevant. For a complete list of the transitivity types used in the system, see Appendix A.

2.4 PROGRAMMAR and German

A network like Figure 2.2 gives a description of structures possible in a language, but does not specify how we go about relating a given sentence to this descriptive information. One solution to this problem was given by

Winograd, and for the interpretive grammar, I followed his approach quite closely. Winograd's approach is not the only way, of course, to use the information in the networks to guide parsing, and I will come back to this issue in section 2.7.2. The German interpretive grammar was written using PROGRAMMAR, a LISP-embedded language documented in (Winograd,38). In brief, PROGRAMMAR provides facilities for constructing, inspecting, and manipulating a parse tree. The basic function is (PARSE <features>), which inspects the input sentence and tries to add the specified node to the tree. A group of feature-examining functions (NQ, HQ, CQ, ISQ, etc.) allow inspection of the sentence or tree for particular features, and another group (among them, F, FQ, R, RQ, ADD-F-NODE, and REMOVE-F-NODE) allow changes to be made in the features of a node. For moving around the parse tree there is the function MOVE-PT, while MOVE-PTW can be used to move around the input sentence. If a parse turns out to be incorrect, the backup functions POP and POPTO may be used to remove particular nodes from the tree. The basic statement type is the branch statement:

```
(:(<function or variable>) <label-1> <label-2> <label-3>)
```

Control goes to label-1 if the function or variable evaluates to non-nil, to label-2 if the value is nil, and to label-3 if the end of the sentence has been reached. Label-3 is optional.

Since PROGRAMMAR was designed to handle English, some changes and extensions were necessary for processing German. These involved the addition of another syntactic level, that of the phrase, the expansion of the apparatus that assigns feature lists to words and nodes, and a mechanism for handling partial information as it is accumulated. The rest of this section will discuss these additions to PROGRAMMAR.

As in Winograd's system, the actual parsing is done by the syntactic specialists called by the PARSE function. Syntactic specialists correspond to units, so we have CLAUSE, NG, ADJG, and PREPG. In addition, the German parser contains routines that handle phrases: that is, constituents that often appear together and for which subroutinization makes sense, but which do not enjoy the theoretical status of a group or clause. The basic difference between the treatment of groups and phrases is that a node is not established for a phrase when its routine is entered, but a group always has its own node marked with its name and features. Phrase programs are used to handle things like verbs and objects of verbs. The components of these phrases are interrelated, so we would like to handle them together. They are not full units of the grammar, however, since the components do not have to remain contiguous under all circumstances. For example, in German a direct object may appear before the finite verb, while the indirect object of the same sentence comes after the verb. No changes had to be made in PROGRAMMAR in order to write phrase routines, since they are treated like ordinary subroutines. Phrase programs look like group programs except that they do not use the reserved tags RETURN and FAIL. Modifying PROGRAMMAR to permit use of these would not have been particulary difficult, and for uniformity the change should probably have been made.

The case-gender-number combinatorics of German (described below in the morphology section) made it necessary to switch from a single list of possible features associated with each input word in the original PROGRAMMAR to multiple feature lists. Each possible usage of a word, then, is expressed as a different feature list. To handle these, a few simple changes were made in PROGRAMMAR. First, most functions now handle a list of features with an implied <u>and</u> linking the entries. Thus (NQ MASC SING) checks to see if some feature list associated with the next word has both MASC and SING properties.

It would fail, for example, if (NOUN MASC PLUR) and (NOUN FEM SING) were the feature sets in question.

Two other additions to PROGRAMMAR, FIX and NONEX, were also motivated by the proliferation of case, number, and gender possibilities. They are used for dealing with partial information and are discussed further in section 2.6.2. To discuss the interpretive process, let us start with the morphology program, since this is the first stop made by an input sentence.

2.5 Morphology

2.5.1 Analyzing Morphological Tags

Given a word, the job of the morphology component is to determine its root and then make up a list of syntactic feature sets from information associated with the root and endings. Morphology finds its information in the German dictionary, which contains both roots and endings. Syntactic information for a root is listed under the keyword FEATURES, where there is one feature set for each possible usage. Thus for breit (wide), which might appear in contexts like "die breiten Strassen" ("the broad streets") and "Die Chromatophoren breiten sich aus," ("The chromatophores spread out") the syntactic part of the dictionary entry looks like:

```
(DEFS  BREIT  FEATURES (
                (VERB REG -SEPPR AUS)
                (ADJ ATTR COMP SUP) )
```

In other words, breit may be either a regular verb that has the separable prefix aus, or an attributive (that is, prenominal) adjective that can form a comparative or a superlative. Actually, such dictionary entries contain more information, and the complete specification may be found in Appendix A.

Morphology finds its ending information as a list of feature sets indexed by ENDING, and then the part of speech. Endings are marked by ">", so there is no chance of confusing the ending >ES with the word es. (This distinction is more for the benefit of the user, since the ENDING index is enough to keep the system on the right track.) Part of the information associated with >ES looks like:

```
(DEFD >ES ENDING (
        (PRON
            (PRON* ABSTRACT* GEN SING))
        (NOUN
            (NOUN* STRONG* GEN-ES* GEN SING)
            (NOUN* MIXED* GEN-ES GEN SING))
        (ADJ
            (ADJ* ATTR* DECL STRONG NEUT NOM SING)
            (ADJ* ATTR* DECL STRONG NEUT ACC SING))))
```

There are two main routines in morphology, INPUT and TRY.

2.5.2 The Routine INPUT

INPUT is the German equivalent of Winograd's morphology analyzer. It starts at the end of a word, making successive cuts until all ending possibilities have been tried. With German we get involved in compound endings, for example when the present participle is used as an adjective, as in veranlassende (causing) = veranlass + end + e. In addition, there are some prefixes to consider, as is the case with ge in the past participle, and there are also some infixes, i.e. ge and zu, as in

ausgearbeitet (worked out) = aus + ge + arbeit + et

anzuschauen (to look at) = an + zu + schauen

The input program handles all regularly occurring morphological changes and also takes care of some non-standard situations like the addition of an umlaut to the verb lassen in third person singular (lässt). Cases that are not handled by INPUT's ending analyzer appear directly in the dictionary. These include the various incarnations of *sein* (*to be*), past participles with a vowel change like *gebrochen* (*broken*), and nouns with unusual plurals like *Sepia* (pl. *Sepien*).

2.5.3 The Routine TRY

Once INPUT thinks it has a likely split, it sends the stem and ending list off to TRY. The first step here is to check to see if the proposed root does indeed appear in the dictionary. If so, we pick up its syntactic features and hold on to them. We also pick up features associated with the different endings. It is at this point that we begin to notice some special problems associated with German morphological processing. Since an adjective ending may have four cases, three genders, two numbers, and may be strong or weak, the combinatorics begin to be a problem. And this accounting does not even take into consideration verbs, nouns, or participles some of whose endings may coincide with adjective endings. For this reason, TRY makes a first pass to determine the parts of speech possible for the root and then looks at the endings to see of they form a permissible pattern, given the part of speech. For example, *breites* (*broad*) need not be tried as a verb, since no German verb is formed by adding *es* to a root. Here, only the possibilities for an adjective need be considered, and since the endings lists are all indexed by parts of speech, it is a simple matter to pick out the relevant possibilities.

Having narrowed the field somewhat, the next step is to call the routine MERGE. MERGE moves through the endings lists, starting with the lists for the last ending and working back to the list for the root. Its job is to compound information and eliminate bad combinations. To do this, MERGE needs to know which part of the ending possibility list is required information and which is new. In the system, stars denote information that must be present in the preceding feature list and unmarked features are simply added. As an example of the matching and compounding done in MERGE, take liegendes (lying), which is a participle:

```
lieg:     (VERB IRR NVB)

-end:     (VERB*  PRESP)

-es:      (PART  PRESP* STRONG NEUT NOM SING)
          (PART  PRESP* STRONG NEUT ACC SING)
```

Matching on starred elements, and adding the unstarred information, we get:

```
(PART  PRESP STRONG NEUT NOM SING IRR NVB)
(PART  PRESP STRONG NEUT ACC SING IRR NVB)
```

These are simplified versions of the two feature sets that will be associated with the word liegendes. A special action was taken here by the routine TRY because we are dealing with a change in part of speech. For this special case, the feature VERB was removed after PART was added, to prevent the part of speech designation from becoming ambiguous in the final feature sets. This deletion is done by a simple check in TRY, since a part of speech change occurs in only a few cases.

Each call of TRY by INPUT may add feature sets to the syntactic feature list, so that in fact a word may be divided in several ways. To keep the possibilities straight, a root list is also constructed, with an entry that corresponds to each feature set.

2.5.4 Special Features of INPUT

If INPUT is not successful with its initial ending analysis, it looks for a compound word. Compound words appear frequently in German; often where English would use a classifier plus a noun, German uses a compound. The method used is a brute force one. The word is split into two parts and each of these is fed to a recursive call of INPUT. If both sections turn out to be words, we construct a feature list. If not, then we move over a letter, make another split, and try again. All sorts of refinements, e.g. only making splits between syllables, are possibilities here, but nothing like this has been done. Right now, the system handles only compounds made from two components, but the code could be generalized fairly easily. The compound analyzer will accept noun + noun pairs like Chromatophorenspiel ("play of the chromatophores"), verb + noun pairs like Leuchtorgan ("light organ"), and pronoun + infinitive used as a noun pairs like Sichaufinden ("finding each other").

Another feature of INPUT is its handling of infinitive verbs used as nouns. If a word is evaluated as an infinitive verb, a small routine is called to add features and semantics for an infinitive used as a noun. Note that in a normal text, both verb and noun would not be possible at the same time, since nouns would be capitalized. The terminal used here for input, however, had only upper case, and it seemed that any special conventions for nouns would be a burden on the user. For this reason, nouns are not distinguished from the rest of the German input, and the system just works a little harder to pick them out.

2.5.5 An Alternative

The morphology component works well, with only one real hitch: words ending in en take a considerable length of time, even with the special passes made to cut combinatorics. This may be an indication that morphology and syntax should not be a two-pass proposition, but rather that the state of the parse should be used to limit possible morphological analysis in the next word. Thus if we have "einem alten Mann" ("an old man", dative case), by the time the einem has been parsed we have gathered enough information about alten to limit it to two possibilities: either (ADJ WEAK DAT SING MASC) or (ADJ WEAK DAT SING NEUT). Therefore, to solve morphology's combinatoric problems, we might distinguish two levels of morphological features. The higher level features could be assigned in a preprocessing pass like the present one. At that point we might just specify, say, that an adjective had been found with a permissible adjective ending (ADJ DECL). Later, in parsing the noun group, a second morphological pass could check to see if the proposed adjective exhibited the correct case, gender, number, and type with respect to the determiners and other adjectives in the noun group. With this approach, the combinatorics of adjective endings need only be tackled when it appears absolutely necessary. This approach would not be difficult to implement, but it would involve considerable effort to convert the existing system to use it.

2.6 The Operation of the Grammar

The interpretive grammar operates on the string of words and feature lists output by the morphology routines. It goes to work to construct a parse tree, and at any given time the grammar will be following up only one parse. If syntactic ambiguities lead the grammar astray, special backup routines are called to find the difficulty and set the grammar onto another, hopefully more successful, path. It does not seem profitable to discuss the interpretive grammar in detail, since its behavior is so close to that of Winograd's English grammar. A sample parse may be found in Appendix B, and readers wishing further information on the approach to parsing should consult Winograd (39, chap. 5) and, for details, Rubin (32). What do seem to be worth discussing are the places where German presents special problems or where a different approach was taken, so this section will be devoted to an assortment of special topics.

2.6.1 The Demise of the Verb Group

As was mentioned previously, the German grammar has special programs for clauses, noun groups, preposition groups, and adjective groups. There is no verb group program, but there is a verb phrase routine instead. The demotion of the verb group follows Hudson (20), who argues that one criterion for a group is that its components must be contiguous. In English, of course, verbs do stick together a good part of the time, and it is easier to make a case for the existence of a verb group. In German, however, verbs are separated quite frequently, for example, in modal constructions: "Er muss früher aufstehen" ("He must get up earlier," literally, "He must earlier up get"). In fact, whenever there are verbs other than the finite verb in a major clause, German word order requires that these other verbs go to the end. Giving verbs a phrase instead of a group status does not prevent the grammar from developing the special relationships that occur between verbs in a sentence. It is the clause program, however, instead of a verb group program, that is responsible for developing these relations or calling its associated semantic routines to do so.

2.6.2 Handling Partial Information

The original PROGRAMMAR comes equipped with three mechanisms for recording information accumulated in the course of a parse. First, there is the construction of the parse tree itself. Second, there are routines to add features to a parse-node, and third, we can set and access variable-like registers associated with a node. Actually, there is a fourth mechanism, since the control structure itself is a way of holding onto distinctions within a routine. None of these mechanisms, however, automatically distinguish between information that is in some sense fragmentary and information on which further decisions can be based. By fragmentary or partial information, I mean information about choices that have been narrowed, but not fully decided. For example, the definite article *dem* in a noun group is DATive, SINGular, and either MASCuline or NEUTer. It is useful to know that FEMinine has been eliminated from the gender system as a possibility, but this is only partial information, since at this point the parser still sees *dem* as ambiguous. A parsing system should have ways to deal with this partial information easily, both to designate the information itself and to tell the parser where the partial information is.

Largely in response to the gender-number-case combinatorics, two new facilities were added to PROGRAMMAR: FIX and NONEX. FIX gives us a way of handling partial information before a parsenode is constructed. In the English version of PROGRAMMAR, the only way to specify the features required in a word is to use these as parameters of the actual function call of PARSE. With FIX, we can eliminate possible parses of the next word as the relevant information is encountered. For example, (FIX MASC SING) moves any feature sets that are not both MASC and SING to the back of the list. Similarly, we can say (FIX OR MASC NEUT) , which exiles feature sets that are not either MASC or NEUT. Feature sets that have been eliminated are put behind a marker so that they are no longer accessible to the PARSE function, although it is easy to recover old possibilities by erasing the marker. FIX may be used several times, then, to narrow the possibilities before PARSE is finally called. One facility that FIX could have but does not right now is FIX NOT, a way to disqualify feature sets that contain the feature given.

One place where FIX is used frequently is in dealing with verb phrases. Consider the example:

Eine Eledone, deren sämtliche Saugnäpfe entfernt worden sind,...

An eledone, whose suckers have all been removed,...

After parsing the past participle worden, we know that the next word must be a form of sein and either an INFinitive or a finite verb. We can use FIX to record these facts, and then check to see if the verb is finite. (All but first and third person plural of most verbs can be unambiguously distinguished from the infinitive form.) Since it turns out that sind is a finite form, it should agree with the subject, so we can call FIX again with the person and number of Saugnäpfe: (FIX P3RD PLUR). Note that the addition of FIX does not give PROGRAMMAR any new power, since we always could have done the same sorts of things by setting variables to be used later in the call to PARSE. FIX merely makes it easier to accumulate information about the next word to be parsed, even if this information is found in widely scattered parts of the grammar.

NONEX, for non-exclusive parse, allows the PARSE routines to live with ambiguity, at least to a limited extent. When we make a call like (PARSE ADJ DAT SING NONEX), we are saying, "Eliminate feature sets that don't agree with the features specified, but if more than one feature set is left (say, sets with different genders), don't worry right now." In the grammar, NONEX parses are used within noun groups, so that we are not forced to make case, number, and gender distinctions on adjectives and determiners before the necessary information is in. In the noun group, the ambiguity will only rarely persist beyond the point where the main noun is parsed. Code is built into the noun group specialist to allow us to go back and clean up NONEX parses (i.e. pick the correct feature set) when it is possible to do so. Another place that the noun group specialist uses NONEX is in parsing pronouns. We may not know the gender or number of a pronoun until its referent has been found. (We may also not know the case, but that is handled by backup instead; see the next section). NONEX allows us to do the best we can with a pronoun, maybe using subject-verb agreement to limit the possibilities. When the referent is found, a process usually not done until the end of the sentence, we can clean up the pronoun node.

FIX and NONEX, then, make it easier to handle partial information. They do not, however, really come to grips with some of the deeper problems of parsing uncertainty, which are discussed in section 2.7.1.

2.6.3 Objects of the Verb

While the main verb in English almost always precedes its objects, in German this is much less often the case. As was mentioned above, whenever there is more than one verb in a major clause, all but the finite verb go to the end. In addition, most secondary clauses are end-order, i.e. all verbs are at the end. Thus, we frequently find ourselves confronting objects of the verb with no inkling of what the main verb is. To further complicate the situation, the ordering of different objects may depend on whether one or both of them are pronouns, and what sort of pronouns they are. The situation is complex enough, I think, to force a factoring of the problem. To this end, I have made two distinctions, one of which seems successful, while the other seems less satisfying.

Let us first consider the more successful measure. Object parsing was divided into two passes, the VERB-OBJECTS routine (a phrase, rather than a group, routine) and the WORD-ORDER-CHECK routine. VERB-OBJECTS finds noun groups, preposition groups, etc, and then WORD-ORDER-CHECK decides whether they are in the correct order. This division of labor had nothing to do with efficiency or a vision of "what people do." It was merely an attempt to avoid a rat's nest of complex programming. Ideally, the search for objects and a check on their word order should probably be done in parallel, but the only disadvantage I can see in the way I have factored the problem is that it might take a little longer to reject a bad parse on ordering grounds.

A second aspect of the grammar's handling of objects is the design decision that the case of a noun group is a higher level feature than its gender or number. That is, as it stands now, the noun group specialist must always be called with a case specified. My motivation for doing this was that at a given point in the parse, noun group case is often predictable. This was a bad decision, since if we make the call (PARSE NG ACC) and if there is a dative noun group rather than an accusative one, the noun group specialist will fail, never knowing what it missed. The alternative is to permit NG to be called without case specification and to have it parse any noun group, reporting back the case that it finds. This would be a relatively simple change to make, although we could get into trouble with possible ambiguities between nominative and accusative cases for neuter singular, feminine singular, and plurals. As it stands now, it would take some extra mechanism to handle this, but a simple change to allow negative specification of a parse (i.e. "parse anything but a nominative noun group") would be sufficient.

Allowing the noun group specialist to be called without case specification would improve the efficiency of VERB-OBJECTS and several other routines, but it would not change the structure of VERB-OBJECTS greatly. As it stands now, VERB-OBJECTS is called both when the main verb has been found and when it has not been. The routine works from a shopping list found in the register FILTER. If the main verb has been found, then FILTER is a list of the sorts of transitivity types of objects it may take (These were described in section 2.3.13). If no main verb has yet been found, the main clause program can usually tell whether we are looking for active or passive object types and set FILTER to these. In something like a secondary clause, FILTER is set to all possibilities, of which there are currently 24. Note that this is not as inefficient as it seems, since any attempt at parsing a set of objects will give us information that can be used to update the possibilities in FILTER. For example, FILTER is always ordered from longest to shortest, with noun groups considered longer than prepositions (i.e. A+A before A+P before A before P). Thus, if at any point we fail to find a noun group for a second object, we can eliminate all double noun group types from FILTER. With the exception of the problem with noun group case specification mentioned above, parsing of verb objects proceeds in an orderly and fairly efficient fashion, even when the main verb has not been found.

2.6.4 Limiting the Parse

When I started this project, I naively thought that parsing German would be a simpler matter than parsing English. The reason for this belief was the very thing that gave the morphology component such a headache: the abundance of case, gender, number, and person distinctions. Natural languages, however, are very finely balanced. The German syntactic components are carefully tagged because word order has a much wider degree of flexibility. I already knew that objects of the verb have more freedom in German than in English: "Den Mann kenne ich" is perfectly fine ("I know the man," literally, "The man know I"), even "Dem Mann gab ich es" is not surprising ("I gave it to the man," literally, "To the man gave I it"). What gave the parser the most trouble as it was being developed were things like the possibility of post-fixed, as well as pre-fixed, preposition groups, adjectives that take preceding noun groups as objects, clauses appearing prenominally, post-fixed genitives that are marked by case but have no helpful of marker as in English, etc., etc. In this section, I would like to run through a list of the methods used to keep the parser on the track. Some of these measures were more desperate than others, but I tried to at least handle them in a uniform manner. Note that one other very important mechanism for limiting the parse is semantics, which is discussed in the next three chapters.

The first set of methods for limiting the parse comes built into PROGRAMMAR. It is easy to move around the parse tree and to interrogate the next word in the sentence about its features. The CUT variable can be set to prevent parsing from going past a certain point in the sentence. Finally, message variables can be set, so that reasons for a failure can be recorded. One way message variables were used throughout the system was to prevent a second parse attempt when the first one had already failed. For example, if we call (PARSE NG ACC) and an accusative noun group has already been unsuccessfully attempted at this point, the noun group routine returns failure without any reparsing. Although there is some overhead in setting up a node (here the noun group node) when a PARSE call is made, checks at every calling point would be cumbersome. Therefore, the individual syntactic specialists were made responsible for checking the failure list.

In addition to the use of message variables, the programs do some look-ahead in the sentence. I am not sure whether to be happy with this approach or not. There is nothing inherently wrong with look-ahead in text processing, but I suspect that it should be more fully integrated into the design of the parser (see section 2.7.2). Whatever the case, the look-ahead used was simple and reasonably effective. An example is the prenominal clause like "die im

Nervensystem verlaufenden Erregungen" ("the excitations that run through the nervous system.") If there is no present or past participle anywhere in the sentence, a call for a PRESP or PASTP clause will use look-ahead and fail immediately. If this were not done, we would probably end up parsing the main noun of a noun group as a verb object and go to a lot of trouble before it became clear that no prenominal clause was present. Since prenominal clauses can occur in just about any full noun group, this could slow down the system considerably. I should note that the look-ahead mechanism here could break down in very long and complex sentences, where a lot of different syntactic structures are present. As it is now, if there is a member of the word class anywhere in the sentence - even twenty words away - the look-ahead will be satisfied and a parse will be attempted. This rudimentary look-ahead, then, is not a panacea, but it does allow the parser to take simple actions for simple sentences.

Two other parser-limiting mechanisms are more conventional. First, at the beginning of the more complicated routines, entry conditions were set up. If the next word's possible parts of speech do not match these in the entry condition, failure is immediate. A second measure is to distinguish three levels of noun group: FULL, SIMPLE, and NO-RSQ. When PARSE is called for a FULL noun group, it is free to try anything. SIMPLE noun groups exclude rankshifted noun groups (RSNG - like "Ich weiss, <u>dass</u> <u>es</u> <u>wahr</u> <u>ist</u>" / "I know <u>that</u> <u>it</u> <u>is</u> <u>true</u>.") Finally, NO-RSQ noun groups cannot be RSNG and may not have rankshifted qualifiers. Verb objects in major clauses are parsed with FULL, while objects of prepositions, for example, are parsed with SIMPLE. NO-RSQ is used primarily within prenominal clauses to prevent embedding. (Actually, we might want to allow embedding to one level of PRESP and PASTP clauses. This would be a simple change.) The noun group distinctions save time and parser effort, and I do not feel that they seriously curtail the generality of the grammar. Multi-level embedding strains human comprehension, and it is fair to treat any embedding of like elements beyond a certain level as a pathological case.

Finally, if all else fails, there is backup. The routines used in the system will be described, but I should note that the backup mechanism used is not the best possible one. Interested readers are referred to Hill (19) for an analysis of backup methods and a description of "implicit backup." While the backup routines in the grammar were written only as the need arose, they do handle some of the more common hazards. The four routines are VERB-BACKUP, TRANS-BACKUP, SUBJ-BACKUP, and SEMI-CL-BACKUP. The first, VERB-BACKUP, is specific to the system. Its job is to rescue verb infinitives that were mistakenly parsed as nouns - a problem that would not arise if the system distinguished between upper and lower case letters (since nouns are capitalized in German). TRANS-BACKUP is called if the main verb is parsed after its objects and if the transitivity possibilities of the verb do not match the objects found. This routine pops nodes from the tree and sets the necessary registers for another call to the VERB-OBJECTS routine.

SEMI-CL-BACKUP takes action if the objects of a semi-clause have been parsed but if there is no participle or *zu* plus infinitive to be found. The routine checks to see whether the verb was absorbed by one of the objects of the verb. If so, it pops the objects, sets a cut point at the verb, and returns control to the CLAUSE routine for another try. The last backup routine, SUBJ-BACKUP, specializes in those German inverted order clauses in which the finite verb is the main verb. Since the subject follows the verb in inverted order, there is nothing to prevent a subject from absorbing preposition groups and objects that belong to the clause. (The same is true of secondary clauses, but this is handled by TRANS-BACKUP.) Semantic checking helps some, but often it is not until the parser actually tries to parse objects that it discovers that something is wrong. As the system stands now, a subject noun group will not mistakenly appropriate a preposition group (for an explanation of this, see the next section), but it could pick up a verb object, thinking the object was a genitive. It is the job of SUBJ-BACKUP to intercede if the main verb comes up short of objects, and to check to see whether the subject is holding on to more than it should. Object problems with inverted order clauses whose finite verb is not the main verb are the same as for secondary clauses and are handled by TRANS-BACKUP.

2.7 Problems and Observations

2.7.1 A Sticky Problem and a Partial Solution

One difficulty with using backup for natural language parsing is that, at any given point, it is not always clear whether backup should be initiated. In this section I would like to discuss a problem which occurs in English, but which appears in much more florid form in German. The solution proposed does depend on backup, but it attempts to minimize the instances where incorrect parses will remain undetected.

Consider a secondary clause from our example paragraph:

Da aber manche in der Tiefsee lebenden Tintenfische in verschiedenen bunten Farben erstrahlende Leuchtorgane besitzen, ...

But since many deepsea dwelling cephalopods possess light organs that shine in various bright colors ...

The structure of this secondary clause is binder-interjection-subject-direct object-main verb, and the correct division between subject and direct object is after <u>Tintenfische</u>. A program parsing along blindly, however, could appropriate the preposition group "in verschiedenen bunten Farben" as a qualifier for the subject Linienfläche. Syntactically, the parser would never be the wiser, since "ausstrahlende Leuchtorgane" is a perfectly good accusative noun group. Semantically, too, it would take a fairly sophisticated program to know that something was amiss.

In an attempt to avoid this sort of situation the noun group specialist is cautious, and, whenever a noun group could absorb preposition groups or adjective groups (as ADJG POSTNOM) that do not belong to it, it will refrain from doing so. Such preposition groups, etc. will then often end up bound to the major or subordinate clause node. This is not a bad fate, because it is often the correct one. Since academic German makes such frequent use of the prenominal clause, it is less likely that simple relations will be expressed with postnominal qualifiers. If, on the other hand, this is the wrong decision, we are in a better position to find that out. Since major or subordinate clauses tend to be more fully specified than their component noun groups, even a relatively weak semantics component might be able to determine that a clause has something extra, even though it might fail to do so for a noun group.

Note that in the example above, assigning "in verschiedenen bunten Farben" to the subordinate clause would be incorrect, since it really belongs to the prenominal clause modifying the direct object. When the semantic specialist for the clause, SMCLAUSE, is called, it will presumably decide that this preposition group cannot modify the relation represented by the verb besitzen and transfer control to a backup routine. Because of time limitations, this backup code has not been written, but its basic task would be to detect jurisdictional disputes. We already know that the preposition group does not modify the clause, but we have to check to see whether the components on either side want to claim it. With some simple syntactic information - the entry conditions for different structures and the ways they may terminate - we could eliminate certain claims on the preposition group. In the example sentence a simple syntactic check is not enough to decide whether the subject or the direct object should get the preposition; so we would have to depend on the fuller syntactic check of a parse attempt, with the accompanying semantic checking. The backup code could call for the two different parsings, which in effect would allow the semantics programs to direct the parse. This scheme is similar to one outlined by Woods for English (48); the difference is the heuristic that assumes that modifiers belong to the dominant clause unless proven otherwise.

2.7.2 Other Approaches to the Problem

When I began writing the interpretive grammar, the question I was asking was how to adapt Winograd's approach to handling German. SHRDLU is not the only way, however, to translate information like that in systemic grammar networks into a parser. In the time since Winograd's system appeared, there has been some interesting work on English parsing, notably that of Martin (24) and of Marcus (23). Martin's approach takes the form of a parser similar in spirit to Wood's transition networks, but which always expands all possible parses instead of attempting to choose the most likely one and backing up. Martin contends that semantics will limit the number of possibilities at any given point, so that combinatoric explosion is not an issue. Marcus proposes a "wait and see" approach, with decisions delayed until the necessary information is in. The ability to delay parsing decisions would be particularly useful for German; it will therefore be interesting to see the results of this work. The merit of these approaches is that they avoid backup with its accompanying problems of programming complexity and the possibility of overlooking ambiguous cases.

Chapter 3 -- Ordering Concept Markers

3.1 The Conceptual Structure

Considering language as a "structure organized to convey meaning," we have discussed the structure part to some extent and now will turn to the question of meaning. By the meaning of a word, I mean the group of entries in the conceptual structure associated with that word. But this, of course, means that we now need definitions for "entries" and "conceptual structure." Another word for entries here is concepts, and it is an open question just how concepts should be embodied. Should we use rules, procedures, images, some combination of these, or something else altogether? Should the organization of the conceptual structure be a net with no constraints on linkage, or a highly structured hierarchy, or, again, something else? Since no component to do understanding has been implemented here, my answers to these questions will have just enough specification to motivate other choices that must be made in the system.

For the proposed deductive component, meaning can be defined in terms of certain symbols, data structures, and programs. Entries in the conceptual structure are theorems and assertions in a deductive programming language like Planner (18) or Conniver (25). For the most part, these are either directly or indirectly associated with entities called concept markers. Within the system, the concept markers are primitives: assertions in the deductive data base are built from them and, as mentioned, theorems are associated with them. It is easy to imagine another level in the system with, say, visual images. The concept markers would then point to these images, and procedures would be available to do operations on them like inspection, updating, simple manipulations like rotation, etc.

The chapter that follows is concerned with the way concept markers should be ordered, and chapter 5 goes into more detail on the kind of processing we would expect from a deductive component. The ideas discussed here are not original, but I think this chapter, and chapter 5, will be helpful in describing some of the issues that have to be considered if the other parts of the system are ever to interact successfully with a deductive component.

3.2 Objects, Relations, and Properties

As in Winograd's system, the things in the world will be represented as objects, relations, and properties. Note that these three categories will be used from now on only as distinctions within the conceptual structure, not to name things in the real world. To an extent, I am willing to consider these three categories as primitives, since it is difficult to come up with water-tight definitions. As an attempt, however, let us say that the process of creating a conceptual object effectively differentiates the part of the world to which it corresponds from the rest of experience (with experience and world viewed broadly - not just reality but imaginary experiences, etc. as well). An object, then, points to (or, less metaphorically, represents) anything viewed statically, that is, any phenomenon considered as an entity. An object might be a physical object like #TRUCK, an institution like #CITY-HALL, a mental phenomenon like a #DREAM, some abstract entity like #TRUTH, etc. The sharp sign here is used to distinguish concept markers, which are part of the system's internal conceptual structure, from plain words, which are part of language.

A relation binds one or more entities, the exact type of linkage depending, of course, on the type of relation involved. The entities bound may themselves be relations, may be objects or properties, or may be some combination of the three. One common type of relation is an event; events always involve a specific point or span of time in addition to their other participants. Properties are relations with no arguments. This is the only way that properties distinguish themselves from other relations, but the distinction seems to be a useful one both conceptually and for purposes of implementation. Properties are used to describe and modify objects and relations, as well as other properties. To represent this, the system has a special relation (called, not surprisingly, #HAVE-PROPERTY), which links properties with the concept modified.

Our three conceptual categories correspond roughly to the jobs of different syntactic structures: noun groups often refer to objects; clauses, preposition groups, and some adjective groups represent relations; and most complement and attribute adjective groups represent properties. This correspondence should not be taken to imply that objects, relations and properties are derivable from syntactic categories. The conceptual structure is an independent level; in fact it is not surprising to see high level semantic distinctions reflected in the syntax of at least English and German, and probably all languages.

3.3 Selection Restrictions

Considering the huge classes formed by classifying the world into objects, properties and relations, it seems likely that particular elements will be more useful and accessible if the classes are structured in some way. Since any choice of structure should be influenced by function, let us look at the way the system will want to use information about the world. Basically, there are two kinds of activities that we want happening in the system. One is the use of semantic selection restrictions to eliminate incorrect interpretations. The other is the use of the full range of real-world information to make a final choice between interpretations, including determining pronoun reference, etc. These two functions make different demands in terms of the kinds of knowledge structure that each can use most easily, so a decision on structure requires a closer look at semantic restrictions and deductive interpretation.

Selection restrictions indicate the outer limit on what types of concepts may appear in a relation together. For example,

Das Stück Seife ist liebevoll.

The bar of soap is affectionate.

is an odd sentence because affection is an attribute of humans, maybe of animals too, but it certainly can not be an attribute of a non-living thing. (This ignores, of course, personification, but this phenomenon, as well as any sort of metaphorical speech, will not be considered now. See section 5.8 for a discussion of some of the issues involved.) From this example, we want the selection restriction associated with affectionate to be ALIVING-THING. Such a selection restriction gives us a criterion for rejecting bad parses. If the semantic component ever finds itself trying to link affection to soap in a straight scientific text, it will find no possible meanings of <u>liebevoll</u> that will satisfy the selection restriction. Semantics will then fail, which will cause the syntactic component to try another parse.

In addition to rejecting a parse because no meanings of a word are acceptable, the semantic component could also use selection restrictions to eliminate possible meanings of a word. This does actually happen, but it is a rather tricky business. For example, we might expect that selection restrictions could help us out in distinguishing between:

(i) Der Film _spielt_ heute Abend.

The film is _playing_ tonight.

(ii) Unter Lichtreiz _spielen_ die Chromatophoren.

The chromatophores _play_ when stimulated by light.

For the meaning of the verb _spielen_ in (i) we might specify that the first argument of the relation should be something like #THEATRICAL-PRESENTATION, i.e. a film, play, puppet show, cabaret, etc. It would not be unreasonable to expect the semantic markers for these phenomena to be classified under #THEATRICAL-PRESENTATION. For (ii), however, it is hard to say what the first argument might be. The lights on a marquee can play in this sense, the Northern Lights can, chromatophores, of course, do - even sounds can. The unifying characteristic here seems to be that these things form a system, whose individual members perform their particular activity (flash, emit a sound, move etc.) in an apparently (to the perceiver) random order. This is obviously not a simple characteristic, and probably the best we can do for a selection restriction here is something very general like #CONCRETE. Since a thing that is a #THEATRICAL-PRESENTATION would also be classified under #CONCRETE, if we are given sentence (i), selection restrictions alone will not be able to tell us whether the film is running at the theater, or whether we should expect to see it flashing on and off. For this reason, although the selection restrictions in the system do resemble the semantic markers of Fodor and Katz (6), they serve a different purpose. In this system, the selection restrictions are not expected to give a full accouat, or even the major part, of the meaning of a word. Something as simple as a selection restriction will not be able to represent semantic constraints as to which participants may take part in which relations with any degree of accuracy. What a selection restriction can offer is a negative criterion for eliminating impossible interpretations. In this way they can be quite useful for giving feedback to the parser, and sometimes, but not always, they can be helpful in eliminating impossible interpretations of a word.

More detail on how selection restrictions are used will be given in the next chapter. Based on the discussion so far, however, we might predict that selection restrictions can be made to work most efficiently with a strictly hierarchical structure. A single tree has been used for selectional restriction processing, with the relation between levels on the tree the general one of class membership. Note that a single classification scheme is not the only choice for selection restrictions. Multiple trees are a possibility, as in Winograd. Operations on a single tree, however, can be done with a minimum of time and effort, and, since the reason for using selection restrictions is efficiency, it seems to be the best choice.

The selection restriction tree should be organized to eliminate the largest number of interpretations as often as possible, but beyond this goal the organization issue seems to be a question of balance and a matter of taste. Given a constant number of concepts, less depth means a faster search, more depth means that the selection restrictions can make finer distinctions. The feature that is essential here is the hierarchical structure. Thus, our tree might have a top node labeled #WORLD, whose three descendents are #OBJECT, #RELATION, and #PROPERTY. The upper nodes of the relation subtree are shown in Figure 3.1. Here, #STATIC-RELATION includes things like spatial relations, relations of comparison (X is similar to Y), etc. #MENTAL-PROCESS includes intellectual, emotional, and perceptual processes. (This organization is taken from Halliday (16).) Figure 3.2 shows the upper nodes of the property subtree. The #MANNER-PROP, #SPATIAL-PROP, and #TEMPORAL-PROP classes contain

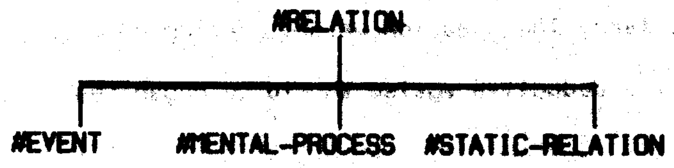


Figure 3.1

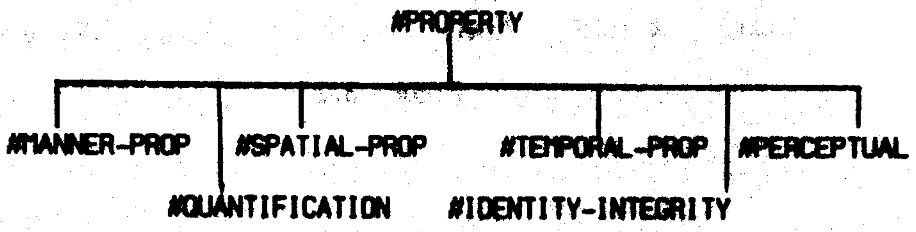


Figure 3.2

properties that answer the questions "how", "where" and "when". #QUANTIFICATION deals with extent, which is either number, for objects, or intensity, for relations. #IDENTITY-INTEGRITY contains properties like wholeness and uniqueness, while #PERCEPTUAL deals with any properties that can be detected by the senses. Note that this property classification is not exhaustive, but it does give an idea of the organization of the tree. A larger section of the tree used in the system appears in Appendix C.

3.4 Structuring the World for the Deductive Component

Since the selection restrictions can provide only negative information, we would also like a finer-grained check to reject the rest of the possible interpretations. There should be some positive criteria, so that we can begin to know that our choice makes sense. For this we need the full power of a deductive system. The question we are working on is how the data base and theorems of the deductive system should be organized. For a start, the

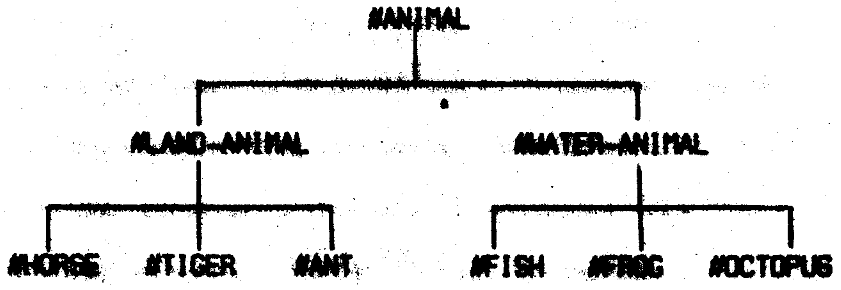


Figure 3.3

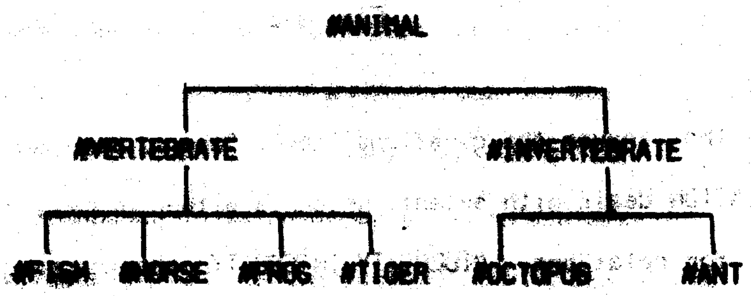


Figure 3.4

classifications used by the selection restrictions look useful. Note, however, that in one context Figure 3.3 might be a useful classifications scheme for objects, while another context might favor Figure 3.4. Even given the same general context, i.e. biology, psychology, chemistry, or the supermarket, many different classificatory schemes are possible. We want more than just a single classification scheme for the general data base, so the structure used will be a group of trees. Actually, since the trees are not completely disjoint, we can merge them into a single lattice structure. Doing this to the two classifications above, we would get the classification shown in Figure 3.5.

What does the hierarchical structure of the deductive data base buy us? First, the motivation for hierarchy is economy of definition. An #OCTOPUS is an #ANIMAL which is a #LIVING-THING which is #CONCRETE which makes it an #OBJECT. Without a hierarchical structure, we would be specifying, for example, that octopuses have some mechanism of locomotion (property of being an #ANIMAL), that they have some sort of reproductive system (property of being a #LIVING-THING), that they are perceivable, although possibly aided by instruments (#CONCRETE), and that they are entities rather than processes or

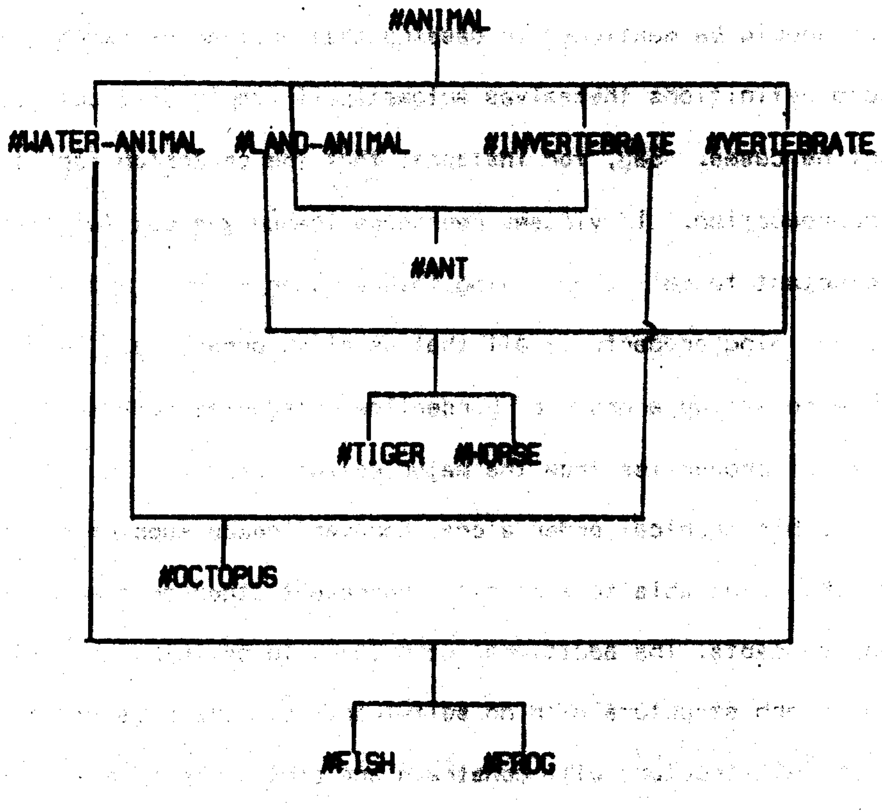


Figure 3.5

attributes (property of being an #OBJECT). All this would be associated with #OCTOPUS, and much of the same would have to be duplicated for #HORSE, #SEAGULL, and the rest. So from hierarchy we get economy of storage.

In thinking about the system, I have limited myself to a single context and a single static hierarchy to avoid the difficulties inherent in adding to, deleting from, and reorganizing conceptual structures. These are interesting but major problems, related as they are to processes like learning, hypothesizing, and shifting from one context to another. A working translation system would require a knowledge structure of considerably greater flexibility than the one outlined here.

It should be mentioned in passing that neither hierarchical organization nor word definitions themselves automatically imply difficulties with borderline cases. Say, for instance, that the criterion for "living thing" is self-reproduction. If viruses reproduce themselves but for other reasons we are reluctant to call them living things, then this is a dilemma only if a single defining property is all that we allow ourselves. In fact, it is possible to set up a group of borderline categories defined by appropriate mixtures of properties from the major categories.

The hierarchical order alone, however, needs supplementing, since we would like to be able to explicitly represent other ordered relationships between concepts. The additional structures to be introduced will lead us to a general graph structure with no self-loops, but the presence of our underlying hierarchical structure will constrain and control the finished result. Since the lexicon is going to be interacting rather actively with the conceptual structure, it might help to look at the situation for words before any additions to the structure are made.

3.5 The Relation of Words to Concepts

There are several ways of approaching the problem of word definitions. A word might be defined in terms of a class and some number of attributes. This sort of formulation is an old one - "genus" and "differentia" are the classical terms - and the standard form of a dictionary definition shows that it is a familiar one: an octopus is "any of a group of mollusks having a soft, saclike body, a large head with a mouth on the undersurface, and eight arms covered with suckers." Here, the phrase up to "mollusks" represents the genus and then the differentiae follow. As their name implies, the differentiae offer criteria for distinguishing between different words defined with respect to the same conceptual class. A definition like this could be used in a learning (i.e. non-static) structure, and it presupposes a top-down learning process. The genus is assumed known and the differentiae give distinguishing characteristics for the new node that is added. (In actual practice, the differentiae may be used to pick out a concept which is already known. In this case both genus and differentiae are known, and the problem is merely learning a new word for an already familiar concept.)

This genus-differentia type of definition might have its place in a system with learning capabilities, although it would not be a fundamental place, since a simple top-down process can not adequately model a good part of the learning that we see people doing. Since, however, I am not considering issues of learning at all, definitions will take a different form. The knowledge data base is assumed to be static in the sense that for the words defined one or more nodes (i.e. concept markers) are always present in the conceptual structure. Furthermore, the differentiae - those bundles of information that distinguish a concept from other members of its class - are also presumed to be in the data base already. Therefore, a word can be most easily defined by reference to the concept node or nodes it represents, and so words can, in a sense, be "plugged in" to the conceptual structure. Since the system already knows about the concepts represented by the restricted number of words it encounters, no conceptual restructuring is going to be necessary.

The fact that a word can be "plugged in" in this manner means that words are ordered with respect to the conceptual structure. Note, however, that not all conceptual distinctions need be reflected lexically; that is, some concept markers might have no words defined for them. This makes sense if you think of sensory impressions: natural language vocabulary does not begin to approximate the number of different shades of color that can be distinguished and remembered, or the number of sounds or tastes.

3.6 Fields

With words in place, it is time to go back to an unsolved problem. The conceptual structure displayed similarities and differences between members of a class, but we have no way of ordering either the members of a single class or members of different classes among themselves. A phenomenon like word contrasts suggests that something like this is needed. One alternative is to order the descendants of a concept node, but if more than one criterion is relevant, bookkeeping could become annoying. And, of course, this does not even address the similar problem of ordering nodes from different classes. The solution proposed is the introduction of a linguistic field. The term "linguistic field" is not new; for a discussion of its history, see Robins (31, p.81). On the basis of what we have developed so far, the field should be an ordering of object, relation, or property concepts using a property as a criterion. We do not want to think of a field as ordering words, since the ordering here is semantic and thus strictly speaking should not deal with the words themselves.

To represent word contrasts, we could use + and - at the extreme ends of a field. One special sort of two member field represents the case where the elements exhaust the domain of the field's property, for example the case of negatives. (There need not be any middle ground between "big" and "not big", where the criterion of size applies.) Antonyms present an interesting situation. Take, for example, "good" and "bad". (Here, and in the next few sections, I will use English examples, since the issues discussed seem to be independent of the language involved.) Most people would call "good" and "bad" opposites, and so their associated concept markers are alloted a two-member field along a property like &WORTH. But what happens when we expand the context to include "great", "excellent", and "lousy"? With respect to these, "good" and "bad" undergo a subtle shift in meaning and are no longer the absolutes for which we constructed the two-member field. Such word pairs can be called polysystemic. We therefore need one or more additional fields for the different frames of reference, and the process of interpreting a word may entail deciding which field is relevant for the particular usage. The implementation details for fields seem to depend on the way knowledge will be structured in the system, so fields will not be considered further here.

3.7 Synonymity and Connotation

In terms of the conceptual structure we have been describing, synonymity can be defined as the relation that holds between words that share the same associated concept marker. Given a set of synonyms, the present structure can therefore represent their similarity; the next step is to look into ways to represent their differences. The first question to ask is whether there exist any pairs of words that are interchangeable in every case. Consider, for example, the words feline and cat. That these are synonyms is exhibited by the fact that a person would be willing to use either to denote the standard furry animal with four legs and whiskers. Feline, however, is a more formal word and one whose usage would tend to be restricted to scientific or poetic contexts; cat is a general purpose word. The distinction here is the familiar one between connotation and denotation. In general, one would want to call denotation the conceptual meaning of a word - what we have been talking about so far - and connotation the phenomenon associated with the use of the word itself. Connotation tells us about the frame of reference in which a word is being used and gives information about the speaker. The properties formal or informal, archaic or modern, educated or uneducated, objective or biased, are some aspects that might enter into the connotation of a word. Language exhibits such economy in other areas (e.g. German nominal declension, see Frey (7)) that it would be surprising to find extravagance at the lexical level. Thus, it seems unlikely that there are two words within a language that are completely equivalent. We have a pair of words - cat and feline - that seem virtually interchangeable with respect to denotation. The difference comes, of course, at the level of connotation, and it seems that this will be the case for all synonyms.

To get this new connotation information into the model, we introduce another sort of field. This field takes synonymous words and orders them along properties like formality, technicality, etc. The definitions in the system use only binary properties, i.e., +- formal, +-technical, etc. In another system that deals with a wider context and has a richer vocabulary, I suspect that one would want greater expressive freedom than is given by binary categories.

3.8 Choosing Concept Markers

In the chapter so far, we have discussed the way concept markers should be ordered, but there has been no indication of how they should be chosen. In fact, it is not at all clear what a set of semantic markers should look like, because they are used in the system by more than one component. Structures built from concept markers are of central importance in the system, so although the following discussion gets ahead of the exposition, I think it is important to stop and consider some of the issues involved in choosing concept markers.

For each sentence, the semantic component will construct a semantic representation. This will be built from concept markers supplied by word definitions and the semantic specialist routines themselves. After a semantic representation has been constructed for a sentence, its component concept markers will be used to call deductive routines associated with them. It is the job of the deductive component to pick the most likely representation and ship it over to the generator. The generator, in turn will use the semantic representation to produce English, and, for this, the English dictionary is ordered so that words are associated with semantic markers or groups of semantic markers. The concept markers, then, have a fourfold role: in source language definitions, in target language definitions, in the semantic representation, and in the deductive data base. (A closer look at a fifth role, that of selection restrictions, is deferred until the next section.)

Given the various roles played by the semantic markers, let us consider the choice of a dictionary definition for the German verb brechen ("to break"), as used in the example:

Fritz brach das Fenster. / Fritz broke the window.

First, we could use a special semantic marker #BREAK with a standard definition procedure for relations (see section 4.2.1). This definition will be compact, easy to write, and easy for a person to read. On the other hand, the marker #BREAK is a rather high level one, and we might choose instead to write the definition in a special form, writing a routine to build a chunk of semantic representation from the three markers #CAUSE, #BECOME and #BROKEN.

Note that the issue is more than just a question of size of primitive set versus ease of expression. A representation using #BREAK is extremely language dependent; this is of dubious merit in general, and particularly so when we are dealing with two separate languages in a single system. The generator takes the semantic representation to be fully unambiguous, and expects to be able to generate from it without further calls to the deductive component (except in certain special cases which will be discussed). The segments of the semantic representation must therefore be simple (low-level) enough so that no information is missing when the semantic representation reaches the generator.

One possible solution might be to use the semantic representation only for deduction. It might therefore contain very high level components and be unabashedly source-language dependent. Deduction, then, would not be choosing between representations, but rather constructing its own, much lower-level representation to be passed on to the generator. This low-level representation would try to approximate language-independence, in the sense that it would aim to be input for a generator of any language.

The approach sketched seems like a good solution, in that source language dictionary definitions could be compact and uniform, while at the same time no information would be lost on the way to the generator. I did not choose it for two reasons. First, a low level representation causes problems with target language definitions. With a lower level semantic representation, a large number of target language definitions would involve associating pieces of the semantic representation, instead of just a single marker, with words. This is a pattern matching problem that, given a large number of words, could slow down the generation process intolerably (at least given current memory architecture). In addition, note that the (#CAUSE X (#BECOME Y #BROKEN)) representation offers no clear advantages for organizing the deductive data base. The kernel of meaning associated with #CAUSE may not vary from one situation to another, but, as the participants change, the nature of the causal relation may, as well. For example, compare the #CAUSE in "Jay broke the mirror" with the causality involved in "Erwin popped the popcorn." The actions associated with causing breakage are hitting, dropping, etc. To cause corn to pop, on the other hand, involves putting a pan of it over some heat source. What I am trying to get at here is that with a low level representation, information used in deduction still would have to be associated with combinations of concepts, i.e. higher levels.

Because of the considerations mentioned, in the system semantic markers are chosen as the union of different word boundaries. If something can have two different lexical representations in German or in English (exclusive of connotation differences, that is), the lower level representation is chosen. For example, kennen and wissen ("to know") wind up with the two semantic markers #KNOW-A-FACT and #KNOW-A-PERSON, while gale and wind (in German, Wind) get the two markers #GALE and #WIND. With this sort of organization, the deductive component is given basically a selective role, and only adds to the semantic representation it is given by filling in certain slots left open by the semantic specialist routines. The semantic representations generated from these semantic markers have no claim at all to language independence, but the loss in generality is compensated for by faster generation.

3.9 When to Use a Restriction

When using concept markers as selection restrictions, it is important to remember that they give only partial information. If used in a system where multi-category definitions and restrictions also exist, the semantic marker restrictions are actually redundant. One justification for this redundancy is efficiency, as discussed earlier in the chapter. Another attraction of the concept marker restrictions is that they are an easy way to get a handle on semantics -- in fact one of the few ways that we know -- without getting entangled in a large and complex web of real-world knowledge. Selection restrictions, then, look like a promising way to use semantics to guide the parse.

The first version of my system did use selection restriction information to guide the parse, although only to a very limited extent. This was in the routine that parses verb objects: if the verb had been found, its restrictions were checked to see if they included a location, a manner, or some other property. If this was the case, prepositional phrases and adverbs were checked to make sure that at least one of these fit the semantic requirement. If this were not the case, then special action could be taken, like checking other syntactic components to see if they had absorbed a preposition by mistake. This was, of course, only limited use of semantics to guide the parse, but it was in an area where a lot of help is obviously needed, namely binding of adverbial modifiers.

In the most recent version of the system, the parser does not use selection restrictions in this way. This decision was primarily motivated by German word order. As was mentioned in chapter 2, for English clause structures, the main verb precedes its objects. We can therefore access the verb's restrictions and use them to look for or evaluate objects. In German, however, only some main clauses have a subject-verb-objects word order, while secondary clauses are usually ordered subject-objects-verb. In addition, any main clause with an auxiliary, modal, or passive verb structure will also have its main verb at the end. Since using selection restrictions to guide the parse requires a certain amount of structure (to handle multiple definitions, optional objects, and variations in word order), the investment in programming effort seemed to promise less return for German than it does for English. Selection restrictions are, of course, still part of the system, but they are used exclusively by the semantic component to eliminate impossible semantic representations.

In the current implementation, semantic restrictions are hung on the concept markers as LISP properties. Since the markers have an explicit tree ordering, the restrictions need not be associated with each semantic marker, but may instead be tacked onto the highest node for which the restrictions hold. This saves space, although of course at the expense of the small amount of time it may take to trace up the tree to fetch restriction lists.

***

In this chapter we have made some decisions about the ordering of concept markers. First, concept markers were divided into objects, relations, and properties. Two main orderings were presented: a tree to implement selectional restrictions and a lattice as a primary ordering for deduction. A secondary ordering was provided by fields, to relate concept markers to each other along a dimension. Words were ordered by virtue of their association with concept markers and according to their connotations. The static, strictly hierarchic concept marker ordering proposed here would not be adequate for a working translation system; however, the conceptual structure is now well enough specified that we can go on to describe the semantic component.

Chapter 4 -- Semantic Processing

4.1 An Overview

4.1.1 The Semantic Component

In the last chapter we considered a static semantic structure, but we have not yet discussed how a particular sentence relates to this general framework. In this chapter and the next, I will try to remedy that situation.

As soon as the German grammar has successfully parsed some section of the sentence, the semantic routines are called in. Their job is to construct a semantic representation for each possible interpretation of the sentence. In many places, the shape of a semantic representation might parallel the parse tree, but at other points, the divergence will be obvious. Where the parse tree is a record of syntactic relations, the semantic representation is an independent structure to record semantic relations that are both implicit and explicit in a sentence. The highly structured semantic representation reflects systematic linguistic phenomena and it is a step on the way from the syntactic representation to the body of information that would be invoked by the deductive component.

The general organization of the semantic component follows Winograd's system, although there have been some fairly high level changes. In the translation system, the semantic representation itself plays a prominent role. Whereas in SHRDLU the representation is essentially an intermediate step in the process of building theorems for deduction, here it is also important as the input to the generator. To use the semantic representation in this way, I have made a number of additions in terms of the information it contains, especially in the direction of a more systematic treatment of thematic features (section 4.9).

Where possible, word definition types have been standardized, so that the individual word procedures can be used descriptively as well as imperatively. That is, the definitions can be examined by the semantic component, which can then take any appropriate actions before executing the procedure. A mechanism has also been added to handle partial information in an orderly fashion (the #UNBOUND marker; see section 4.7.1). The semantic component can add a relation to the representation before all its arguments are bound, and then return to add the arguments of the relation as they are encountered.

The chapter that follows contains a great deal of detail, but it actually does not begin to exhaust the issues discussed. Section 4.2 deals with the way word definitions are used to build the semantic representation, and section 4.4 discusses other contributions to the representation. The actual information in the representation is summarized in section 4.3. Not all words contribute to the semantic representation directly, and this is considered in section 4.5. Section 4.7 discusses markers that cause special actions in the system, while sections 4.6 and 4.8 handle the representation of idioms and coreference, respectively. Finally, section 4.9 discusses the representation of thematic information, and section 4.10 treats semantic case.

4.1.2 The Representation

The semantic representation is constructed from three sorts of components: object, relation, and property semantic structures (OSS, RSS, and PSS, respectively). These components are bundles of information, and their linkage reflects interrelationships within a sentence. In general, a semantic representation is a network rather than a tree. By differentiating between two sorts of linkage, however, we can always find an underlying tree structure in a well-formed sentence. A sample representation is shown in Appendix D, and I will spend most of this chapter discussing why it looks the way it does. The diagrams that will appear below are actually representations of the semantic representation, since the output of the semantic component contains more information than is shown. In the machine version, information is hung on the LISP property list of special atoms produced for the occasion.

A question should be raised here as to exactly what a semantic representation should represent. While I can not really answer this until the semantic component has been discussed in more detail, let me just distinguish two levels of semantic information here: the propositional and thematic levels. Basically the propositional level is related to what is said, the thematic level to the way it is said. The thematic level deals with questions like what information is important in a sentence, what the speaker wishes to convey, and the assumptions he has about what his listener knows. The semantic representation used here is a mixture of propositional and thematic information, and I will come back to the question of what a semantic representation should look like below.

4.1.3 Building the Semantic Representation

The semantic component, like Gaul, has three parts: the semantic specialist routines (SMSPEC), the semantic utility routines (SMUTIL), and the dictionary definitions. The actual building of the representation is done by the SMUTIL routines, and for much of the semantic representation it is the dictionary definitions which make the calls to SMUTIL. The dictionary definitions, in turn, are unleashed by SMSPEC after it has set all the necessary calling parameters. It is the SMSPEC routines that are actually called by the grammar, and many of these routines correspond to grammatical constituents: there are SMCLAUSE, SMPREPG, and SMADJG. The noun group, on the other hand, has a series of semantic specialists, SMNG1 through SMNG3. For a simple noun group, SMNG1 is called as soon as the main noun has been found, and it evaluates any prenominal modifiers starting with those closest to the noun. Control then returns to the syntactic component to parse any qualifiers, and then SMNG2 links these qualifiers to the noun. SMNG2 is also responsible for evaluating relational nouns, which are not touched until the entire noun group has been parsed. SMNG3 checks for reference to other parts of the text if the noun group is definite. Also part of the noun group package is SMCOMPOUND, which handles compound nouns that are not in the dictionary but whose component elements are. Finally, SMPRON and SMPRON2 handle noun groups that are pronouns.

Note that there is no separate semantic specialist for verbs. All the actions necessary for verbs are done by the single SMCLAUSE routine. By the time SMCLAUSE is called, the subject, verb objects, and modifiers have all been parsed. (This is true enough as far as it goes, but not completely true - see section 4.7.1.) SMCLAUSE binds the relation specified by the verb to its participants (subject + verb objects), then binds the modifiers to this relation. This whole process is rather elaborate in practice, and the discussion later on will shed some light on the kinds of information SMCLAUSE has to process.

Wherever the semantic specialists are called in the parse, their general role is to be yea- or nay- sayers. Whenever a semantic relation, say the definition of a verb, is bound, checks are made using the selection restrictions described in the previous chapter. If at any time no representation can be built for the section parsed, then semantics returns failure to syntax. In the system as it stands now, semantics never touches the parse tree except to get information from it, although the unimplemented ideas in section 2.7.1 would involve a more active semantics.

4.2 Lexical Semantic Structures

4.2.1 The Standard Definitions

To get a better idea of the way a semantic representation is built up, let us take a bottom-up approach and start with examples of the three standard definition types.

```
(DEFD  ELEDONE
   SEMANTICS (
       (NOUN (OBJECT  CONCEPT: #ELEDONE
                CONNOTATIONS: +SCIENTIFIC
                LABEL: D1))))
(DEFD  FOLG
   SEMANTICS (
       (VERB #RELATION
                CONCEPT: #GO-BEFORE-IN-TIME
                TYPE: NONE    ORDER:  LEXPASS
                ARGS: 2   LABEL: D2 ))))
(DEFD BLAU
   SEMANTICS (
       (ADJ  (PROPERTY  CONCEPT:  #BLUE
                CONNOTATIONS:  +COMMON
                LABEL: D3))))
```

The semantic definitions here have the following parts:

The Selector:

The first entry in any semantic definition is a syntactic feature, which need not necessarily be the part of speech, any distinguishing feature will do. The syntactic features that were chosen in the course of the parse can thus be used to eliminate semantic possibilities by matching against the first entry in each semantic definition. For example, if schwimmen has been parsed as a noun, there is no need to consider its meanings as a verb. If more than a single feature is needed to discriminate between definitions, then a list of features may be used. In addition, if more than one syntactic feature list takes a particular semantic definition, then a list of distinguishing features prefixed by EITHER may be used.

The Routine Name:

OBJECT, RELATION , and PROPERTY routines are part of SMUTIL, and build OSS, RSS and PSS components, respectively. The rest of the information in the definition supplies parameters for these routines, indexed by the following keywords:

CONCEPT:

This is the semantic marker used in building up the semantic representation. It is a part of the concept structure discussed in the previous chapter.

TYPE: (relations only)

This specifies the relation of the surface arguments to the semantic ones. Types are ONE, TWO, THREE, NONE, TWO&THREE, etc., and they tell which of the semantic arguments may be left understood in the surface representation. For example, we may expect certain relations to have an instrument specified at the semantic level. The relation #CUT, then, would have three arguments: actor, patient and instrument. In the sentence "Karl schnitt das Wurst mit einem Messer" ("Karl cut the sausage with a knife"), the definition used for the verb schneiden would have type NONE, since no arguments are left understood. In "Karl schnitt das Wurst" ("Karl cut the sausage"), a definition of type THREE would be used, since the third argument - the instrument - is left understood. Similarly, "Ein Messer schnitt das Wurst" ("A knife cut the sausage") needs a definition of type ONE, since the actor is left understood.

ORDER: (relations only)

This is another way to supply information for matching surface arguments to semantic ones. Order is used for word pairs like precede and follow, or in German vorgehen and folgen. We want these pairs to map into a single semantic marker, and to do this, one of the pairs is labelled as lexically active (LEXACT) and the other as lexically passive (LEXPASS). When order is LEXPASS, as in the example relation above, then what the syntax has labeled as subject (in an active sentence) becomes the second argument of the semantic relation. The decision about which word is the lexically active one and which the passive one is arbitrary.

ARGS: (relations only)

This is redundant information, since we can always recover the number of arguments a relation takes given its name (by checking the length of its associated restrictions list). ARGS is specified in the definition anyway, however, partly for efficiency reasons, and partly to help me keep track of things when writing the dictionary definitions.

CONNOTATIONS:

This holds the connotation information mentioned in section 3.7, which is expected to be in binary form (+slang, +scientific, -technical, etc.). Right now, this information is optional, and it is not used by the system until generation.

LABEL:

If a word has more than one semantic definition, each is given a label. These are used for error messages and cross-referencing with other information.

The system has a group of functions that pick information out of semantic definitions. These are used in SMSPEC to make sure that the right global variables are set in various cases. What this set of definition searchers amounts to is the ability to use definitions descriptively as well as imperatively. This statement actually needs some qualification, since some information in the definition, like Type and Order, is really meant for use by the SMSPEC routines, rather than as actual semantic information. The definition searchers are not used solely on this Type and Order information, however, so that it is fair to say that the system treats definitions both descriptively and procedurally.

Another sort of information that should be in the definitions, but is not now, is a measure of plausibility. This could take the form of a number that specifies the probability that a semantic sense is used, given that the word occurs. Such probability measures appear in Winograd's system, but were not used here because of time limitations.

4.2.2 Other Definition Types

In addition to the three standard definition types, there are several other dictionary functions: SPECIAL, SMNGMINALIZE, and SMREL-PART. A SPECIAL definition type is used for relations that do not fit existing definition types. A SPECIAL definition just has a small routine associated with it. This sort of freedom is really valuable for exceptional cases like sein ("to be"), but it has been used sparingly elsewhere in the system. Often, a definition will start out as SPECIAL, but then another word like it will come along, and eventually we have a class, warranting its own function type.

SMNGMINALIZE is a definition type designed to handle rankshift. Just as we have clauses serving as rankshifted noun groups ("Ich weiss nicht, warum er hier ist." / "I don't know why he is here."), there are also relations that may be represented by nouns ("Das Stehen fällt ihm schwer." / "He has trouble standing.") In both German and English, any verb infinitive may be a rankshifted noun, and there are also words like "der Aufstand" ("the uprising") whose semantic definitions are basically relations rather than objects. In the semantic representation we want these to appear as relations; but, since the syntax is a noun group, special actions have to be taken (for example, to handle time). SMNOMINALIZE, then, is responsible for some of this action directly and also acts as a signal to other noun-handling routines that special treatment is in order.

SMREL-PART handles nouns that name a participant in a relation. For example, the word Zeichen ("indication") can be defined as "something that is acting as the first argument for the relation #INDICATE." The SMREL-PART for Zeichen, then, builds up a piece of representation that may itself be represented by Fig. 4.1.

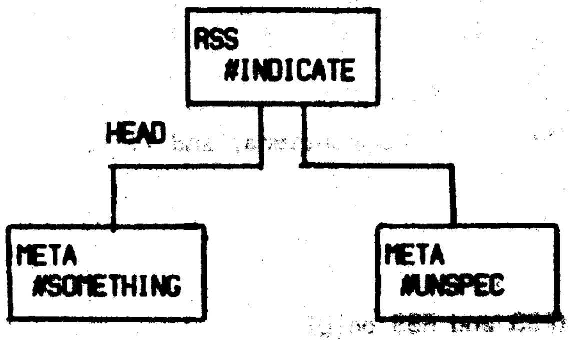


Figure 4.1

The #SOMETHING and #UNSPEC here are marked META because they are meta-concepts, or concept variables. The #UNSPEC marker (for "unspecified" -- see section 4.7.2 below) would probably not be used here, since the information would be given somewhere in the noun group, as it is in our sample text ("Ein deutlich sichtbares Zeichen für die ... Erregungen" / "A clearly visible indication of the annotations") Both the SMREL-PART and the SMNOMINALIZE routines are attractive because they offer economy; we do not need a separate concept marker for a relation or relation participant treated as an object.

4.2 Information in the Semantic Structures

When the definition types above are evaluated, we get semantic components that contain the following information:

OSSNODE=, PSSNODE=, or RSSNODE=

This is the name generated for the semantic node; no two node names are alike.

VARIABLE=

A variable is assigned to an individual instantiation of an object, relation, or property. The node names above may change in the course of building a representation for a particular syntactic structure, but the variable remains the same.

CONCEPT=

This comes from the calling parameters, and it is our old friend the semantic marker.

RESTRICTIONS= (PSS and RSS only)

Selection restrictions are hung from the conceptual structure, being associated with a particular relation, property or subtree thereof. Once the semantic utility programs have retrieved a set of restrictions, they are kept on the semantic node for later reference.

LINKAGE= (RSS only)

This is a list of the participants in a relation, and it represents half of the explicit linkage that holds the semantic representation together.

MODIFIERS=

Another mechanism for linking, this thematic feature binds DSS, RSS and PSS components to their modifying relations. While LINKAGE values produce a tree structure, when values from MODIFIERS are added in we can get a general network structure. A semantic node with modifying relations is called the "head" of these relations.

CASE=

CASE is set to a concept marker that is found above the CONCEPT on the selection restriction tree and distinguished by a special tag. This is redundant information, since the case is always derivable from the concept marker. It is useful information to have around, however, especially for the deductive component. CASE is set only for PSS's and RSS's that act as modifiers, that is, those which are connected to the semantic representation by MODIFIERS linkage. Sample cases are SPATIAL, TEMPORAL, and MANNER. The use of the term "case" for this semantic feature may be misleading, and it is discussed in more detail in section 4.10.

TYPE= (RSS only)

This is the TYPE information supplied by the dictionary routines. It is not really semantic level information, but it is put into the representation anyway for efficiency reasons.

ORDER- (RSS only)

See TYPE.

REFERENCE-SCOPE-

This is set to either GENERIC or PARTICULAR, depending on whether the information given is about a particular object, relation or property, or about the class thereof.

GIVEN-NEW-

This is set to GIVEN or NEW. See section 4.9.2 for an explanation.

COREF-

This gives a list of semantic structures that are coreferent with this one. It is discussed further in section 4.8.

INFO-ORDER-

This is set to either UNMARKED or to a list of the modifying relations in a clause in the order that they appeared in the surface structure. It is discussed further in section 4.9.3.

CONNOTATIONS-

This is the connotation information from the dictionary definition.

THEME-

This is set to the semantic node that corresponds to the theme of the clause. See section 4.9.3 for an explanation.

RESTRICT-DESCRIBE=

This is set to either RESTRICT or DESCRIBE. See section 4.9.3 for further discussion.

CLAUSE-TYPE=

This is set to COMMAND, QUESTION, STATEMENT, or SUBORDINATE. See section 4.9.3 for an explanation.

PARSENODE=

This is set to the parse-node that supplied the concept, if there is one.

PARALLELS=

This is used in representing a variety of coreference. See section 4.8 for details.

4.4 Non-Lexical Entries in the Semantic Representation

The word definitions discussed above form an important part of the semantic representation. Not all entries in the representation, however, are formed by words - some entries represent relations implicit in the syntax of the sentence. Some examples of this will be discussed here - e.g. the postnominal genitive, adjectives that modify nouns, and compound nouns. These are closely-related cases and by no means exhaustive, but they give a representative indication of the issues involved. Wherever there is an implicit relation, it is the SMSPEC routines that supply it and make the call to SMUTIL.

4.4.1 The Genitive

Starting off with an example, the semantic representation for the genitive construction "der Regenschirm der Dame" ("the lady's umbrella," literally, "the umbrella of the lady") might look like Figure 4.2.

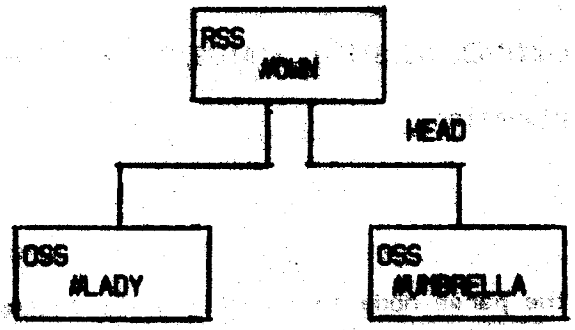


Figure 4.2

For "das Auge des Hummers" ("the eye of the lobster") we might have Figure 4.3.

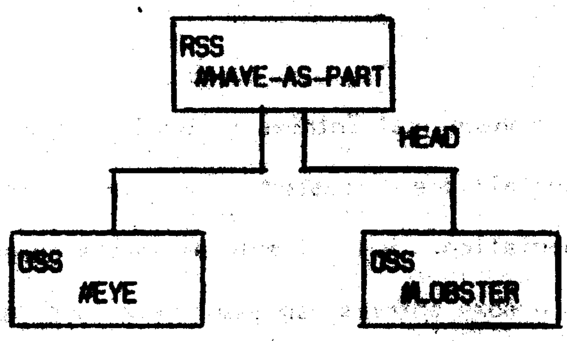


Figure 4.3

In the diagrams, "head" is used to indicate that the linking of the two OSS's is done using the register MODIFIERS, rather than the LINKAGE register. In the two examples, the OSS's are formed in response to the nouns in the phrases, but the RSS reflects an implicit relation. (Actions taken for the determiners have been left out of this initial pass for the sake of simplicity.) The two phrases give no explicit clues to guide the choice between #OWN and #HAVE-AS-PART as interpretations. What's more, there are a number of other possible relations that may be implied by the genitive, e.g.:

der Geruch des Käses = the smell of the cheese
aspect of a thing + thing

das Buch des berühmten Poeten = the famous poet's book
creation + creator

der Stadt meiner Geburt = the city of my birth
aspect of a relation + relational noun

das Geschichte meines Lebens = the story of my life
account + subject matter

There are many such relations that can be expressed using the genitive, but the possibilities here are not completely open. For example,

das Pflanze meines Schreibtischs / the plant of my desk

cannot be construed to mean the plant that is on my desk. To say this, both German and English use a preposition (auf / on) to explicitly express the spatial relationship. Thus, if the number of relations that are implicit in the genitive is bounded, as I believe to be the case, it makes sense to talk about producing semantic representations for the different possibilities. Some of the possibilities are in fact constructed by the system, using selection restrictions as a filter to block the blatantly impossible combinations. The next step is to take a closer look at these semantic representations using the deductive component; but let us first finish up the discussion of how to build semantic representations before going into the question of how to choose between them.

4.4.2 Noun Modification

Another example of a semantic relation that is implied by a grammatical structure is #HAVE-PROPERTY. Any property used to modify an object (or a relation or another property, for that matter) is given a #HAVE-PROPERTY relation. Thus, "der kurzsichtige Wissenschaftler" ("the nearsighted scientist") would be represented by Figure 4.4. The #HAVE-PROPERTY relation is special, in that the selection restriction for its first argument is found associated with its second. In the example, the restriction #LIVING-THING is associated with #NEARSIGHTED, so our scientist would pass the test. Instead of conventional selectional restrictions, therefore, #HAVE-PROPERTY has a procedure, which is automatically executed by the restriction checking procedure. The restriction code for #HAVE-PROPERTY retrieves the selection restriction from its second argument and uses it to perform the check on its first.

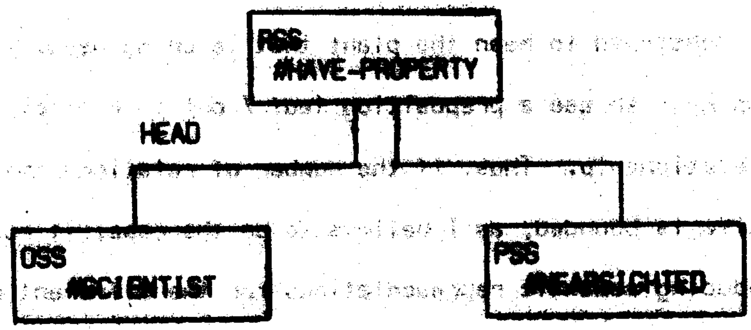


Figure 4.4

One issue in noun modifier representation should be mentioned here. In German, as in English, adjectives may be stacked up in front of nouns in two ways: serially or in parallel. The parallel version is the one that often has a comma "die kleine, lebhafte Eledone" ("the small, active eledone"). The

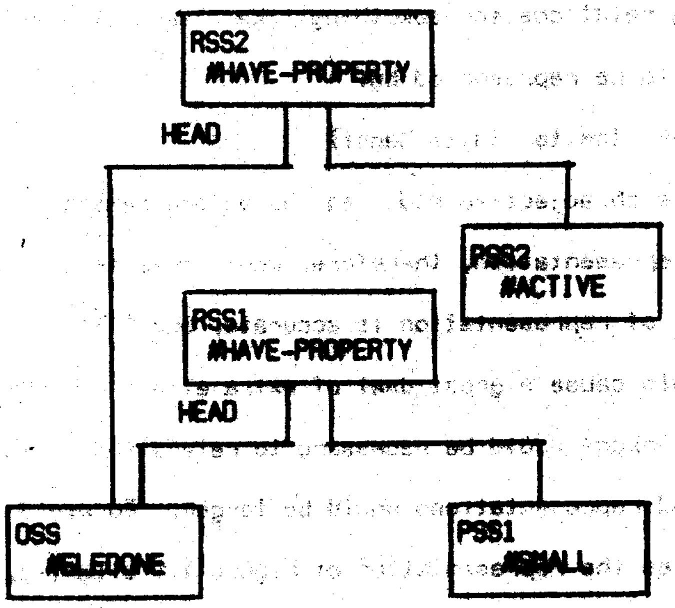


Figure 4.5

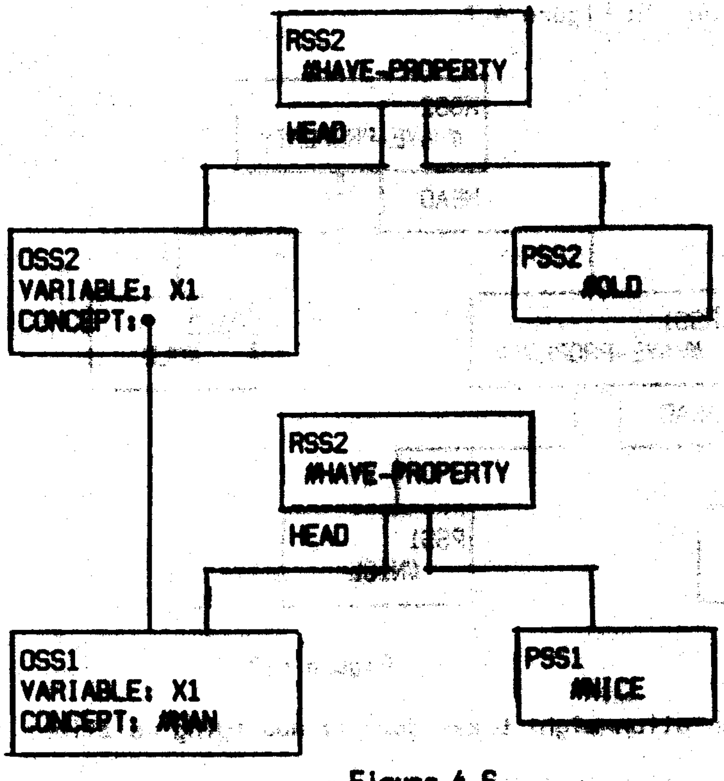


Figure 4.6

representation for this is straightforward, as shown in Figure 4.5. Serial adjective lists, on the other hand, pose problems of representation. The modifying relations for something like "der nette alte Mann" ("the nice old man") could be represented by:

der (nette (alte Mann))

That is, each adjective modifies the entire remainder of the phrase. The correct representation, therefore, would seem to be that shown in Figure 4.6. This sort of representation is accurate, but I think the proliferation of OSS's would cause a great deal of extra effort for the generator. Another sort of linkage would be necessary to relate the different OSS's of a noun group, and representations would be larger. To avoid this, the system abbreviates the representation of Figure 4.6 slightly, using the modifier relation established between adjective and noun instead of a new OSS. The result is shown in Figure 4.7.

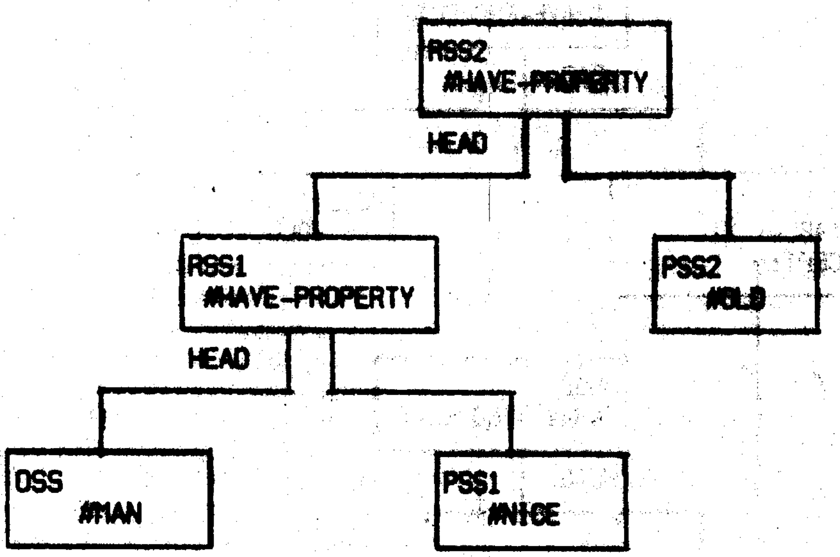


Figure 4.7

This representation might break down if modifying relations themselves have a lot of modification, but this would only be happening for prenominal clauses, not #HAVE-PROPERTY relations, and clauses will probably not be stacked up serially more than two-deep.

I should note that only the simplest sort of adjectives are currently handled by the semantic component. For an analysis of some of the complexities involved in English adjectives, see Vendler (35).

4.4.3 Compound Nouns

The German compound noun is often translated by a classifier plus noun in English (for example, "die Feuerwehr" / "the fire department"). Among the relations that occur between the parts of German compounds are relations discussed for the genitive and simple noun modifiers, so the representation described in the last two sections is also applicable here. With compounds, we also get relations that could be expressed using prepositions, such as:

die Gummistiefel = Stiefel aus Gummi
rubber boots (material of)

die Trinkgläser = Gläser zum Trinken
drinking glasses (use or function of)

die Todesanzeige = eine Anzeige wegen des Todes
death notice, obituary (occasion of)

die Seereise = eine Reise auf der See
sea voyage (place of)

Especially with the more common implicit relations, we would like to be able to handle compounds that are not in the dictionary but whose parts are. The system does this in the routine SMCOMPOUND. In a procedure analogous to that for genitives, the semantics of the component words are bound to a group of possible relations, selection restrictions permitting, of course. The representation produced looks either like the output of the genitive routine or like representations of other noun modifiers. This approach is desirable because the representation resembles those built for similar structures, the dictionary is not loaded with redundant definitions, and the system is able to cope with new compounds it has not seen before.

The drawback of using the semantic component to supply implicit relations from a built-in set is that the range of relations possible between the compound's components is much wider than the range of relations for either genitives or simple noun modifiers. I suspect that it is an open set, and this would mean that certain word pairs could have implicit relations that are completely idiosyncratic. If a relation appears only in one compound, it obviously does not belong in SHCOMPOUND. This seems to be the place for a dictionary definition, and in fact the SPECIAL definition facility in the system could handle the situation with no trouble. A RSS for the implicit relation could be built in the definition, binding the compound's components as participants. This is clearly an efficient approach, as long as the system has a definition for each idiosyncratic compound it encounters.

Between the compounds formed from a predictable set and the completely idiosyncratic compounds are a group that show some regularity, although the particular relations involved are unpredictable. These are given representations by the semantic component, but an #UNSPEC (for "unspecified") marker is used. The marker is discussed below in section 4.7.2, and the compound class is investigated in more detail in section 5.5.

Finally, there are compounds for which the meaning of the whole is different from the sum of the meanings of its parts. Consider "der Tintenfisch" ("the cuttlefish" or "squid;" literally, "ink fish"). Leaving aside the point that the squid is not a fish to a biologist, we note that "der Tintenfisch" refers not just to any water animal that spews an inky cloud, but to the cuttlefish. If there is some other fish-like creature that also spews ink, it would not be designated by <u>Tintenfisch</u>. Such a situation seems to warrant a separate concept like #SQUID, and so the representation for this type of compound looks no different from that of a regular noun.

For representing compounds, then, the system offers four alternatives: pre-packaged implicit relations, special dictionary definitions, representations that leave the relation unspecified for the time being, and standard object definitions for compounds whose meanings are more than the sum of the meanings of their parts.

This last group of compounds mentioned raises a question. Do not all compounds, in fact, tend to be more than the sum of their parts? When faced with "Gummistiefel" ("rubber boots"), we know something about this special type of footwear, just as we have specific information about "Schnürstiefel" (literally "lacing boots" - any boot that has a shoelace) and "Holzschuhe" ("clogs"). We would want to associate this information with a specific concept marker, rather than the more general concept #FOOTWEAR. Given the system we have now, if the representation for "Gummistiefel" was supplied by SMCOMPOUND, there is nowhere to put the information. This is a problem, but it is one that the system does not have to face, since it is not intended to do any learning (i.e. it is assumed that the information in a sentence would never be used to permanently change the deductive data base).

If we wanted to allow learning in the system, we might try the following approach. When a new compound is encountered, the implicit relation could be selected by SMCOMPOUND, but then instead of adding this to the semantic representation, we could create a new concept marker. For example, a new "Gummistiefel" concept would have #FOOTWEAR as genus and something like (#MATERIAL-OF X #RUBBER) as differentia. Then any new information learned about rubber boots, e.g. that they are worn in the rain, could be associated with the new concept. In addition, we could add a dictionary definition for

Gummistiefel, so that the next use of the word would invoke the new concept. It would only be economical to build new concepts, of course, if they are really useful for organizing information in the deductive data base. There would therefore have to be some criterion for how much special information is needed to justify the creation of a separate concept marker. The creation of new concept markers would also affect generation, so we would need a facility to update dictionary definitions for the generator accordingly.

Since, however, we are not trying to describe a framework for learning, there is no need to generate new concept markers. The compromise used in the system was to give concept markers to compounds that had a good deal of information associated with them per se - like "das Nervensystem" / "the nervous system". Words for which the bulk of special information would probably be encountered in the input text for the first time - like "das Chromatophorenspiel" / "play of the chromatophores" - are represented by chunks of semantic representation (relation + participants). These are constructed by SMCOMPOUND, as described above, unless an idiosyncratic relation is involved.

***

It is thus necessary to supply implicit relations for genitives, adjectives, and compounds in the noun group in order to develop representations for the different possibilities. In general, only the deductive component can decide between the set of different representations produced.

4.5 Words Without Semantic Representations

In the last section we looked at relations that were not tied to specific words, but now the question is whether an entry in the semantic representation must be formed for all words. In fact, not all words add to the representation; generally, words that do not are low in semantic content and very high in syntactic function. The most common case of this is prepositions that mark participants in relations. For example,

Die Eledone reagierte *auf* den Reiz.

The eledone reacted *to* the stimulus.

Such prepositions occur both with verbs and adjectives, but only the former will be discussed here. Obviously not all prepositions are of this type; those that mark location, time, causality, etc. are high in individual semantic content and not dependent, except in the most general way, on the particular verb used. The prepositions I am considering are those closely tied to individual verbs and whose functions are performed by case in other situations: "mir gefällt *es*" versus "ich freue mich *darüber*" in German (roughly, "I like *it*" and "I am happy *about it*"), *look at* versus *observe* in English. Where such prepositions occur, their semantics is essentially a no-op, i.e. the semantics of the prepositional object is stuck into a register marked by the preposition name. When the relation associated with the verb is evaluated in SMCLAUSE, a list of its required prepositions is retrieved from the collocations list. These prepositions are indexed by semantic definition labels, since a difference in preposition can indicate a difference in semantics ("dient als" / "serves as" versus "dient zu" / "serves to"). For each definition, then, we know exactly where to look for the semantics of its participants. In the case of a required preposition, we just pick up the object's semantics from the register that was set. Note that essentially the same procedure is followed for separable prefixes, since a separable prefix verb is considered to be one word. This uniformity is desirable in light of the close relation between prepositions and separable prefixes.

It is not, perhaps, entirely fair to claim that prepositions that mark objects of the verb are devoid of semantic content. In fact, I was surprised at the regularities that I encountered in the course of organizing the selection restriction tree. Fillmore's case theory [5], of course, is partly based on this sort of regularity. As an example, the object of a mental process (including in this perception) is often marked by <u>an</u>

(a) Ich denke oft dar<u>an</u>.

I think <u>of</u> it often.

(b) Ich errinere mich dar<u>an</u>.

I remember it.

(c) Das ist <u>an</u> Sepia zu beobachten.

That can be observed <u>in</u> Sepia.

In a sense, these regularities are not surprising, since people have to remember which prepositions go with which verbs, and the more regularities, the better. On the other hand, the situation is complicated by the fact that the same preposition may be used with a rather wide variety of verbs in a number of different ways. Thus we have for <u>auf</u>:

(d) Er wirkt <u>auf</u> das Publikum.

He had an effect <u>on</u> the public.

(e) Er reagiert nicht dar<u>auf</u>.

He did not react <u>to</u> it.

(f) Wir haben stundenlang <u>auf</u> ihn gewartet.

We waited <u>for</u> him for hours.

The semantics of these required prepositions, then, can give a clue to the relation between subject and verb object, but the knowledge is never definitive without the evidence provided by the verb.

4.6 Idiomatic and Special Usages

Another situation in which the words in a clause might not map one-to-one onto the semantic representation is when idioms are present. I am using "idiom" here in the semantic sense, to mean any phrase whose meaning is different from the sum of the meanings of its component words. A semantic representation for an idiom, then, is properly associated with the phrase itself. In the system, the most general way to handle an idiom is with a SPECIAL definition. For something like:

Das Eisen schmieden solange es heiss ist.

Strike while the iron is hot.

we could write a SPECIAL definition for <u>schmieden</u> ("strike") that would check for the presence of the rest of the phrase, then provide a new semantic representation to embody a meaning like "Act while there is an opportunity." Note that the system does not deal with such extensive idioms right now, because none are present in the sample paragraph.

The system does handle more restricted idiomatic usages in two ways: through the collocations list and through the selector mechanism. The selector was mentioned above in section 4.2.1. Its purpose is to cross-reference semantic definitions with syntactic features. A facility not mentioned above is the appearance of a word, as well as a feature, as selector. In such a case, the semantic definition is applicable only if the next word in the sentence matches this word.

The word selector facility is obviously limited, and it would not be included in the system if it did not come essentially "for free." The collocations list is potentially of greater generality, although right now it is used only for associating prepositions with verbs, as described in the last section. In the collocations list, we can index the meaning of a verb by a preposition or separable prefix elsewhere in the sentence. It would not be difficult to extend this mechanism to handle idioms like our "Strike while the iron is hot" example, and I suspect that this would be a good way to proceed. Writing a SPECIAL definition for every idiom that we want to add to the system would be time consuming, and the code would be repetitious. The best policy seems to be to use SPECIAL definitions sparingly, and to use the collocation mechanism to reflect the regularities that can be found.

4.7 Special Entries in the Semantic Representation

Several special concept markers are used in the semantic representation as it is built up. These are the concept variables #SOMETHING, #UNSPEC, #UNBOUND, and #REFERENT. The first, #SOMETHING, helps represent nouns that name a participant in a relation, and its use was illustrated in section 4.2.2. The three other metaconcepts, which will all have been replaced by the time that the generator gets the semantic representation, are the subject of this section.

4.7.1 The #UNBOUND Flag

The #UNBOUND marker is a temporary placeholder which disappears by the time the semantic component finishes its work. The purpose of this marker is to allow evaluation of relations before all their participants have been bound. This is not in any way a theoretical necessity, since we can always wait until all the participants are in before evaluating a relation. I find it a satisfactory solution, however, because it keeps the semantic component fairly modular; that is, the #UNBOUND mechanism allows as much processing as possible to happen as soon as a phrase has been parsed.

As an example, consider the phrase "die im Nervensystem verlaufenden

Erregungen" ("the excitations that run through the nervous system"; literally, "the through the nervous system running excitations"). Since we want to handle the prenominal clause "im Nervensystem verlaufenden" as soon as it is parsed, SMCLAUSE produces the representation in Figure 4.8 (and maybe others, of course, for the alternative interpretations). Note that the prepositional phrase "im Nervensystem" was also handled using #UNBOUND, but the processing for the prenominal clause has already done the binding there. The representation shown is left at the clause node, where it sits until the main noun Erregungen is parsed and SMNG1 is activated. SMNG1 calls the

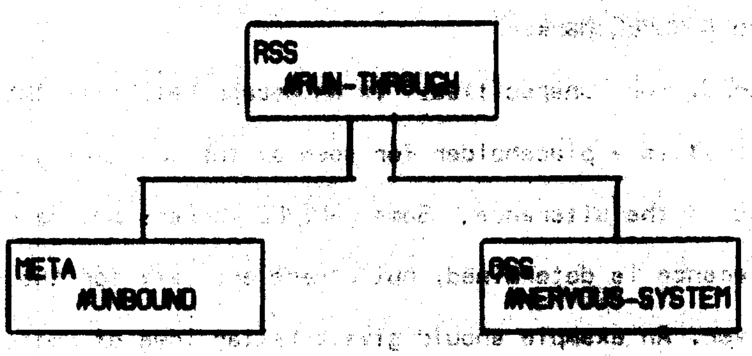


Figure 4.8

SMMODIFIERS routine, which is, as its name implies, a general clearing house for modifiers of nouns, verbs, adjectives, and the rest. SMMODIFIERS takes note of the fact that the modifier is a prenominal clause, so it knows that an #UNBOUND needs to be replaced by the OSS for the noun. Seeing this, it calls REBIND to make a selection restrictions check, do the binding, and supervise any renaming that is necessary to keep the semantic representation consistent.

Besides prenominal clauses, the #UNBOUND mechanism is used for preposition groups (since prepositions are generally represented by two-place relations), for subordinate clauses of various sorts, and for those adjective groups that are represented as relations. Although in some cases all constituents for participants of the relation will have been parsed, usually not all will have been evaluated semantically (for example, the main verb). Therefore, the use of the #UNBOUND mechanism is justified in most cases, and to keep things uniform, I have used it throughout. This sort of feature is probably also useful in interpreting English (assuming that several other high level decisions are also kept). The special nature of German syntax (prenominal clauses, end-order verbs), however, makes some sort of partial binding mechanism essential.

4.7.2 The #UNSPEC Marker

#UNSPEC, for "unspecified," is an escape hatch for the semantic component; it is a placeholder for some of the information that is left understood in the utterance. Some #UNSPEC markers may be replaced when noun group reference is determined, but others are left for the deductive component to mull over. An example should give a better idea of what #UNSPEC is used for. One place for this sort of marker is in relational nouns. Often time is left for the reader to fill in, as in

(a) Karl errinerte sich an das Rennen.
    Karl remembered the race.

(b) Karl freute sich auf das Rennen.
    Karl looked forward to the race.

In (a), the race predates Karl's mental action, in (b) Karl's mental process comes first. Further, for many uses of relational nouns, some participant is left out, frequently the agent. In a lot of cases, the agent is understood to be the universal anyone, for example, "Schilaufen kann gefährlich sein" ("Skiing can be dangerous", i.e., anyone who skis can find it dangerous). Not all such constructs imply "anyone," however. Some agents can be unique, as in "der Erfinder der Buchdruckerkunst" ("the inventor of printing"); while most understood agents can only be determined with respect to the context: "das Abschalten des Stroms" ("the cutting off of the power") could be done by a homeowner, a company, or a repairman. The #UNSPEC marker can be used, then, to defer the decision until more information is available. Another use for #UNSPEC is when the grammatical passive has no agent ("Der Strom wird abgeschaltet" / "The power is being cut off"). We get the same range of possibilities here as in the relational noun with understood agent.

4.7.3 #UNSPEC for Ellipsis

The #UNSPEC uses above are basically determined by syntax. In some other situations where information is left out, there seems to be a lexical basis for the deletion. In our paragraph, for example, there is a discussion of whether octopuses can perceive color. After giving evidence that supports the existence of color perception, the author says, "...so scheint das zum mindesten für diese Formen für einen Farbensinn zu sprechen" ("...this, then, at least for these species, seems to support a color sense"). Here the author has substituted a noun group for a relation such as "the existence of" plus the noun group. It seems to me that abbreviations like this depend very much on the special sense of particular words (with possibly some grouping into classes of words that allow similar types of ellipsis). In this example, the ellipsis might be triggered by "sprechen für" ("support"). For these cases, then, it will be up to a SPECIAL definition routine to introduce the #UNSPEC marker and do the necessary binding. Deduction can then decide what relation is understood. This definition approach guarantees that the system can handle special cases and know what it needs to bind for each particular case.

4.7.4 The #REFERENT Marker for Pronouns

The #REFERENT marker is used for third person pronouns, since these have no concept markers to call their own. This reflects the fact that for the semantics of a pronoun, we are totally dependent on information from the coreferent noun group (or on our knowledge of the actual referent, as for the first and second person). The #REFERENT marker is supplied by pronoun definitions, and it is similar to #UNBOUND in that it has been replaced by the time the semantic representations reach the deductive component. The mechanism for handling #REFERENT is also functionally similar to that for #UNBOUND, in that the same sort of rebinding is done.

Let us look at the use of the #REFERENT marker in more detail. First, for things like personal and relative pronouns, #REFERENT might be replaced almost immediately. SMNG1 causes evaluation of the pronoun definition, setting up the #REFERENT marker. SMNG2 probably will not be called, as we generally will not have qualifiers following these pronouns. SMPRON is then called to handle reference. Its job is to construct a list of possible referents (using heuristics taken with little change from Winograd's system) and to eliminate those that do not agree with the pronoun syntactically (on the basis of gender and number). The #REFERENT OSS is then rebound to each of these possible referents. As semantic processing continues, some of these will probably be eliminated by selection restriction checks, and the final coreference decisions will be made by the deductive component.

From what has been said so far, the reader might conclude that the #REFERENT marker is not necessary in every case. It is true that if the pronoun OSS is rebound immediately, then the marker is an extra step. Even personal pronouns, however, can make forward references, which are enough to justify the marker. Consider the example:

Ehe _er_ immatrikulieren kann, muss _Johann_ die Aufnahmeprüfung machen.

Before _he_ can enroll, _Johann_ has to take the entrance exam.

Here, the #REFERENT marker is inserted for _er_, and it is not rebound to the OSS for _Johann_ until SMCLAUSE is called for the major clause. Another situation where the #REFERENT marker is justified is the da-compound, where forward reference is frequent. In the sample paragraph we have:

...was ebenfalls sehr _dafür_ spricht, dass diese Tiere Farben zu unterscheiden vermögen

...which again very strongly suggests that these animals are able to distinguish colors

Here, _dafür_ refers forward to the "dass" clause. For these sorts of pronouns, the #REFERENT marker frequently remains as one possible interpretation until the end of the sentence, waiting for the referent to be found.

When the #REFERENT marker is rebound, we will want some way to represent coreference. Since coreference in full noun groups will be represented like pronoun coreference, both are discussed together in the next section.

4.8 Representing Coreference

It seems desirable for pronouns and definite noun groups (more properly, for all noun groups that are coreferent with other noun groups in the text) to have similar semantic representations. By the time the generator sees the representation, there will be no explicit indication of whether the surface structure contained a pronoun or a full noun group. This makes sense, since the target language has its own rules for coreferent noun groups and pronoun insertion, and these may or may not coincide with the rules in the source language. In this section, I will discuss finding the coreferent noun group, the representation built, and issues of pseudo-coreference. Although I will speak of a noun group "referring" to another noun group, I really mean that the two are coreferent. Only extra-linguistic things (or quoted words or phrases in a linguistic discussion) are actually referred to by noun groups. Note that references can be made to other statements by using pronouns like *das* and *was*. In the section below I will concentrate on noun group coreference, but the situation is much the same when a relation is involved.

For a pronoun, we said that the semantic component accumulates a list of possible coreferent structures. For full noun groups the situation is slightly different, and the approach outlined here follows Winograd. Pronouns are so weakly specified semantically that they cannot be separated from their referents by a great distance. Full noun groups, however, are much better specified, and a referent could potentially be found anywhere in the text. While I suspect that references outside a paragraph are limited to certain key noun groups, I also think that determining these noun groups is non-trivial, i.e. not obvious from surface structure in every case. So to proceed for full noun groups as we did for pronouns - constructing a possibilities list and narrowing it - will not be feasible. Even if we were to limit our search for referents to the scope of a paragraph, our possibilities list would not be very interesting, since we have no good way to narrow down the possibilities. In some situations we might be able to use selection restrictions to narrow possibilities, but in general they will not be adequate. Furthermore, coreferent full noun groups do not agree with their referents in gender and number. We would have a potentially bulky possibilities list and nothing to do with it. For this reason, it is left to the deductive component to determine which full noun groups are coreferent. Although the deductive, rather than the semantic component, will be adding coreference information to the semantic representation for full noun groups, I would like to finish up the discussion of coreference at this time.

The main feature of the representation is that coreferent items are given the same variable (register setting VARIABLE=), although their OSS or RSS names differ. Pronoun semantic nodes will always take their CONCEPT setting from the coreferent noun group when #REFERENT is rebound. A full noun group, on the other hand, always supplies its own concept marker. Given a variable name alone, it is not always easy to find other semantic nodes sharing the same variable, so the register COREF is set to a list of all the coreferents of this node. That is, when we find a back reference, we set the COREF register both on the referring semantic node and on the node refered to. This will be useful for generation, since a back reference in one language might be better translated as a forward reference in another. Figure 4.9 shows some of the information that will be present in the OSS's of the two coreferent noun groups: "the cephalopod" and "this animal."

```
OSSNAME= OSS1
CONCEPT= #CEPHALOPOD
VARIABLE= X1
COREF= (OSS2)
```

```
OSSNAME= OSS2
CONCEPT= #ANIMAL
VARIABLE= X1
COREF= (OSS1)
```

Figure 4.9

Note that other registers are set separately for coreferent semantic nodes, so that they may, and often will, differ in MODIFIERS, GIVEN-NEW, etc.

There is another phenomenon that behaves much like coreference, but which we might call pseudo-coreference. Consider the example:

Anna benützte das grosse Wörterbuch und ich benützte <u>das kleine</u>.

Anna used the big dictionary and I used <u>the small one</u>.

The noun group underlined in the German is elliptic for "das kleine Wörterbuch" ("the small dictionary"), and we can supply the main noun by looking at a noun group earlier in the sentence, "das grosse Wörterbuch" ("the big dictionary"). We might think of the second noun group as making a back reference to "das grosse Wörterbuch," except that it refers to the general class "Wörterbuch" ("dictionary") rather than to the more restricted class "large dictionary." In this case, we are not really dealing with coreference, but with an abbreviated way of distinguishing two separate but related objects. We represent the two with different variables, but also note that the abbreviation has been used by filling in the slot PARALLELS for both noun groups. This register is also used to represent parallelism exhibited by conjunctions. (In our example sentence above, the RSS's for the two conjoined clauses would also be marked parallel.) PARALLELS is also used for parallel main clauses, either connected by a semi-colon or in separate sentences. The basic idea is that this parallelism information is not really language independent, since many languages might have different rules for conjunctions or noun group pseudo-coreference. However, the use of the PARALLELS register saves time spent in making comparisons in the generator.

4.9 Semantic Representation for Thematic Features

Thematic systems represent the organization of an utterance as a message. As such, they are not restricted to the discourse level (i.e. linguistic organization above the sentence). In fact, thematic systems can be found at the word, group, and clause levels of a systemic grammar. The distinctions in this section are adapted from Halliday's work, hopefully with the original intentions intact. Readers wishing to judge for themselves are referred to Halliday (13,18).

4.9.1 What's in a Semantic Representation - Revisited

Before considering thematic phenomena in detail, let us return to the question "What should a semantic representation represent?" We can divide the information in a semantic node (see section 4.3) into three classes, one of which will be thematic.

First, the semantic representation contains information from the propositional level - more or less. Strictly speaking, I would consider the propositional level to be raw knowledge - semantics minus the thematic systems. But, as we will see below, the object-relation-property distinction can be called thematic. To call the propositional level semantics minus thematic information, then, means that it is an extremely low level of organization. Let me therefore qualify the original statement and say that the semantic representation contains information from the propositional level, augmented by the object-relation-property distinction. In this category, I would place the registers CONCEPT, CASE, LINKAGE, VARIABLE, and REFERENCE-SCOPE.

A second kind of information in the semantic representation relates the surface structure of an utterance to this semi-propositional level. For this, we have the registers TYPE, ORDER, RESTRICTIONS, OBSNODE, RBSNODE, PSSNODE, and PARSENODE. This is information that is useful in deciding whether a representation is appropriate, deciding between different semantic representations, and keeping track of the way the semantic representation corresponds to the syntactic structure that is being built by the parser.

The rest of the information in the semantic representation is thematic. This includes CONNOTATIONS, GIVEN-NEW, COREF, PARALLELS, INFO-ORDER, CLAUSE-TYPE, MODIFIERS, THEME, and RESTRICT-DESCRIBE. The remainder of this section will be devoted to these thematic categories.

4.9.2 Thematic Organization Below the Clause Level

Most thematic information seems to be related to the clause; the group level thematic features that will be discussed here are also found at the clause level. One system discussed in this section - information focus - is a discourse system, and should properly be represented above the sentence level. Since, however, its manifestations are seen at the group and clause levels, it will be represented at the relevant group and clause semantic nodes.

A thematic phenomenon at the word level is connotation, related as it is to the speaker's choice between different ways of expressing the same concept.

Looking next at the group level, the head / modifiers linkage seems to be thematic. Consider, for example, the difference between the two noun groups:

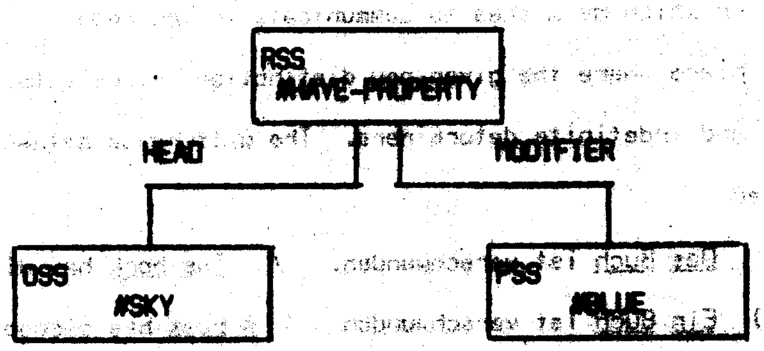
(i) der blaue Himmel / the blue sky

(ii) die Bläue des Himmels / the blue of the sky

In the semantic representation, the only difference between them is the MODIFIERS register.

Another group level thematic feature is the distinction between objects, relations and properties. Objects and relations can both be seen as bundles of properties; in defining them, we are making a commitment to a coherent world view, i.e. to some sort of "identity" in the case of objects and to the assumption of "relatedness" instead of randomness in the case of relations. This conceptual leap of faith does not seem to be a conscious choice on the part of an individual speaker, but rather a choice that is built into the language. The reason I say language here instead of conceptual structure is that it is possible, for example, that the concept of "objecthood" differs from language to language, as Whorf contends in his analysis of Hopi (36). Nevertheless, for the translating system, the object, relation and property distinction is expected to be maintained in the deductive data base.

(i)



(ii)

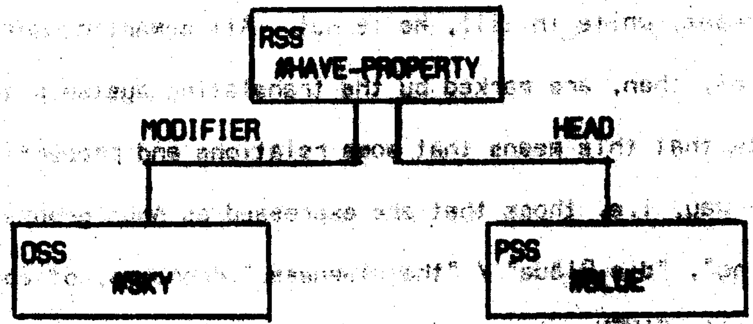


Figure 4.18

This is because the distinction seems to be integral to both English and German. If, in fact, languages do distinguish objects, relations, and properties in different ways, and if this reflects deep conceptual differences as well, then the conceptual representation chosen here is heavily language dependent in this respect.

Finally, the discourse categories "given" and "new," which belong to what Halliday calls the information system, also appear as a group level thematic feature. Since this system is realized primarily phonologically, only certain aspects of information organization are going to be relevant. GIVEN information is that which the speaker (writer) thinks the listener (reader) can deduce, either because it has been stated explicitly or because it is common knowledge or because it is in some sense unique, etc. NEW information, on the other hand, is the reason for the writer's sentence, i.e. the information which he wishes to communicate to the reader.

One place where the given-new distinction is reflected in text is in definite and indefinite determiners. The difference between the noun groups underlined

(i) Das Buch ist verschwunden. / The book has disappeared.

(ii) Ein Buch ist verschwunden. / A book has disappeared.

is that in (i), the reader is expected to know, or very soon find out, which book is meant, while in (ii), he is not. All semantic nodes corresponding to noun groups, then, are marked by the translating system with either GIVEN or NEW. Note that this means that some relations and properties are also marked along the way, i.e. those that are expressed as noun groups: "das Schwimmen" / "swimming", "die Bläue" / "the blueness." Pronouns, of course, are automatically GIVEN, since their referent is always expected to be derivable, either when they appear (for back reference) or as soon as more of the sentence has been processed (for forward reference).

Another place that the information focus system is reflected in text (although it is not now handled), is shown in the following example:

(i) der blaue Klotz / the blue block

(ii) der Klotz, der blau ist / the block that is blue

One important difference between these two is that the entire noun group in the first example must be GIVEN; while in the second, the subordinate clause allows "blue" to be NEW, even though the rest of the noun group is GIVEN.

At the word level, then, there is connotation, and at the group level there are the distinctions between head and modifiers and between objects, relations, and properties. These two group level distinctions also appear at the clause level. Finally, while the information focus system is properly discourse level, the given-new distinction is represented both at the group and at the clause level. Note that two other registers in the semantic representation related to information focus are COREF and PARALLELS, which were discussed in section 4.8 above.

4.9.3 Thematic Organization at the Clause Level

An important clause level thematic system is the theme-rheme distinction. In terms of Halliday's definition, the theme is the first constituent of the clause. In the translating system, therefore, the theme is marked by the syntactic component when the clause is parsed, but it is also given a semantic representation. In semantics, the theme register on the clause RSS is set to the semantic node associated with the theme. Halliday has characterized the theme as "the peg on which the message is hung" (16,p.161). The rheme is the rest of the clause. In terms of the information structure, the theme is often GIVEN, as in:

<u>Die Chromatophoren</u> spielen. / <u>The chromatophores</u> play.

This is not always the case, however, as shown by this sentence from the sample paragraph:

Nach von Hess, sollen sie sich wie der farbenblinde Mensch verhalten.

According to von Hess, their (the cephalopods) behavior is like that of a color-blind man.

From the reader's viewpoint, theme acts as a set of directions for interpreting the information in the sentence. When the theme is GIVEN, the writer is saying, "Here is a concept with which you are familiar, on which you can hang the information I am going to give you." On the other hand, when theme is NEW, the writer is setting the scene, giving information he considers helpful or essential to interpreting what will be said in the rest of the sentence. In the example given, it is important for the author to qualify his statement by attributing it to von Hess. In essence, he is saying, "Don't assume I believe what I am going to tell you." In the example sentence, this qualification is also expressed lexically by the use of the verb <u>sollen</u> ("supposed to be"). In fact, the author goes on to give evidence that some squids do perceive color.

Another thematic mechanism that relates to what the reader is expected to do with the information he is given is what I will call the restrict-describe distinction. Basically, a head-modifier type relation is RESTRICT if the modifier is expected to give the reader useful or essential help in identifying the HEAD. DESCRIBE information may also be useful, but it is treated by the writer as supplementary information. A relation has the attribute RESTRICT when modifiers are used to limit the reference of the head. "Der rote Pulli" ("the red sweater"), for example, exhibits this kind of relation, since not all sweaters are red, and the adjective has been used as a distinguisher. "Die rote Feuerspritze" ("the red fire engine"), on the other hand, would probably be a DESCRIBE relation. This is because red is generally a property of fire engines, and so the modifier has been used purely descriptively, rather than as an attempt to single out a particular object. This distinction is reflected syntactically in the English restrictive and non-restrictive clauses:

(a) Cephalopods that live in coastal areas build their houses out of stones.

(b) Cephalopods, which live in coastal areas, build their houses out of stones.

In (a), the subordinate clause is used as a distinguisher, while in (b) it gives supplementary descriptive information. Note that English requires commas for (b) but not for (a), and the distinction is often emphasized by contrasting _that_ and _which_. In German, on the other hand, the translation for both (a) and (b) would be:

Die Kraken, die in der Küstenzone leben, bauen ihre Wohnhöhle aus Steinen.

or better -

Die in der Küstenzone lebenden Kraken bauen ihre Wohnhöhle aus Steinen.

Since both types of clauses have the same surface realization, semantic knowledge must be used to make the distinction. Here we have a situation where semantic interpretation is necessary for German to English translation, since if we are to choose the correct English representation for such a clause, we must interpret the German connective.

Another clause level feature is the register INFO-ORDER. If one had to classify this register, it would be assigned to the information focus system, although the INFO-ORDER register itself is a very ad hoc measure. The semantic component checks the semantic cases of adverbials in the clause and marks INFO-ORDER accordingly. If the adverbial ordering is the default one, the register is set to UNMARKED. If the ordering is not the default one, then the register is set to a list of the RSS's of the adverbials, in the order of their appearance in the sentence. Presumably, those nearest to the end of the clause are considered most important by the speaker (unless there is some other reason for the ordering, like abundant modifiers), and this might be useful information to preserve for the generator.

One last thematic category used at the clause level is CLAUSE-TYPE, which is a register set on the RSS corresponding to the clause. This register may have the values COMMAND, QUESTION, STATEMENT, or SECONDARY. It is assumed that this information will be used by the generator and also by the deductive component. We would expect the deductive component to know something about the implications of the type of clause used, i.e. the presuppositions and expectations associated with it. For connected text, this sort of information would primarily be useful for disambiguation, i.e. to indicate the ways the information in a clause could be used in deduction. For more interactive uses of language, the expectations associated with CLAUSE-TYPE would indicate the type of action that must be performed, e.g. carrying out a task, finding an answer, etc.

4.9.4 Discourse Semantic Structures

Originally, I planned a seperate discourse semantic structure (DSS) which was to be associated with each sentence and carry information about intersentential relationships. Except for the information system treated above, however, very little of this seems to be derivable from the surface structure of a text. I will defer a discussion of discourse level structuring, then, to section 5.7.

4.10 The Place for Case

A case grammar in the style of Fillmore uses semantic case information for several purposes. For some of these functions, I have used other mechanisms in the system. When particular prepositions are required by the verb, for example, the collocations list is used (section 4.5 above). For objects of the verb in general, I have ignored case entirely, assuming that this information would be used at the deductive level. Winograd's system has the case-like global variable SMLOC (location), which is bound to a location required by the verb, as in "Put the book <u>on the table</u>." My system, in contrast, uses only SMONE, SMTWO, and SMTHREE to specify participants in a relation. This was done because selection restrictions filled the role of case here; SMLOC and the other globals that one could add, like time and manner, were redundant.

Semantic case does, however, have a place in the translating system, although in giving it one I have extended the meaning of the term. For the rest of the section, let us consider modifying relations exclusively. In the system, selection restrictions specify the constraints a modifier places on its head. In addition, we also need a way to express the types of modifiers that a head can take. For example, we might want to specify that events can be modified by location, time, and manner, or that physical objects may have shape and color. These constraints are not now implemented, since I expect them to be embedded in the deductive routines. What is implemented is the characterization of the individual modifying relations by high level categories which I will call semantic cases: orientation, location, shape, etc.

While the constraints on modifiers have not been implemented, semantic case information does have other uses in the system; the rules for ordering modifying relations are expressed in terms of case. That is "die graue grosse Tintenfisch" ("the grey big squid") sounds strange in both German and English because the rule "size before color" has been violated. (Note that we may need more generality than the simple case categories to express all the ordering rules, but at least case goes a long way toward expressing the more common regularities that occur.) Case also helps to explain verb modifier ordering. In German we would be more likely to say "Wir traffen uns gestern (time) in London (location)," while in English the more frequent arrangement would be: "We met in London (location) yesterday (time)." I should add that right now the German end of the system does nothing more with case than find it. The semantic component does not care whether it sees "die grosse graue

Tintenfisch" or a "graue grosse" one. These rules, of course, could be added easily enough. Semantic case information does have a place in generation, however, since modifier ordering rules in the target language are expressed in terms of case.

The mechanism for retrieving semantic case information is a simple one. Since I am assuming that different concept markers always imply different cases, the logical place for case information is the selection restriction tree. To build the semantic representation for a relation concept marker that acts as a modifier (including, of course, #HAVE-PROPERTY), we trace up the selection restriction tree until a marker is found with its CASE property set. Semantic cases are associated with subtrees of the selection restriction tree, although there may be several cases along a branch. This allows us to handle exceptions, since the first case found is the one used.

***

In this chapter we have discussed the semantic representation and the way a set of representations are associated with individual sentences. The next chapter discusses some of the issues that must be considered if we are to choose a single interpretation for a given sentence.

Chapter 5 -- The Role of Understanding

5.1 Introduction

The process of understanding a sentence may be thought of as the process of relating it to an internal knowledge structure. As was already mentioned, the translating system implemented makes no gesture toward understanding, but instead by-passes the problem entirely (with the possible exception of selection restrictions). This omission arises not out of the belief that understanding is unimportant to translation, but, rather, out of the conviction that a fragmentary solution is no solution at all. While the understanding component for the system remains a "black box," the mechanism needed to fill this gap is not as ill-defined as it once was. Recent work by Minsky (27), Charniak (2), Goldstein (9), McDermott (29), and Sussman (34) is extremely exciting, and constitutes substantial progress toward a theory of representing and structuring knowledge.

The chapter that follows relies heavily on the ideas in the references cited above, but I will be considering much more restricted questions. First, given the system that is described here, what sorts of interactions would we expect between a knowledge structure and the rest of the system? Second, I will take a short look at the sorts of special problems that come up in text and some of the cues we can take advantage of. In what follows, I will refer to our "black box" as the deductive component, although this term is misleading. Deduction is probably an important part of understanding, but not necessarily the primary mechanism. I use the word "deduction" instead of "understanding," however, since the interactions outlined between the component and the system here might not be identical to the interactions between a more general "understander" and a system's linguistic components.

In what follows, I will assume that the deductive component is written in a language with at least the representational power of Conniver.

5.2 The Basic Functions of the Deductive Component

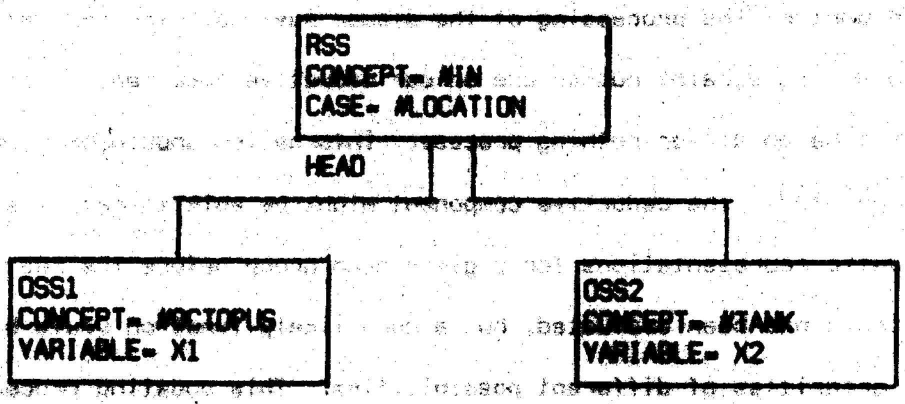
Understanding plays a crucial role at the interpretive end of the translating process: we need to understand in order to decide which sense of a word is intended, to untangle pronoun references, and so on. We would therefore expect strong interaction between the deductive component and the parsing and semantic components. As will be discussed in the next chapter, we might also need to draw on our general knowledge structure for generation - in particular, when paraphrase is necessary. This seems to be a more specialized mechanism, but I am not prepared to discuss it further, so it will not be considered here. This leaves, then, the interpretive role of the deductive component, which can be divided into two functions: disambiguation and supplying information that is implicit, but not explicit, in text. In terms of the system here, these tasks can be reformulated as choosing between possible semantic representations and filling in the slots left open in them (the #UNSPEC marker). These processes are not independent, but rather intimately interrelated. Clearly, a choice between representations is made easier when all the information is in. On the other hand, we can make a decision about implicit information only when we have committed ourselves (at least temporarily) to a particular context. The situation is not as hopelessly circular as my presentation of it; I merely wish to emphasize that implicit information can have its uses in disambiguation, and disambiguation, in turn, will supply implicit information.

We cannot really ask how the deductive component will interact with the rest of the system before we ask when it will do so. Ideally, of course, the understanding section of the system would be active, directing the parse. Since, however, text interpretation is still at the stage where syntactic information is usually the best information available, the question is really when the semantic component should call deduction. A likely place to make the call is at the end of each semantic specialist, so that bad representations can be eliminated right away and do not have to be carried forward. In addition, if all representations are rejected by deduction, we have some good information to send the parser. The sooner deduction realizes the error, the less complicated backup will be, since we are still at or near the scene of the difficulty.

Understanding is not, however, a single monolithic process. Some linguistic structures require a delay in parts of the process. For example, where in English we would say, "Give me the red pencil and the blue one," the German could be, "Gib mir den roten und den blauen Bleistift" (literally, "Give me the red and the blue pencil"). Here, the first noun group in the conjoined structure cannot be fully evaluated until the second has been parsed and evaluated. Similarly, a German end-order clause construct requires that a good part of the processing of the clause must wait for the main verb to be parsed. Constraint number one on our deductive component, then, is that it cannot be an all-or-nothing process. Information should be usable as it is accumulated. The deductive component might be able to reject some possible semantic representations for a given noun group before the rest of the sentence has been evaluated, but a more likely function would be to reshuffle the priorities of different possibilities. This updating process will be discussed further in section 5.4. Related to the use of partial information, we would also want to activate information not just for full grammatical units (e.g. noun group, clause, etc.), but also at certain important intermediate points, for example, as soon as a verb is found. Thus, the implementation of the deductive component outlined here would also require some changes in the semantic specialists currently in the system.

5.3 Relating a Sentence to the Knowledge Structure

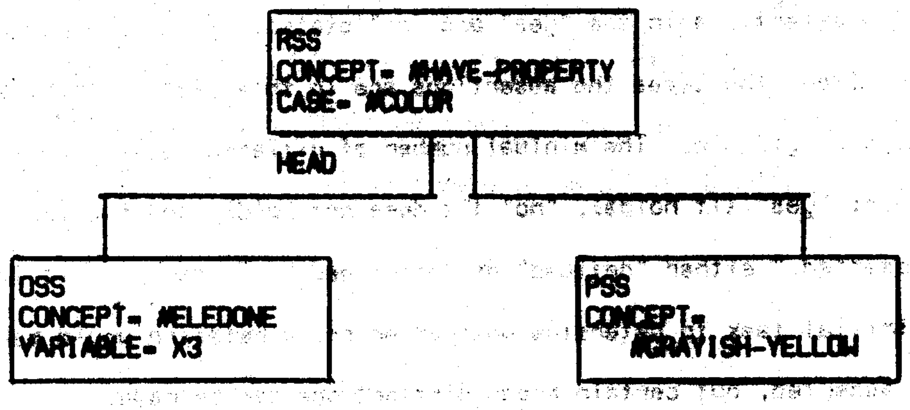
It is assumed that the deductive component will be activated by an approach similar to that used in Winograd's system. Each possible semantic representation of a sentence is converted into a set of assertions for the deductive data base, and procedures are automatically built from these assertions to perform the necessary deductive processing. Conversion of the semantic representation to an assertion set is straightforward. The variable in an OSS is combined with the concept marker and the relation #IS, for example, (#IS X1 #OCTOPUS). Each such tuple is given its own assertion name, as well, and an assertion name can appear as an argument in other tuples. For RSS's, the relation concept markers are combined with the variables of their arguments (if these are OSS's) or the assertion names of the relations formed



A1: (#IS X1 #OCTOPUS)
A2: (#IS X2 #TANK)
A3: (#IN X1 X2)
A4: (#LOCATION X1 A3)

Figure 5.1

from arguments (if these are RSS's). Finally, modifying relations are given assertions based on concept markers and assertions based on case. Figure 5.1 is an example of the assertions that could be produced for a modifying relation. For the special case of the modifying relation #HAVE-PROPERTY, we build only one assertion, namely one made up of the case of the property, the variable or assertion name of the head, and the property's concept marker. Figure 5.2 gives an example of this. These assertions do not represent all the information in the semantic representation, and it is expected that the procedures will use thematic information as they go about turning the assertion set into routines to be used by the deductive component. Just how this will be done, however, seems to depend very much on the way general knowledge is to be structured, so I will not consider the question further here.



A5: (#IS X3 #ELEDONE)
A6: (#COLOR X3 #GRAYISH-YELLOW)

Figure 5.2

When tuples like those given above are asserted in the deductive data base, it is expected that they will trigger all or some part of the related information that is stored as permanent knowledge. In the simple hierarchic model from Chapter 3, this would involve a chain reaction up through the hierarchies, so that (#IS X1 #OCTOPUS) would trigger the assertion (#IS X1 #WATER-ANIMAL), (#IS X1 #INVERTEBRATE), (#IS X1 #ANIMAL), etc., along with the related information in each case. Most assertions will be more complicated than these #IS tuples, and we will want to be able to record information about the status of an assertion, e.g. under what conditions an assertion can be expected to hold. Certain minimal distinctions are essential. First, we would want to distinguish between a relation that we know does not hold and one about which we simply have no information. In addition, we want to distinguish between a relation that is unasserted but could hold for a situation, and one that is not only unasserted, but also irrelevant. For example, we can ask "Where did Harvey put the book?" but not, "Where did Harvey put the rapidity?" Location, of course, is relevant for concrete objects but undefined for abstract ones. Furthermore, we would also want to qualify assertions in the "yes" and "no" states, by the sources of the information, the times the assertions are in this state (sometimes, often, Mondays only), etc. The minimal number of different assertion states, then, is four: "yes" (it holds), "no" (it does not hold), and two varietes of "unasserted," either "defined" or "undefined." In general, it is probably a non-trivial task to determine whether or not a relation would make sense if it were asserted, but certain broad distinctions can be made.

5.4 Positive Selection Restrictions

The use of a verb raises certain expectations about the nature of the participants involved. Similarly, attributive adjectives raise expectations about the noun modified, and prepositions carry constraints on their objects. The selection restrictions introduced in chapter 3 allowed us to express minimal conditions on participants in a relation, but there is more information that is potentially useful. For example, if we know one of the participants in a relation, we often have a much better idea of what the others might be. Let us call the restrictions from chapter 3 negative selection restrictions, and define a positive selection restriction to be a block of information about the expected participants in a relation that can be used by the deductive component to decide between possible interpretations of an utterance. When we try to specify more closely what positive selection restrictions should look like, we run into some issues which we did not have to face when dealing with their comparatively simple negative counterparts.

As mentioned above, if all possible semantic representations for a phrase are rejected, the parse itself will be rejected. Since we might expect positive restrictions to be rather intimately related to the general knowledge structure, we could find ourselves in a situation where an incorrect statement by the author violates the restrictions and is sent back to be reparsed. This is obviously undesirable; the deductive component should distinguish between false statements and nonsensical ones, at least as much as possible. Note that nonsense will be considered a misparsing here, since I am assuming that when a nonsensical statement actually appears in text (as in children's stories or a discussion of "colorless green ideas"), we will have been adequately warned by context. An example of the potential confusion of nonsense with misstatements will probably be helpful here.

> Eingehende Untersuchungen über einen etwaigen Farbensinn der Cephalopoden sind sehr erwünscht.
>
> Thorough investigations on the existence of a sense of color in cephalopods would be very desirable.

Let us assume that <u>erwünscht</u> is defined by the concept marker #WISH-FOR (We might want something more precise, but this will do.), and that the first argument here is the generalized #PERSON modified by #ANY. To specify the second object, we would give it the negative restriction #RELATION and treat the situation where an object appears in this place as the sort of ellipsis discussed in section 4.7.3. What else can be said about relations that are wished for? One important factor is that the relation does not now hold, or at least the speaker believes that it does not now hold. If, in fact, conclusive studies of cephalopod color perception did exist at the time this article was written, the reader must assume that the author did not know about them. Our knowledge of these hypothetical studies would not, however, block the interpretation of the sentence altogether.

It is harder to envision some positive restrictions becoming involved in misstatements than others. Our "wish-for" example is probably a fairly common candidate for misstatement. On the other hand, the exchange in (ii) is quite a bit less likely to occur than the one in (i):

(i) Harry is a bachelor. -- No, he got married Saturday.

(ii) Jane is a bachelor. -- No, she's a female.

A mistake about marital status is presumably easier to make than one about the sex of an individual.

For positive selection restrictions, then, our knowledge structure should provide a mechanism to evaluate how likely a speaker is to make a mistake. This mechanism could take the form of a value, function, or procedure associated with each restriction to calculate the probability that the restriction will be violated in the given context. We might not need completely explicit information here about the likelihood of violation, since some of it might be deducible from the knowledge structure: if knowledge in the system is arranged so that certain facts are much less prone to mistakes than others, then the system could assume that the writer is also much less likely to make misstatements of this variety. Of course, this would only serve as a take-off point, since particular context (e.g. whether a child is writing) or other misstatements from the same source might alter our expectations about the likelihood of particular errors.

Interested readers are referred to McDermott [26] which treats questions of belief and doubt in the assimilation of new information.

5.5 Filling in the Blanks: Dealing with MUNSPEC

We have said that in addition to deciding between possible representations, the deductive component should be able to supply information that has been left implicit in the text. In the implementation, it has been assumed that such implicit information comes in two varieties: information that will be necessary in generation and information that will not. Therefore, a lot of implicit information will not be reflected in the semantic representation at all. Consider the examples:

Merkur flog nach Athen. / Mercury flew to Athens.

Lindbergh flog nach Paris. / Lindbergh flew to Paris.

Every object has an implicit time setting and duration. If the Mercury here is the historical one, we would have good evidence for assuming that he made the trip under his own power, since the airplane had not yet been invented. On the other hand, since Lindbergh's flight postdated the invention of the airplane, it would be highly unlikely that the flying here is done in any other way than in an airplane. (Knowing that Mercury had wings on his feet and that Lindbergh made the first U.S. to Paris nonstop solo flight would be even better information, of course, but that is not the point here.) Implicit noun tense, then, seems to be useful information for disambiguation.

Since implicit noun tense seems useful, should it also have a place in the semantic representation? Chafe's group at the University of California at Berkeley has identified the question of implicit information as an important one for translation. In the German-English language pair, I have not encountered any instances where the "tense" of objects must be explicitly represented in the semantic representation in order for smooth translation. (There may, of course, be "emergency situations" where the generator cannot find a well-formed translation and might need to make additional calls to the deductive component, but right now I am only considering cases in which the generator is successful.) My system is based on the assumption that for a <u>given target language</u> we can predict which implicit information will be required for generation and which will not. It will be interesting to see whether this assumption of the predictability of explicit information is correct. I should note that because of this assumption, the semantic representation in the system is highly language dependent; that is, dependent on the two languages involved. We might expect a system with languages less closely related than German and English to have very different information in a semantic representation and to differ considerably from the present system in its representation-building behavior.

We can say, then, that the deductive component will be supplying implicit information both for its own purposes and to add to the semantic representation for the generator. Let us concentrate on the information that will clearly be necessary for generation and see how it can be supplied. In section 4.7.2 we discussed the use of MUNSPEC for things like times, locations, agents, etc. With a mechanism such as that proposed by Minsky (27), this sort of information would be supplied by a rich default structure. Of course, a default need not be completely specified by the internal structure. For example, we might choose a general default for the location of an object to be, "If there is no reason to think otherwise and the location of the object has been specified previously, assume it is still in the same place." It seems fair to assume that defaults will usually be heavily context dependent.

In chapter 4 we suggested the use of #UNSPEC for elliptic situations where a relation was implied by its arguments. There seem to be several possibilities here. The first variety appears in our sample paragraph:

> Ob die Cephalopoden selbst auf Farben reagieren, ist nich bekannt... so scheint das zum mindesten für diese Formen _für einen Farbensinn_ zu sprechen.
>
> Whether the cephalopods even react to color is not known... at least for these species, this appears to support _a color sense_.

The full relation has been presented directly or can be immediately deduced from the context, and the argument of the relation appears later as an abbreviation. In the example, we would expect the deductive component to use information about reacting to immediately reformulate the first sentence as a question of whether cephalopods have a sense of color. Later, when we see the reference to a color sense, we can check context and supply the implicit relation(s):

```
(#SUPPORT
     A10
     (#HAVE-FACULTY #CEPHALOPOD #COLOR-SENSE))
```

Here, A10 refers to another assertion. Thus, with good understanding of what is happening in the text, this sort of ellipsis can be handled in a straightforward manner.

For implied relations that are not immediately supplied by context, the system will need a more general mechanism for finding a typical relation given one or more of its arguments. We will see this mechanism used again for compound nouns below. I suspect that these non-contextual implied relations will be drawn from a rather restricted group, with relations like #EXISTENCE-

OF heading the list; but this speculation is not based on extensive analysis. Note that some implied relations are derivable unambiguously from surface structure (for example, the English "if possible" is equivalent to "if it is possible"). We would expect to find more variety in semantic representations for these cases, but they are not relevant right now, since they would not be marked with #UNSPEC.

Finally, let us consider a use of the #UNSPEC marker that was mentioned but not discussed in chapter 4. Between the compounds that are linked by completely idiosyncratic relations and those that are linked by predictable relations (#HAVE-PROPERTY, #MATERIAL-OF, #HAVE-AS-PART, etc.) is a group of compounds that can be handled with the #UNSPEC marker. Consider the example:

die Haustürschlüssel = die Schlüssel für die Haustür
door key (instrument + object of action implied
- here, öffnen, to open)

Here, we want to find a relation that is the function of the key, i.e. (#OPEN #KEY #DOOR). There are a whole series of these functional compounds, where the relation itself is unspecified. So once again, the deductive component will have to suppy a typical relation when given its arguments. Note that for the #FUNCTION-OF case we are not asking for a whole new mechanism, since we would expect a knowledge structure to have strong links between an object and its function.

Compounds do exhibit relations besides #FUNCTION-OF that we might want to represent by #UNSPEC. For example,

der Handkoffer = hand luggage
(luggage that can be carried by hand)
der Kabinenkoffer = steamer trunk (literally, cabin trunk)
(luggage that can be used in a ship's cabin)

For these and other more associational compounds, it is not clear that there is only one relation that can describe the connection between the compound's components. It seems desirable, however, for the deductive component to supply a likely relation or relations to the semantic representation. Then, if there is no equivalent compound or classifier plus noun combination in English, the generator can use the semantic representation to produce a clause or parenthesized explanation to describe the object.

5.6 Other Contributions of the Deductive Component

Except in dealing with the #UNSPEC marker, the deductive component is not expected to alter the semantic representation. We can expect it, however, to set registers on semantic nodes with information that will be useful for generation. In particular, this information would include categories like Halliday's known-unknown and variable-value [13]. Final choices between the categories restrict-describe and generic-particular, would also be made by the deductive component, as well as decisions about coreference. As the generator becomes more sophisticated, there will no doubt be other categories that we would want to add to this list.

5.7 Deductive Processing above the Sentence Level

In order to decide between semantic representations, the deductive component will have to construct a model of the text. Little information above the sentence level is currently incorporated into the semantic representation, since it is not yet clear how such information can be used in text generation. Nevertheless, I think it is worthwhile to take a look at some structural aspects of our sample paragraph. As we can see from the English translation on page 15, the example paragraph can be divided into six sections. These divisions can be characterized as follows:

1. An upward pointer: relating what is to come to what has come before in the text
2. What it is: general description of the mechanism
3. What it is used for: function or use of the mechanism
4. What causes it: how to activate the mechanism
5. What goes with it: accompanying actions
6. Issue: can cephalopods perceive color?

In this analysis, sections 1-5 are clearly related, and one would expect them to be handled using special knowledge of the way an author would describe a process or mechanism. The sixth point here is not directly related to the rest, and we would not expect the process-description handler to deal with it directly. The issue of color perception is not, however, unrelated to the other five sections. The basic reasoning here is, "Cephalopods are color producers. Are they also color consumers (i.e. perceivers)?" The sixth section, therefore, investigates the inverse of a key relation in the paragraph. The associational link between section six and the rest of the paragraph is a fairly common phenomenon. Similar associational links might be used to introduce historical information or to make points that are too short to merit a new paragraph of their own. From this analysis, we can conclude that while paragraphs are generally organized around a single topic, we cannot expect a strictly top-down, one-topic-per-paragraph organization. One good heuristic seems to be to look for associational links near the end of a line of description or reasoning (where the six segments above would be considered lines of description).

Any attempt to build a model of the sample paragraph would also have to recreate the reasoning used. One of the patterns we see in the paragraph is:

(i) Generalization: X is a cause of Y.

(ii) Evidence: Example where X is blocked and a change in Y is observed.

This sort of reasoning pattern would be associated with causality, and it could be cued lexically by "veranlassenden" ("causing" line 9 of the German text on page 13), "bei" ("upon," here a causal relation, line 11), "abhängen von" ("depend on" line 13), and "eine Wirkung ausüben" ("exert an effect" line 19). A variant of this reasoning pattern - where evidence precedes generalization - might be slightly more difficult to handle. It is obviously easier to interpret evidence if we know what it is evidence of. It seems clear that at the paragraph level, too, we will want the deductive component to handle partial information as it is accumulated.

5.8 Metaphoric Language

Most types of text that would be considered for mechanical translation probably would not contain phrases like "the babbling brook" or "the raging storm," so one might conclude that the ability to handle metaphoric language is not important in a practical system. This would be, I think, an unfortunate conclusion. Metaphor is relevant, because it is part of the more general problem of the creative use of language. In metaphor, we take the definitions of the individual words involved and suspend some of their rules, while transforming others slightly. Metaphor is one situation in which a deductive component will have to reason by analogy, but it is not the only one. Often, we see a phenomenon similar to metaphor in word use. Consider, for example, the word Erregungen from the first sentence of our paragraph. In a technical sense, as it is used here, the word means an electrical excitation, or impulse. When talking about human emotions, Erregung can mean agitation, while in a social situation it can mean commotion. We might handle this by giving three different definitions to the word Erregung, but then we risk missing the common ground that exists between these three senses of the word. A knowledge structure should be able to represent the fact that the three senses are analogous: there is a disturbance of a state of rest by a phenomenon that is unstable or unpredictable in some sense. Metaphoric language, then, seems to differ from other sorts of language use in degree rather than in quality.

One possible difference between metaphor and normal language use is that metaphor is unpredictable. When a new metaphor is encountered, we presumably have to call in our analogy processor to find the points of similarity and the points that are irrelevant. Once the senses of a word are known, however, it is not clear that the closeness of the similarities is as important anymore. (There could be the usual benefits of a shared model - economy of storage, uniformity of representation, etc. - but it is not clear that these issues are relevant.) This contrast of metaphor and regular language ignores the point that language usage has to be learned. When we encounter the word Erregung in a technical sense for the first time, we are already prepared with a set of possibilities that can be evaluated to determine which are relevant to the new usage. To handle both metaphoric language and other more common sorts of language use, then, a deductive component will need to be able to inspect and alter word definitions (or some model of them), and will need the general ability to reason by analogy.

A related issue here is that of "dead metaphor." Some clichés and idiomatic expressions have lost the freshness of their analogies, and one might conclude that the original senses are no longer relevant. Some English examples might be "That takes the cake!" and "He was bowled over." I have no idea of the original senses of these, although it would probably be interesting to find out. The point here is that we can still use such phrases even when their original comparisons are unknown or long forgotten. For these sorts of dead metaphors, a system should probably hook the phrases directly to their intended meanings, without worrying about analogies. Thus, the first example would have roughly the same meaning as "That's outlandish," and the meaning of the second example would be about the same as "He was astounded."

It is not always easy, however, to tell when a metaphor is dead, and I think we should be careful not to throw away too much. Consider the situation of prepositions. For the purposes of chapter 4, the basis for associating a preposition with a verb or an adjective was treated as arbitrary. If one looks deep enough, however, there is often a compelling reason for the choice. Consider the example "abhängen von." We would translate this as "depend on," but the literal meaning is "hang down from." Here, dependence is formulated in terms of a physical situation. If we hang X from Y and then, say, move Y, this will have an effect on X. Note that the English "depend on" uses a similar physical analogy, but it is a slightly different one, that of support: If X is set on Y, then a motion in Y will effect X as well. (The Latin ancestor of <u>depend</u> - <u>dependere</u> - shares the hang-down-from analogy, but that is irrelevant here.) One could argue that the spatial analogies here represent dead metaphors, and that hanging things down and piling things up have no real connection to our thoughts about dependence. The more closely one studies prepositional use in both German and English, however, the more spatial analogies can be found. This situation seems to be especially interesting in the light of Minsky's suggestion that a single mechanism could account for both visual and conceptual processing (27). Prepositions seem to be one more example of the intimate relationship between visual and conceptual processes.

5.9 By-passing the Deductive Component

The implementation uses an extremely simple mechanism to by-pass understanding. It is described here for completeness only, since it will be obvious that such a scheme would be impractical in a working system. The program requires that the user understand the text and supply the information that would ordinarily be supplied by a deductive component. By the time all this interaction has taken place, the user could have long-since translated the text by hand. Nevertheless, the deductive by-pass routine does allow us to get an idea of how the system would behave if it were more complete.

After the semantic component has produced the possible representations the user is asked to decide which representation he would like to send to the generator (by typing in a number), or whether he would like to see the generator try its hand at all the possible representations in succession (by responding ALL). For each representation that is to be generated, the user is then asked to supply concept markers for the nodes with #UNSPEC. Finally, nodes for noun groups marked GIVEN and PARTICULAR are presented, and the user is asked to specify other nodes that are coreferent. The other information mentioned in section 5.6 is not requested right now, since the generator is not fine-tuned enough to use it.

Chapter 6 -- Generation

6.1 The Process of Surface Generation

6.1.1 Input to the Generator

Having completed syntactic and semantic analysis of an input sentence, we are now at the point where generation of an English sentence can begin. The first question is just what the surface generator should have as its input, and in general the answer to this is not difficult: we want to work with the semantic representation that has been constructed. In designing the semantic representation, every effort was made to include as much information as possible, with the hope that this would be sufficient for the generating process. As will be discussed below, the semantic representation is in fact not adequate for every eventuality, but it still constitutes the major input to the generator.

One could question whether the semantic representation is the proper input for generation. For example, when translating written German into English, I find myself using syntactic guidance. One look at a prenominal clause like "die unter der Haut liegenden Zellen" (literally, "the under the skin lying cells") and "relative clause" or "that" comes to mind. This use of language-dependent translating rules or heuristics may be a personal idiosyncracy, or it may, in fact, be one of the shortcuts that people often use when translating. At any rate, rules depending on the relation of source to target language were not used in the translating system. There seemed to be no case where this was necessary, since it was always possible to formulate semantic rules corresponding to language-dependent syntactic ones.

6.1.2 Comparing Generators and Interpreters

The surface generator was written in an extended version of PROGRAMMAR, and in many ways it is similar to the interpreter. Since the interpreter and generator were built for two different natural languages, we would not expect them to be identical on a line-for-line basis. I would argue, however, that even if only a single language were involved here, we would not expect the generator to be a line-for-line inverse of the interpreter. This is because there are different knowns and unknowns for the two processes. In the end, both the interpreter and the generator will have made analogous choices, since both will be using linguistic information based on a systemic characterization of the languages involved. But different information will be available at different times for the two processes. An interpreter may have to delay several processing steps until a local ambiguity can be resolved (or, alternatively, it may try one of the possibilities and backup). A generator, on the other hand, does not necessarily have to concern itself with the issue of ambiguity at all, since all of the information it needs is available in unambiguous form from the semantic representation. (Here, I am using local ambiguity to mean ambiguity that will be resolved by the time a parser has finished syntactic analysis of a sentence; see Hill [19]) In addition, while the choices made for generating and interpreting are comparable, the relative importance of the choices will differ. The interpreter uses its knowledge of grammatical redundancy and, in its better moments, semantic likelihood to decide which choice was made by the author of an utterance. The generator, on the other hand, has a characterization of the meaning intended. It must make choices to communicate its message effectively and, with luck, gracefully and unambiguously. We can therefore expect the two processes to differ in relative timing and emphasis.

6.1.3 Translation and General-Purpose Generation

Several times already I have referred to the generating component as a "surface" generator, and I should perhaps clarify what I mean by this. In particular, how does the generation process envisioned here compare to the general process of writing text in one's own language? The big difference is that the generator in the translating system is starting from a highly specified semantic representation which is in most cases (I will discuss the exceptions later on) unchanged by the generator. Obviously, a lot of "deep" organization has already been done by the time such a semantic representation can be produced. To write the original paragraph, for example, the information had to be assembled and, if it was low-level, aggregated. Patterns of reasoning and argument had to be chosen, and decisions about the relative importance of different information had to be made. Moreover, these steps are not necessarily independent, but may be linked in rather complicated ways. The surface generator in the translating system does not, in most cases, have to consider these choices, since they are already specified in the semantic representation. In this sense, then, generation for translation is easier than its more general-purpose counterpart.

I should note that in characterizing generation for translation as general-purpose generation minus some "deep" steps, I do not wish to suggest that the semantic representation used here necessarily represents an intermediate level in the general generation process. In fact, I suspect that in a general-purpose generator, we would not want to create such a highly organized semantic representation while completely ignoring the lexical and grammatical level of the target language. It would not surprise me, then, to see substantial differences between the organization of a general-purpose generator and the generator described here.

In addition to these differences, we can say that if generation for translation can be considered easier than general-purpose generation, in another way it is harder. The generator does not have to decide what to say, but the other half of the coin is that it has to tailor what is generated to the intent of the original text. And this text, of course, was written in a different language. If we can say that languages are organized to convey meaning, we can also say that the organization of particular messages is influenced by the facilities available in a given language. Word choice is an obvious example here - there is often not an exactly equivalent word in the target language. But the problem actually manifests itself at all levels of linguistic organization. When a mismatch between languages occurs, we often have to compromise, suspending one goal to achieve another. It is true that similar compromises must also be made in general-purpose generation, especially in a situation where one is particularly concerned about style. (The worst case here is translating poetry.) The need for such compromise is much more frequent in translation, however, and occurs even in cases where style is relatively unimportant.

6.2 The English Grammar

6.2.1 The Basic Shape of the Generating Grammar

The English generator has three main parts. There is a group of English syntactic specialists for clauses, noun groups, preposition groups, and adjective groups. In addition, a set of routines exists to build, maintain, and inspect a "generation tree," which records the progress of the generating process to date. Finally, there are definition programs associated with the concept markers. Only two standard definition programs are used, but we shall see that they offer a great deal of latitude in the form a definition may take. The generator translates a single sentence at a time, although there is nothing to stop it from using information about the context of a sentence, or from inspecting the text that has already been generated.

Before going on, I should make a few remarks on the scope of the generator. The system has only a few English dictionary definitions at this point, but the routines for writing definitions are quite general. The English syntactic specialists are not as extensive as their German interpretive counterparts. The reason for this was basically one of time, and extending the breadth of the generating routines would be a straightforward process. This is clear first because the existing programs can generate moderately complex sentences: subordinate clauses, conjoined structures, and rankshifted noun groups. Second, the major gaps in the English generator, most notably that it does not deal with questions or commands, can be plugged with code that will look very much like the declarative code that has been written. Declarative, interrogative, and imperative paths through the generator should share a lot of common code, as they do in the German interpretive grammar. In contrast to extending the breadth of the generator, extending its depth, i.e. giving it the ability to make more informed choices, is of course a more difficult task. The syntactic types that were implemented, however, pose enough questions for a start, since in language there are no truly "simple" examples. Finally, the routines for generation tree construction are fully implemented, and they are the subject of the next section.

6.2.2 Additions to PROGRAMMAR

The first step in building the English component was to make two additions to PROGRAMMAR. A mechanism was necessary for building and maintaining a generation tree, and special functions were needed to specify the nodes to be added to the tree. The first task was a relatively simple one - generation tree nodes are defined as carrying the following information:

FEATURES

The syntactic features of the unit

ROOT

If the node is a terminal one, the root of the word translated

PHRASE

The phrase generated for the unit

DAUGHTERS

The daughters of the node in reverse order

SEMANTICS

The semantic node for which the unit was generated

The information at a node may be read using the functions FE-E, PH-E, H-E, and SM-E with the node as argument. Here, the "E" stands for English and is used to maintain the distinction between this generation tree and the German parse tree. Global variables analogous to those for the parse tree are maintained: C-E, H-E, etc., with the obvious meanings. The message variable mechanism is the same, and so, with some added code, are the backup functions POP and POPTO.

A node is added to the generation tree much as in parsing, and the basic function for this is TRANSL. When TRANSL is called a node is set up, and if TRANSL succeeds, the node is added to the tree. Just as with PARSE, a call to TRANSL may specify the name of a clause or group, or the call may request a lexical unit like NOUN or ADJ. Unlike PARSE, the call to TRANSL also contains the name of a node in the semantic representation for which the generation is to be done. Also unlike the parse tree, the daughters of a node on the generation tree do not have to be accumulated in order. It is possible to specify that a node be attached anywhere in the list of daughters of the currently active node. A call to TRANSL with a clause or group name causes a call to the corresponding English generating specialist, and its success or failure determines the success or failure of TRANSL. If TRANSL is called to generate a single word, it executes a procedure associated with the concept marker of the OSS, RSS, or PSS to be translated. These individual procedures will be discussed more fully below.

The generating routines invoked by TRANSL to generate groups look very much like their interpretive counterparts. The basic statement type is still the PROGRAMMAR branch function ":", although limited to a two-way branch, since it is not clear how the three-way branch should be defined for generation. The inspection functions like CQ-E and HQ-E are analogous to those in PARSE, as are F-E, FQ-E, and related functions that add information to a node. At the outset, it is not completely clear that it is necessary to construct a generation tree, but holding on to syntactic information like the location of an adjective or object of a preposition allows the grammar to make decisions based on the structure of the generated sentence at that point. A tree also permits easy incorporation of backup should the generation process run into difficulty.

6.2.3 The Generating Process

Where the interpreter moved left to right over a sentence, the generator moves basically top-down through the semantic representation. That is, the generator moves top-down through the LINKAGE registers, and at each stage, after participants have been translated, it sequences through the MODIFIERS links. An example might be the easiest way to explain the transitions involved. Figure 5.1 shows the semantic representation for:

Der Cephalopod besitzt gelbe Chromatophoren. The cephalopod has yellow chromatophores.

At the top level the generator calls (TRANSL-CLAUSE MAJOR TOPLEVEL SMNODE: RSS1) which in turn calls the syntactic specialist CLAUSE-E. Since these '-E' tags are no doubt distracting, I will ignore them from here on. Any routine mentioned in this chapter is part of the English component unless otherwise noted. The main relation (here, RSS1) is always the topmost link in the semantic representation, so it is specified in the clause call. The first decision in CLAUSE is whether to generate an interrogative, imperative, or a declarative clause, and the CLAUSE-TYPE register of the main relation is checked for its recommendation. Since here the clause type is "statement," the generator follows the declarative path, placing the tag DECLARATIVE in the current node C. The tag TOPLEVEL indicates that the node generated will not have a parent on the tree. Next, CLAUSE calls the phrase routine VERB-PHRASE and which makes the call (TRANSL VERB MVB SMNODE: RSS1). Note that in English, as in German, we will follow Hudson and speak of a verb phrase instead of a verb group. The generator translates the main verb first, since it is the key to the ordering of the other constituents in the clause. A call of TRANSL with MVB will return the verb node and also returns a participants list with the grammatical structures that the verb expects for its participants. The list has been ordered, taking into account the objects required by the verb chosen, the theme of the sentence, and whether the verb is grammatically passive or active. VERB-PHRASE should also go on to generate any auxiliary verbs and supervise tenses, but only the simplest active and passive constructions are handled right now.

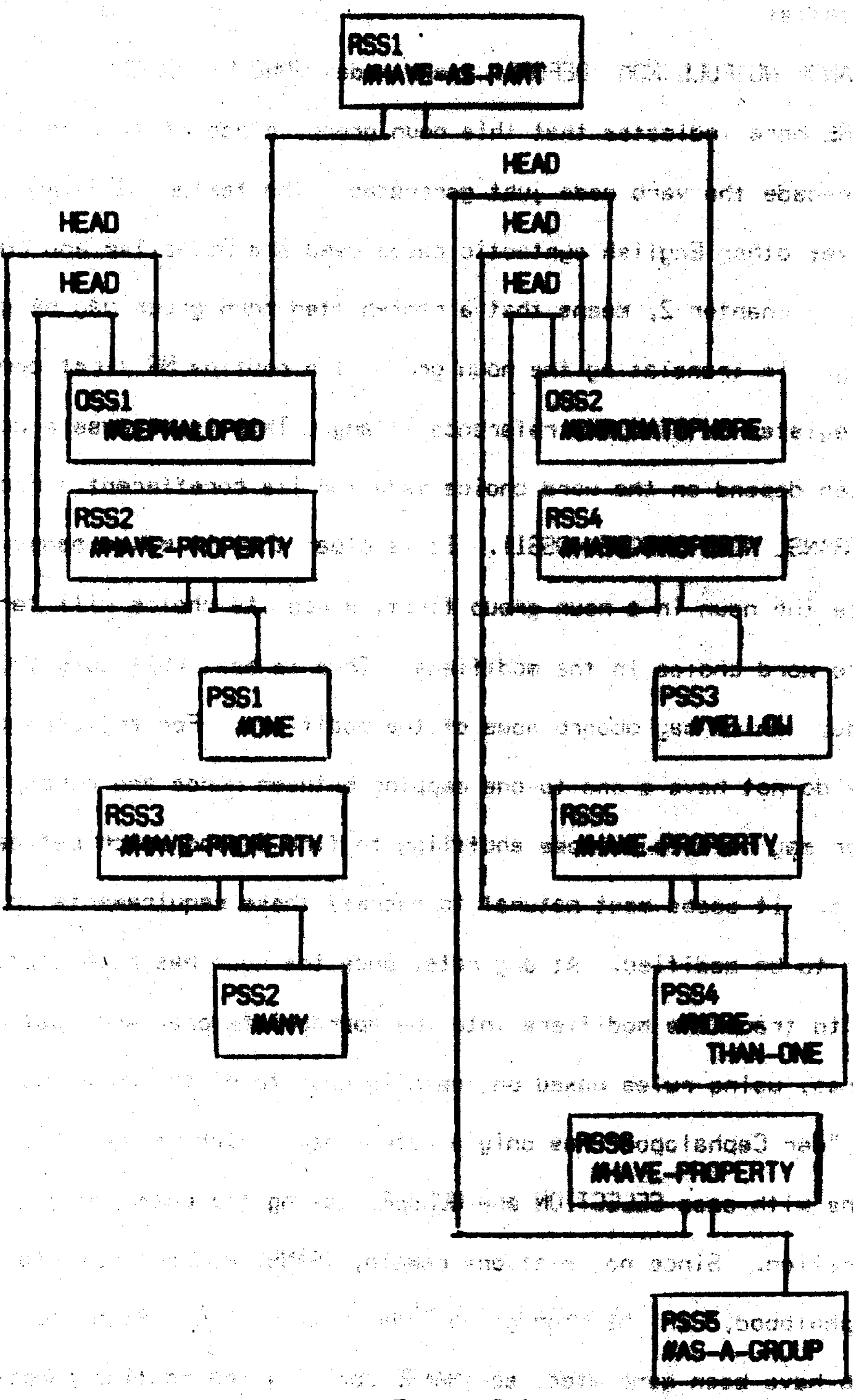

RSS1
#HAVE-AS-PART
HEAD
HEAD
HEAD
HEAD
HEAD
OSS1
#CEPHALOPOD
RSS2
#HAVE-PROPERTY
PSS1
#ONE
RSS3
#HAVE-PROPERTY
PSS2
#MANY
OSS2
#CHROMATOPHORE
RSS4
#HAVE-PROPERTY
PSS3
#YELLOW
RSS5
#HAVE-PROPERTY
PSS4
#MORE-THAN-ONE
RSS6
#HAVE-PROPERTY
PSS5
#AS-A-GROUP

Figure 6.1

The participants list returned by the main verb is in the form of a series of calls, and the clause routine need only execute it. CLAUSE does so now and calls:

(TRANSL NG FULL NOM BEFORE: <verb node> SMNODE: OSS1)

The BEFORE here indicates that this noun group, since it will be the subject, should precede the verb node just generated. The feature NOM indicates nominative; other English syntactic cases used are OBJective and POSSessive. FULL, as in chapter 2, means that a rankshifted noun group may be generated if necessary. In translating the noun group, the routine NG first consults the COREF= register to find the reference if any. This is because a word choice will often depend on the word choice made for its coreferent. From here NG calls (TRANSL NOUN SMNODE: OSS1). It is clear that it makes sense to translate the noun in a noun group first, since its choice will tend to influence word choice in the modifiers. This is most obviously true for compounds, which may absorb some of the modifiers. For any sort of noun, since we do not have a one-to-one mapping between words and concepts, the generator may have to do some shuffling to find a good match between noun and modifiers. It seems most natural to express these requirements in terms of the noun to be modified. At any rate, once the noun has been translated, NG goes on to translate modifiers into the appropriate pre- and post-nominal structures, using rules based on semantic case to do the ordering. Our example "der Cephalopod" has only a determiner, which is generated from relations with case SELECTION and NUMBER, taking the category given-new into consideration. Since no relations remain, TRANSL can collect its results, "the cephalopod," in the noun group node under PHRASE. Both the subject and the verb have been generated, so TRANSL can call the routine AGREE. AGREE makes morphological changes to the verb root so that it will agree with the subject in person and number. Note that in English, agreement is much less complex than in German and that AGREE has a much easier time than its interpretive counterpart INPUT, since it does not have to go searching for an unknown root.

CLAUSE is now ready to generate the next entry in the verb node's shopping list, and it turns out that we want the direct object. The example finishes up as CLAUSE makes a second call to NG with (TRANSL NG FULL OBJ SMNODE: OSS2). NG will generate the main noun from OSS2, then take care of the determiner, and finally call (TRANSL ADJG BEFORE: <noun node> SMNODE: RSS4) to generate the adjective yellow. When the direct object NG call has returned, the generator gathers up the results from the PHRASE entries of the constituents, makes a PHRASE entry in the clause node, decides on punctuation, and returns from CLAUSE.

This was obviously a relatively simple example, and the generating grammar can handle more complex cases. Before further discussion, however, it might be a good idea to take a look at the mechanism for word choice.

6.3 Translation at the Lexical Level

6.3.1 The English Definition Routines

When TRANSL is called to construct a terminal node in the generation tree, it looks up the concept marker associated with the semantic node under consideration and then retrieves an English definition routine from the concept. The definition routines are roughly analogous to the semantic specialist routines in the interpreter. Where the semantic specialists inspect the parse tree in order to build a semantic representation, the English definition routines start from the semantic representation and use special information to not only supply a node for the generation tree (and hence a translation) but also to specify other nodes that will be required if this translation is to be used. The syntactic shopping list supplied by definition routines when a relation is translated is analogous to the global variables bound by the interpretive semantic specialists to supply arguments to relations in the semantic representation.

There are two standard dictionary routines, TRANSL-REL, for translating semantic relations, and TRANSL-OBJ-PROP, for translating objects and properties. The two routines are quite similar, except that more information must be supplied to translate relations. It would probably be helpful at this point to list the information that can appear in a definition. Starred items appear only in definitions for relations, while the rest may be used for all three classes.

ORDER* LEXACT or LEXPASS

TYPE* NONE, ONE, TWO, etc. This specifies the entries in the relation's linkage register that may be left understood.

ROOT the root of the translation

FEATURES: the part of speech of the translation and then a list of other features, which are not now required to match those in the TRANSL call

WORD: specified for irregular forms only

PARTIPICIPANTS:* a variable name or small procedure that specifies the participants list

CONNOTATIONS: currently only one feature and optional

PROBABILITY: number from 1 to 10

CONDITION: a procedure that must evaluate to non-nil if this definition is to be used.

COLLOCATIONS: a list of parts of speech and root of words required. This is currently used only for prepositions and particles.

PROCEDURE: a procedure that is executed if this definition is selected. It allows a concept to be translated by more than one word and does other useful things.

A call to a definition routine would have as its parameter a list of possible definitions, each expressed in terms of these keywords and their values. The possible definitions first go through a preliminary round of elimination based on the part of speech required in the TRANSL call (matched against FEATURES), a match on connotations (If CONNOTATIONS is set in the definition list, it must match the connotations in the semantic node.) and an evaluation of the CONDITION procedure, if there is one. In translating relations, definitions are also screened for ORDER (LEXPASS or LEXACT) and TYPE (NONE, ONE, TWO, etc.) agreement, which were discussed in section 4.2.1. After the preliminary screening, a definition with the highest probability is picked, or, if there are several definitions with this probability, the first one encountered is used. In the English dictionary, the probabilities are expressed as numbers from 1 to 10, and reflect a rough estimate of the order in which the words should be tried. Probabilities could no doubt be better expressed as small procedures that check context and return an appropriate estimate, but no refinements have been made in this direction. Once a word has been selected, its features are added to the feature list of the node. Other definition lists that passed the preliminary screening are placed in the LEXALTLIST register (lexical alternative list) on the node in case backup is necessary later on. If we are translating a relation, its participants shopping list is also attached to the node. Finally, the root of the English word is returned to TRANSL, which finishes building the node, calling morphology routines to add the necessary endings for the individual word.

6.3.2 One Marker, Several Words

The last section presented the simplest case, where a concept marker is translated by a single word. The definition routines must also be able to handle the case in which one marker becomes several words, and the case where several interrelated markers become one word. For situation one, several mechanisms are available. If the marker is a relation, the COLLOCATIONS list is used for prepositions and particles required by verbs, adjectives, etc. The definition procedures access this list, and insert the required word in the correct place in the participants shopping list. For more complicated constructions, PROCEDURE is used. Assume, for the sake of example, that the conceptual structure contained the marker #BEFRIEND, and we wanted to translate the English as "make friends with." One way to do this in the system is with a definition of the form:

```
#BEFRIEND
    (TRANSL-REL ((LEXACT NONE make
       FEATURES: (VERB MVB)
       COLLOCATIONS: ((PREP with))
       PARTICIPANTS:  PREPOBJ-INTRANSA
       PROCEDURE:
         (TRANSL NG NODET PLUR OBJ
          GROUP:  ((TRANSL NOUN PLUR COMMON
                      WORD: friends
                      ROOT: friend))
          SMNODE: <semantic node of the second argument of
              #BEFRIEND>)
```

Here, LEXACT means that the order of the arguments of the verb corresponds to the order of the arguments of the concept marker. NONE means that no marker arguments have been left understood, and <u>make</u> is the root of the main verb. Where a phrase is produced by a definition, the word in the ROOT position corresponds to the part of speech TRANSL would be looking for; in this case, the call would have been (TRANSL VERB MVB). Here, PARTICIPANTS is set to a variable whose value in the system is a series of calls. The PARTICIPANTS would be:

```
(SETQ PREPOBJ-INTRANSA  '((TRANSL NG
                PULL
                NOM
                SMNODE: (CAR ARGSLIST)
                BEFORE: <first verb>)
           (TRANSL PREPG
                SMNODE: (CADR ARGSLIST)
                GROUP:
                ((PREP WORD:   %REQUIRED-PREP%
                      ROOT:   %REQUIRED-PREP%)
                 (NG SIMPLE OBJ))))))
```

ARGSLIST is an ordered list of arguments of the RSS to be translated. The definition procedure will set up a register to allow with to be substituted for %REQUIRED-PREP% when the preposition is generated. Note that here and in PROCEDURE, we see a special sort of call to TRANSL. When ROOT is specified in a TRANSL call along with a part of speech, the definition routines are by-passed, and the node uses the FEATURES, ROOT, and, if given, the WORD supplied in the call. Similarly, if a group or clause is to be translated, then the keyword GROUP can be used, followed by a list of TRANSL calls and other functions. The GROUP feature makes it possible to by-pass syntactic specialists. Note that we do not have to fully specify the words in a group when using these features, for example the preposition group from the participants list above:

```
 (TRANSL PREPG  GROUP:
         ((TRANSL PREP ROOT: %REQUIRED-PREP%
           WORD: %REQUIRED-PREP%)
          (TRANSL NG SIMPLE OBJ))
           SMNODE: <semantic node of second argument of
            #BEFRIEND>)
```

This call says, "Translate a preposition group, looking up the required preposition and using the second argument of #BEFRIEND to produce a noun group." This ability to by-pass definition routines and syntactic specialists is extremely useful, and allows us to write efficient definitions in a relatively economical manner. The one drawback might be the number of features that must be specified along with the words, but it would be relatively easy to incorporate a prompting facility to aid in definition writing, and to package the most common types of PROCEDURE in the same way that the participants lists have been packaged. Ten or fifteen standard patterns would probably handle a good percentage of the different multi-word situations that come up.

6.3.3 Several Markers, One Word

The case just discussed involved one concept marker going to several words. Let us now take a look at case two, where several interrelated concept markers go to a single word. This is done using a combination of the CONDITION and PROCEDURE keywords. CONDITION is used to specify a piece of semantic representation that must match the semantic representation of the

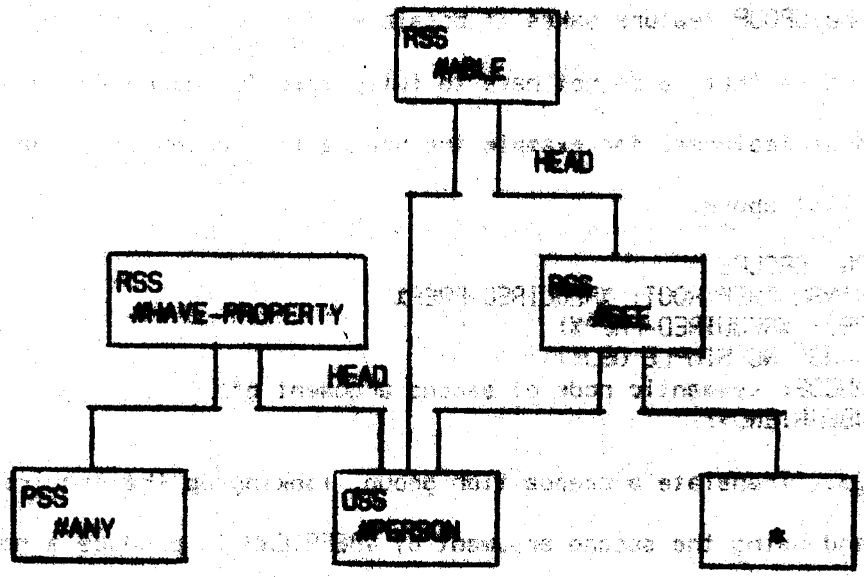


Figure 6.2

sentence, and then PROCEDURE is used to tell the rest of the generator how much of the semantic representation has been absorbed by the particular word. An example of a multi-marker to single word translation is the generation of the adjective visible from the representation shown in Figure 6.2. Here, the star indicates that any semantic node may be in this position. Among other information, the translation routine would contain:

```
#SEE
   visible
      CONDITION:
         (MATCH    (FIND-LINK: #PERSON)
                  (MODIFIER:  #HAVE-PROPERTY #SELECTION)
                  (LINKS: #PERSON #ANY)
                  SM-E
                  (MODIFIER: #ABLE)
                  (LINKS: #PERSON #SEE))
      PROCEDURE:
         (REMOVE  #ABLE #GENERAL  MODIFIERS)
```

When CONDITION is executed, the routine MATCH starts at the place in the semantic representation currently being considered by the generator, in this case the RSS for #SEE. The MATCH routine moves through the MODIFIERS and LINKAGE paths specified, comparing concept markers against the arguments given. If a concept marker alone is not enough to distinguish a modifier, the case may be supplied, as is done for the #HAVE-PROPERTY relation above. LINKS, MODIFIER and FIND-LINK always refer to the arguments of the relation examined just previously. To change the focus of attention, MATCH may be given a semantic node, as was done above. MATCH is currently limited in the kinds of comparisons that can be made, for example it is not now possible to specify that two uses of the same concept marker should refer to different variables. Such extensions, however, would be straightforward. The matching process is not a particularly expensive one, since the ordering of the representation is always fixed, and we know which nodes are to be heads and which modifiers. Note that MATCH is not the only sort of routine that may be used as a condition. CONDITION succeeds or fails depending on whether its associated procedure succeeds or fails.

Once the condition has been satisfied and the definition is chosen, PROCEDURE is executed to tell the generator not to try to translate the modifier #ABLE (where the case here is given as #GENERAL), since this has already been used. The only point worth mentioning about PROCEDURE is that its effects must be limited to the node currently being constructed by the TRANSL call. This is necessary to keep backup simple, since when a node is popped from the tree, we want all the constraints it has placed on the generation process to disappear with it. The limitations on PROCEDURE are currently self-imposed by the dictionary writer, since the system makes no checks on what is being set.

The two definition methods discussed here give the dictionary routines a great deal of power. The method described in the last section allows the generator to translate a single marker into classifier plus noun, or to a whole clause if it is desired; the method in this section allows the generator to handle relation participant nouns such as <u>indication</u>, which are defined as #SOMETHING plus a relation, in this case "something that indicates." We can combine the two methods to translate a piece of semantic representation into more than one English word, which should be useful for idiomatic phrases like "Strike while the iron is hot." Given this generality, the writing of dictionary routines is relatively simple, and it will become easier when more standard patterns and procedures are built into the generator.

6.4 Stumbling Blocks in the Generating Process

In the example used to explain the generating program, the generator was suspiciously successful at each translating attempt. This is not always the case, and several problem situations will be discussed here. The first is the need for backup, then the issue of repetition, and the problem of sentence length.

6.4.1 Backup

There are several cases in which backup is necessary. The most obvious is the case of lexical gaps, or awkwardness, in the target language. In general, if the first translation attempted by the system does not succeed, the generator needs a way to try the alternatives in an orderly manner. The mechanism supplied to handle backup in the generator is a fairly general one, although I have not written enough backup code at this point to give more than an initial report on its performance. The reader should keep in mind that I am discussing a facility provided to allow the designer to incorporate backup into programs, not a full-fledged backup mechanism.

There are two levels in the generator where choices are made: at the lexical level and within the syntactic specialists. There are therefore two sets of lists that are maintained as registers on nodes for backup purposes: LEXALTLIST (the lexical alternative list) and ALTLIST (the structural alternative list). When a definition is evaluated and a word is chosen, other possible definition lists that passed the initial screening described in section 6.3.1 are placed in the LEXALTLIST of the node being created. At the structural level, every choice that might be independently reversible (i.e. would not imply automatic reversal of choices made earlier) leaves tracks on the ALTLIST. These tracks consist of the function in which the choice occurred and a statement label where code for the alternative choices begins.

If the generator runs into difficulty (i.e. all possible TRANSL calls at a given point fail), special backup routines are called. In general, the first step taken will be to try to retranslate the relation for which the problematic semantic node is a participant, if such a relation exists. LEXALTLIST's are used to supply these new alternatives. Backup routines do what they can to keep other participants that have been generated intact, that is, to find lexical alternatives with similar participants shopping lists. If no lexical changes solve the problem, a structural change is attempted. For this, ALTLIST is used, and an attempt is made to reverse the most recent choice first. The offending nodes are popped from the tree, and control returns to the routine and label specified by ALTLIST. The syntactic specialists are structured in such a way that business will proceed as usual after a backup, except that the ALTLIST has been reduced by one possibility.

For the limited number of cases attempted, this backup scheme seems to be general enough. It does rest on the assumption that structural choices can be fully order with respect to each other, and that this ordering will not vary from one situation to another. Only more experience with the generator will determine whether this assumption allows enough generality in the system.

Before going on, I would like to consider the use of backup in the context of generation. It may be that further research will point out ways to cut down on backup by deferring some decisions to later in the process or anticipating others. I do think, however, that backup will remain an important facility in a generator. Unlike interpretation, the possible surface structures for a given semantic representation are not fixed. There can be no question here of generating "all" the possibilities instead of backing up, since with a facility for paraphrase, we could no doubt go on generating possibilities to rather ridiculous extremes. It might be desirable to generate, say, the three most likely lexical possibilities at any given point. This probably would not cause combinatoric explosion, since possible paths would be eliminated quickly. I do not have enough experience with the generator to judge the desirability of this scheme. But no matter what new mechanisms are introduced, a backup mechanism will no doubt remain an important component in a text generator.

6.4.2 Repetitions

The phenomenon of repetition is an essentially stylistic problem; that is, meaning is not affected seriously. Repetitions can, however, be extremely distracting. For example, the first sentence of the sample paragraph contains the phrase "ein deutlich sichtbares Zeichen." The word list for *sichtbar* here contains *visible*, *perceptible*, and *evident*; for *deutlich* the list is *distinct*, *clear*, *plain*, and *evident*. Now as it happens this is translated as "a clearly visible indication," but what if for some reason the generator had chosen *evident* for *sichtbar*? Suppose further that the dictionary only contained one translation for *deutlich* and that one was *evident*. An "evidently evident indication" sounds a little like Gilbert and Sullivan, so the ADJG routine had better retranslate *deutlich* to try to come up with something a little less repetitive.

Not all repetitions are equally annoying. For example, three *the*'s in a sentence would not be noticed at all, while three *however*'s would not be overlooked. In addition, some repetition of clause structure is desirable, e.g. where parallelism is used as a stylistic device. Finally, differences in function tend to influence judgments about what is repetitious.

(i) He went to the zoo to talk to the elephant.

(ii) The problem with meeting with the Board is that Harry won't come.

Here, the two *with*'s in (ii) seem more prominent than the three *to*'s in (i).

This seems to be because the second to heads an adjunct clause rather than a preposition group. The question of which repetitions are repetitious, then, is not as straightforward as it would seem.

In considering repetitions, let us first make the distinction between structural and lexical ones. No efforts were made in the system to handle structural problems like repetitious clause types, repetitious use of conjoined structures, and other relatively high level phenomena. Heuristics to handle these sorts of repetitions would be necessary for a complete system, but I gave this low priority here. This decision was based on the assumption that enough of the variation in the original text would come through, so that sentences in the output would not be carbon copies of each other structurally. As a system became more refined, however, it would need good heuristics, since the different specialists are biased toward particular syntactic types - the ones that they try first - and it would be easy to fall into structural repetitions.

For lexical repetitions, another useful distinction is between function words and the rest. Function words are prepositions, binders, and conjunctions. Since they form a closed set, it would be preferable to handle them by anticipation, rather than backup. Let us look at the system's behavior for prepositional repetitions, the only check of this sort currently implemented. When a preposition is to be generated by the system, the preposition group specialist first looks back through what has been generated, stopping at a sentence boundary or the verb phrase of the major clause, whichever comes first. If other preposition groups are found, a list of the prepositions is made, with those that will be nearest to our new preposition group appearing first. The call to TRANSL is then made, using a tag that we have not yet discussed:

(TRANSL PREP NOT: (prep1 prep2...) SMNODE: <semantic node>)

The NOT: directs the dictionary routine to try to avoid these word choices. If there is no other choice available, the situation is not currently fatal, but the definition routine will try, if possible, to supply a preposition from the end of the list, rather than duplicating a preposition that will be near the new one in the sentence. The NOT feature is currently implemented only for words, but with some revisions of the syntactic specialists, it could also be used to avoid structural repetitions like those described in the last paragraph.

For content words, let us make one final distinction: between elements of coreferent noun groups and other words. The generator does not now worry about repetitions among noun groups, but let me briefly discuss what I think is involved here. The system currently has three options in generating coreferential noun groups. It can either reproduce an entire noun group with <u>the</u> as determiner, it can use <u>this</u> plus the main noun, or it can use a pronoun. That is, the system can refer back to "a big grey octopus" as either:

(i) the big grey octopus

(ii) this octopus

(iii) it

No checks are currently made about other objects in the paragraph that might have similar descriptions. A sophisticated generator would presumably be able to weigh carefully its choices about using reference, balancing distance from coreferent noun groups, possibilities of confusion in the text, and repetitiveness. Any heuristics for repetition among coreferent noun groups, therefore, would have to be integrated into the noun group specialist very carefully.

All the rest of the content words like our "evidently evident" example are handled using backup. The heuristic currently used is that if the root of an adjective, noun, or verb is repeated anywhere within an argument of the main verb or a modifier of the main clause, then an attempt should be made to correct the situation. This involves a preliminary check of LEXALTLIST and then an attempt to replace the node or nodes affected. If this does not work, the repetition is currently left as is, but it would be possible to initiate more extensive backup.

The heuristics used here are obviously very simple and limited ones, and a good deal of refinement will be necessary. Probably the easiest way to do this would be to build in as many basic structural and lexical heuristics as possible, then run the system on some text. This would give the designer a chance to locate the repetitions that actually sound repetitious. The heart of the problem here seems to me to be the analysis, and once good heuristics are found, their implementation should be straightforward.

6.4.3 Those Long German Sentences

Sentence length is a problem that seems trivial until one starts to think about it. Bound up in the question "How long should a sentence be?" are issues of the sentence's role as a carrier of messages. To change sentence length we have to know which information can be safely separated out and which is essential to the integrity of the message. Also involved here are questions about human memory capacities. Any notion of sentential complexity must take into account human short-term memory and processing limitations. Something that is complex to one sort of system might be handled with ease in a slightly different one. Thus, when we try to determine what size a sentence should be, we have left surface generation and are really talking about deep generation: the process of organizing messages.

My system, of course, is based on the assumption that surface generation in the target language will be sufficient. I have assumed that in general we do not need to completely re-build the message for the target language, but rather can generate from the comparatively high level of the semantic representation. In most cases, this seems to be the most efficient way to go, and, since we want to follow the organization of the source text as closely as possible, it seems to be the most effective as well. In dealing with long sentences, I have continued to avoid true deep generation, and I think only further research can determine whether this is a virtue or a vice. Let me discuss long sentences in more detail, and then I will come back to this issue below.

First, I should say that although academic German has a certain notoriety for its long sentences, the sentence length problem is not unique to the languages chosen here. That is, if we were translating French to English instead of German to English, sentence length would still be an issue. Certain things are always easier to express in one language than another; it may take a clause to express in one what a word can express in the other. Because of this, it is possible to start from an uncomplicated sentence in any source language and end up with either a very large and complex translation or a succession of very short sentences that should be combined.

In the system, I have concentrated on dealing with sentences that are too long. This was strictly the result of time limitation, since short sentences can be just as stylistically offensive, and therefore potentially distracting, as long ones. (Although possibly short sentences are not as hard on comprehensibility.) I should also add that the heuristics in the system are very fragmentary and were added more to introduce the issue than to attempt any sort of comprehensive answer to the problem.

The first question that must be asked is "How long is too long?", which should be immediately reformulated as "How complex is too complex?". It is a common observation that iterative structures are easier for people to understand than recursive ones. Ludwig Reiners (38) warns against the dangers of "Klemmkonstruktionen," or successive embedding of clauses. So length alone is not the problem here, although often, of course, excessively complex sentences are also excessively long. Length is one factor in complexity, but not the only one. The recursive structures mentioned, in fact, exemplify one very important contributor to complexity. Whenever a number of things are started but not finished, we can expect the complexity level to be high. Similar, but not identical, is the complexity introduced by back references. If we have, say, three back references in the same sentence, we are not only fighting potential ambiguities, but also the overhead of simply maintaining the links between coreferent items. The complexity introduced by an unfamiliar word or by complicated semantic content must also be reckoned with.

Sentential complexity seems to be an additive phenomenon. Some sentences that I consider too complicated in English seem to be difficult not because they contain a single complicated structure, but rather as a result of a compounding of complexity. To see what I mean by this, consider our prize example sentence and a translation that parallels the sentential organization of the German:

> (i) Ein deutlich sichtbares Zeichen für die im Nervensystem verlaufenden Eregungen ist das Spiel der Chromatophoren der Cephalopoden, jener unter der Haut liegenden gelb, braun, schwarz, violett oder karminrot gefärbten Zellen, die sich entweder zusammenziehen oder durch radiäre ansetzenden Muskeln flächenhaft ausgebreitet werden können.

> (i) A clearly visible indication of the excitations that run through the nervous system is the play of the chromatophores of the cephalopod, those yellow, brown, black, purple or carmine colored cells that lie under the

skin, and which either contract or can be spread out under the skin surface by radially affixed muscles.

This translation would sound better if two English sentences were used, but what about the following?

> (ii) Ein interessantes Beispiel davon ist das Spiel der Chromatophoren, jener unter der Haut liegenden gelb, braun, schwarz, violett oder karminrot gefärbten Zellen, die sich entweder zusammenziehen oder durch radiär ansetzenden Muskeln flächenhaft ausgebreitet werden können.
>
> (ii) An interesting example of this is the play of the chromatophores, those yellow, brown, black, purple or carmine colored cells that lie under the skin, and which either contract or can be spread out under the skin surface by radially affixed muscles.

To me, the English in (ii) is fine as a single sentence. Furthermore, I am not overly disturbed by a sentence like:

> (iii) A clearly visible indication of the excitations that run through the nervous system is the play of the chromatophores of the cephalopod, those yellow, brown, black or carmine colored cells that lie under the skin.

Therefore, it seems to me that no one structure (say, an embedded one) can be considered complex except with respect to its context. One more clause can tip the balance in an otherwise acceptable sentence, but we can not pin the blame on the clause alone: it is the whole sentence (at least) that contributes to judgements about complexity.

It is not always easy, then, to decide which sentences should be broken down into smaller ones. In this respect, I think people have an easier time of it than machines, since they have a very direct method of determining complexity: if the mind boggles, it's too complex. Computers, whose minds are not structured in the same way, don't boggle in the same way. But even if we lack a model of human memory and processing, I think it is possible to derive a measure for sentential complexity. By analyzing sentences judged by people to be too complex, it should be possible to come up with a formula - not necessarily a simple one, though - to predict which sentences are acceptable and which are not.

A look at the specific cases handled in the system would probably be helpful, but before the "what," I must again consider the question "when." When should the translating system make these checks on sentence, or more properly message, length? There seem to be several options. First, there is the deep generation approach. If we start our generation process from a set of assertions from the data base of the deductive component, we can include limitations on how large or deep a semantic representation should be, or what patterns of linkage can occur. Alternatively, we can retain a surface generation scheme but do complexity analysis on the semantic representation. We would be using the same sort of information (although probably not identical programs) as for the deep generation approach, but we would be analyzing a structure after it has been built, not in the process of building it. The third approach is for the surface generator to monitor itself. If the output has become too complex, then suitable action can be taken to split the semantic representation, possibly reformulating some part of it in the process.

I chose the third alternative for the system, primarily because complexity of expression is language-dependent, while the semantic representation is a gesture toward language independence. In a simple inspection of the semantic representation, we cannot tell whether a relation can be expressed by a word, or whether a clause will be necessary. In the present system, it is easy enough, of course, to determine whether single words do exist for a given concept. The point is, however, that we do not know whether a particular word can be used in the translation until we take into account both the interrelationships between parts of the message and the linguistic constraints on structuring in the sentence.

To bring the discussion down to a more concrete level, let us continue with the example sentence given and decide what factors contribute to its complexity. The heuristics presented here are clearly *ad hoc*, and they represent the only case currently handled by the system. I think they suggest, however, the direction in which one should proceed. The English example, again, was:

> (i) A clearly visible indication of the excitations that run through nervous system is the play of the chromatophores of the cephalopod, those yellow, brown, black, purple or carmine colored cells that lie under the skin, and which either contract or can be spread out under the skin surface by radially affixed muscles.

One factor here seems to be the subject of the sentence: "a clearly visible indication." We notice that indication is a relation participant noun, and that the other argument of the relation #INDICATE (the thing indicated) is present as a preposition group ("of the excitations"). We can call a simple nominalization (e.g. any gerund) or relation participant noun "saturated" if all the arguments of the relation appear explicitly in the noun group. The subject of sentence (i) is therefore saturated. Further, if a saturated noun group contains a postnominal rankshifted qualifier modifying one of the arguments of the relation, we can call it "supersaturated." The subject in (i) is also supersaturated, due to the RSQ clause that modifies excitations ("that run through the nervous system"). We would expect a supersaturated noun group to be a key contributer to sentential complexity, but note that supersaturation alone is not enough to cause rejection of a sentence (example (ii) above).

A further difficulty with sentence (i) is the appositive phrase that starts with "those yellow, ... and carmine colored cells." Here, two postnominal RSQ clauses modify cells, and in English we would tend to conjoin the two using and. Furthermore, the second postnominal clause here ("which either contract or can be spread out ...") is itself conjoined by either / or. This double level conjunction looks like an important contributor to complexity, but it is difficult to tell where to draw the line. If our second clause were "and which either expand or contract," the appositive would probably not be too long. In (i), however, the two clauses are not parallel. (The first is active, and the second is passive.) This seems to push the appositive over the line. Let us say, then, that an appositive can be considered complex if it goes beyond two levels of parallel conjunction, or beyond one level of non-parallel conjunction. Once again, a complex appositive alone should not trigger rejection of a sentence, since example (ii) above does not seem to be too complex.

By combining a supersaturated noun group with a complex appositive as defined above, we do get a sentence that is too complex. In dealing with such a sentence, we would expect a generator to interrupt processing if it detects an output that satisfies both complexity conditions. In the case of example (i), nodes would then be popped from the tree back to the beginning of the appositive, since appositives and conjunctions are natural places to break up a sentence. What is left on the tree (in this case, everything up to "the chromatophores of the cephalopod") would become a single sentence. The appositive part of the semantic representation would be detached and would become the second sentence, with generation starting from the top. Note that given the similarity in word order between English major and secondary clauses, it might be feasible to remodel some of the output that was already generated before the interrupt, instead of discarding it altogether. In some cases, however, this patching might require a certain amount of ingenuity.

Note that not all long sentences can be split as easily as the example here, and, for some, we might not be able to find a division that maintains the integrity of the message. Still, I think this example gives an idea of some of the issues involved in handling sentential complexity. I should also remark that this emphasis on using syntactic criteria to formulate heuristics may be misguided. It may be that semantic factors should be considered heavily. For translation, however, I would tend to favor heuristics expressed in terms of the surface structure of the target language, since a given semantic representation has already been embodied in a single sentence in the source language.

6.5 Other Necessary Extensions

Even if it were extended to handle the full range of syntactic structures, the generator discussed here could not be said to be a complete one. There are still some very important processes that have not been considered, and I would like to discuss them briefly in this section. Nothing discussed here is currently implemented, primarily because a great deal of additional apparatus would be necessary.

6.5.1 Dealing with Ambiguity

Ambiguity is not an issue for the generator in the same way that it is for the interpreter, but there are several questions worth considering. Two mechanisms for handling different sorts of ambiguity will be discussed here; one of these would be desirable in a working translating system and the other would be quite important.

The first feature is the ability to translate ambiguity-for-ambiguity, which was mentioned in the introduction. I think this would be a useful mechanism since I find myself doing it occasionally when hand translating. For people, such an ability is used most for pronoun reference and implicit information like understood agents, etc. For a translating system with a weak deductive component, the ability to translate ambiguity-for-ambiguity might also be used to avoid choosing between different senses of a word. Especially in languages as close historically as German and English, it is often possible to find a word in one language that is ambiguous in the same way in the other language. In general, a system would have to be quite sophisticated to translate ambiguity-for-ambiguity, but at the lexical level, we could make a start with either an English interpretive dictionary or with an associative ability to link concepts to definitions. (Curreny in the system, we can ask what concept markers are associated with a German word or what English words are associated with a concept marker, but there is no way to find out easily what concept markers are associated with English words.) It would be a relatively simple matter to take two concept markers produced for a German word, look up a set of English words associated with one of them, and see if any English words in this set could also have the other concept marker as a meaning.

To translate ambiguity-for-ambiguity beyond the lexical level, the generator would need a model of the English interpretive process. The task here would be to analyze the way the target language was ambiguous by inspecting the semantic representation or by accessing pre-packaged knowledge. The pre-packaged knowledge might express such facts as, "In a German nominalization of a transitive verb, the genitive could be either the subject or the direct object of the verb, if no other participants are given." The second phase of this task would then be to find an English structure with a similar ambiguity, which for this case happens to be an English nominalization with an <u>of</u>. To use a familiar example,

das Schiessen der Jäger = the shooting of the hunters

("the hunters shoot" or "the hunters are shot")

The model of structural ambiguities might be derived automatically from a parser of the language, although that would not be easy for the particular parser used here. Wherever it comes from, however, we would want such knowledge about ambiguities to be detached from the parser, since we do not want to have to simulate the parsing process every time we want to determine whether a structure is ambiguous.

The ability to translate ambiguity-for-ambiguity just discussed would be attractive in a generator, but not essential. The second feature I will discuss here, anticipating ambiguities in the output text, is more crucial, I think, since it relates to the reliability of a translating system. What I would like to consider is the situation where the generator unwittingly produces a syntactically or semantically ambiguous sentence. The most extreme step one could take to avoid this problem would be to feed every text output back into an English interpreter, to see if what was produced contained serious ambiguities. This would correspond to a translator reading over and correcting a translation. It would not be an absolute assurance, since the proof-reader would presumably share the same deductive data base used by the rest of the system, and knowledge limitations might cause it to miss ambiguities that are present in the output text.

Even if such a proof reading facility were available - and especially if one were not - a generator should also be able to anticipate some ambiguities and avoid them. In his thesis (19), Hill catalogs four causes of global syntactic ambiguity in English:

(1) Choosing between participle and gerund

(2) Choosing between noun and verb in clause first word position

(3) Choosing the correct transitivity for the verb

(4) Choosing an attachment point for a modifying phrase

It is clear that, given the features saved on the generation tree, checks for these sorts of ambiguities would be relatively straightforward. A generator also might be able to anticipate certain semantic confusions about the scope of a quantifier. ("Quantifier" is used here as an English part of speech; see Winograd (39,p.67).) Checks for these sorts of ambiguities could be built into the syntactic specialists of a generator, or the same information could be embedded in a small routine that would monitor the generation process and interrupt if one of these ambiguous structures were generated.

6.5.2 When All Else Fails

In the implementation as it stands now, if all the backup possibilities are exhausted for a particular sentence, the generator simply fails. This would obviously be undesirable behavior in a working translation system, and we would want a system to be able to make the best out of a bad situation. There seem to be two directions one could go to meet this goal, one being compromise and the other paraphrase. Both features would be desirable in a system, but it is not at all clear how such behavior could be produced. The two will be considered briefly here, but no solutions will be offered, since they depend, I think, on extensive further research.

In the current version of the generator, backup is handled by trying alternative choices, but this is always within the context of a set of fixed choices; no attempt is made to suspend rules. In actual situations, it is entirely possible that no combination of permissible choices adequately translates the original, and, in this case, we would want to produce the best approximation possible. This could be done either by leaving out some of the content of the original, or by violating one or more of the rules of the target language. (The other alternative is paraphrase, which is discussed below.) In general, when translating scientific texts we will choose to violate rules rather than to leave out content. The choice of which rules to suspend, however, can be a difficult one, and in arriving at this decision, we might get involved in delicate trade-offs between different possibilities. The generator would probably need the ability to produce a set of alternative trees for which different rules had been suspended and then use some set of criteria to determine which represented the best compromise. This ability to make compromises would be important in a working generator, since it often seems that the essence of translation is the ability to find good compromises.

If no satisfactory compromise can be found, the next step is to try paraphrase. I am using "paraphrase" in a special sense here to mean a change in the explicit meaning, although not in the total meaning of a text. Let me give a simple example of a situation in which explicit meaning differs but in which total meaning is equivalent. The English phrase "a clear day" appears in German as "ein heller Tag" (literally, "a bright day"). Obviously, if the sky is clear of clouds, then the sun can be expected to shine brightly, and if the sun is bright, we expect the sky to be clear. The two languages pick up on different ends of this if-and-only-if relationship. But, although explicit statements differ, the implication is still roughly equivalent.

This particular example would probably best be handled in dictionary definitions, by translating <u>hell</u> as <u>clear</u> under certain circumstances; we probably would not use a general paraphrase mechanism. Not all possible paraphrases, of course, can be anticipated in this way. If the generator cannot translate a phrase, we will want it to consider the implications to see if another equivalent phrase can be found. This will involve a return to the deductive component, since only a fraction of the implicit meaning of a sentence is carried in the semantic representation. The deductive component would presumably find an equivalent paraphrase, alter the semantic representation, and send this new representation back to the generator for another try.

***

Although the generator currently implemented is very limited in scope, I have tried to foresee the kinds of extensions that would be necessary. The important features of the generator are that it uses the semantic representation as input and that it is not constrained to generate components in the linear order that they will appear in a sentence. With the extensions made to PROGRAMMAR, we can maintain a generation tree, and it is possible to by-pass syntactic specialists or definition routines, to specify a definition list to be used, or to explicitly rule out particular word choices. Emphasis has been placed on some problems that are traditionally considered stylistic, but which are of considerable importance for translation. A great deal more analysis needs to be done on the semantic motivation for particular syntactic choices, and the generator would also benefit from investigations of problems such as repetitiveness and sentence length. Finally, the issues discussed in section 6.5 - avoiding ambiguity, suspending rules, and paraphrase - are problems that are wide open for further research.

Chapter 7 -- Conclusions

In the course of describing the implementation, I have discussed some problems and some solutions. Not all of the problems have been satisfactorily solved, and among these are some, I think, that are interesting enough to justify more intensive research. In this final chapter, let me review what I consider to be the major problems encountered in this project and make some remarks about the different solutions proposed.

Heading the list of problems in mechanical translation is still, of course, the problem of understanding. This was outside the scope of the project, but I want to emphasize again here that true understanding of the source text is crucial to trustworthy translation. Related to this is the issue of accountability. A user should always be able to ask a system what choices were made and why. Just as a human translator could give reasons for a particular disambiguation or word choice, I think it is essential that a system be able to do the same. This will not guarantee reliability, but it does give the user some control by giving him a chance to catch gaps in the knowledge base, incorrect assumptions, etc.

The first problem encountered in the implementation was that of parsing German text. Here, there were two difficult areas - German inflection and the relatively wide (compared to English) syntactic variety, i.e. prenominal clauses, the relative freedom of word order for verb objects, end-order constructions, etc. The former involved changes to PROGRAMMAR to handle multiple feature lists. These changes were extensive, although of a routine nature. If PROGRAMMAR had originally been written to handle Russian or Icelandic, the morphology of German would not have come as such a shock. English is biased almost exclusively toward word order in the linguistic trade-off between morphophology and word order, so one would expect to need fairly extensive changes in order to handle a more heavily inflected language.

With respect to actual parsing performance, I would say that the general approach used by Winograd is as good at handling German as it is at handling English. That is to say, the performance of the parser leaves no doubt that it could someday be extended and embedded in a practical system. There remain certain trouble spots, however, that are present in English but are exacerbated in German. The implementation uses a number of ploys to deal with the more varied syntactic choices of German, but it is not clear that the solution is general enough. The recent dissatisfaction expressed about backup in language parsing seems to be well-founded, and it will be interesting to see the results in this area.

A key question in mechanical translation is what the input to the generator should look like. In designing the implementation, I started with two assumptions about this issue. The first was that for a given target language we can predict the sorts of information that will be necessary for generation and the sorts that will not. Second, I assumed that surface generation would be enough, that the generator could follow the general organization of the source text sentence for sentence. These are based, in turn, on the underlying hypothesis that translation of scientific prose does not need the full power of a general purpose generator. Adopting these assumptions resulted in a commitment to the use of a semantic representation as input to the generator. In chapter 6, we saw that these assumptions do not always hold. Some situations require paraphrase, and in others we might have to restructure the message entirely. I think they are true often enough, however, to justify substantial differences between the form of generators for translation and general purpose generators. I could be wrong in this, however, and only increased research will tell whether efficiency lies in the direction of a special purpose or of a general generating process.

There are many aspects of the two languages considered here that I would like to have investigated in more detail. All of these, I think, are interesting research areas in their own right, irrespective of the implementation involved. Only a token gesture was made toward using information about collocations for parsing and generation. It would be interesting to see how this information could be used to build up lexical expectations about the rest of the sentence, in addition to the syntactic expectations currently embedded in the parser. Furthermore, the area of generation poses a number of interesting questions, many of which have been given only rudimentary answers here. Issues of word choice and sentence length deserve more attention. A good deal more analysis needs to be done on questions of semantic motivation for surface structure choices. Finally, another very interesting problem is that of suspending rules to make good generating compromises.

Throughout this project, I have been continually impressed by both the economy of natural language as a communication medium and the variety of its mechanisms. I find this convincing evidence that any translating system that throws away information, be it syntactic, lexical, or semantic, cannot hope for success. In the end, only a total approach to language will offer even an initial solution to the translation problem, and a lot of intriguing questions still remain unanswered.

REFERENCES

1. <Bar-Hillel 1964> Bar-Hillel, Jehoshua, LANGUAGE AND INFORMATION, Addison Wesley, Reading, Mass., 1964.

2. <Charniak 1972> Charniak, Eugene, "Toward a Model of Children's Story Comprehension," AI TR-266, Artificial Intelligence Laboratory, M.I.T., December, 1972.

3. <Contrastive 1972> Contrastive Semantics Project, First Technical Report, Department of Linguistics, University of California, Berkeley, October, 1972.

4. <Duden 1966> DUDEN-GRAMMATIK: GRAMMATIK DER DEUTSCHEN GEGENWARTSSPRACHE, Der grosse Duden, Vol.4, Grebe, Paul (ed.), Bibliographisches Institut, Mannheim, 1966.

5. <Fillmore 1968> Fillmore, C.J., "The Case for Case," in Bach and Harms (eds.), UNIVERSALS IN LINGUISTIC THEORY, Holt, Rinehart, and Winston, New York, 1968.

6. <Fodor 1963> Fodor, J.A. and Katz, J.J., "The Structure of a Semantic Theory," in THE STRUCTURE OF LANGUAGE, Prentice Hall, Englewood Cliffs, N.J., 1964

7. <Frey 1973> Frey, Eberhard, "Tendenzen in der deutschen Nominalflexion," MUTTERSPRACHE, 83, September-October 1973.

8. <Glinz 1962> Glinz, Hans, DIE INNERE FORM DES DEUTSCHEN, Francke, Bern, 1962.

9. <Goldstein 1973> Goldstein, Ira, "Understanding Fixed Instruction Turtle Programs," Doctoral Thesis, Dept. of Mathematics, M.I.T., September, 1973.

10. <Halliday 1961> Halliday, M.A.K., "Categories of the Theory of Grammar," WORD 17, 1961.

11. <Halliday 1966a> Halliday, M.A.K., "Some Notes on 'Deep' Grammar," JOURNAL OF LINGUISTICS 2, 1966.

12. <Halliday 1966b> Halliday, M.A.K., "The English Verbal Group: A Specimen of a Manual of Analysis," Nuffield Programme in Linguistics and English Teaching, Work Paper VI, 1966.

13. <Halliday 1967a> Halliday, M.A.K., "Some Aspects of the Thematic Organization of the English Clause," Memorandum RM-5224-PR, Rand Corporation, Santa Monica, January, 1967.

14. <Halliday 1967b> Halliday, M.A.K., "Notes on Transitivity and Theme in English," JOURNAL OF LINGUISTICS 3, 1967.

15. <Halliday 1970a> Halliday, M.A.K., "Functional Diversity in Language as

Seen From a Consideration of Modality and Mood in English," FOUNDATIONS OF LANGUAGE 6, 1970.

16. <Halliday 1970b> Halliday, M.A.K., "Language Structure and Language Function," in Lyons (ed.), NEW HORIZONS IN LINGUISICS, pp. 140-165.

17. <Hempelmann 1926> Hempelmann, Friedrich, TIERPSYCHOLOGIE VOM STANDPUNKT DES BIOLOGEN, Akademische Verlagsgesellschaft, 1926, pp. 197-8.

18. <Hewitt 1972> Hewitt, Carl, "Description and Theoretical Analysis (Using Schemata) of PLANNER: A Language for Proving Theorems and Manipulating Models in a Robot," AI-TR-258, Artificial Intelligence Laboratory, M.I.T., April, 1972.

19. <Hill 1972> Hill, Jeffrey, "Backup Strategies for Resolving Ambiguity in Natural Language Processing," Masters Thesis, Dept. of Electrical Engineering, M.I.T. May, 1972.

20. <Hudson 1971> Hudson, R.A., ENGLISH COMPLEX SENTENCES, North Holland, Amsterdam, 1971.

21. <Levine 1973> Levine, David R., "Computer-Based Analytic German Grammar Instruction," Technical Report No. 199, Institute for Mathematical Studies in the Social Sciences, Stanford University, March, 1973.

22. <Lyons 1970> Lyons, John (ed.), NEW HORIZONS IN LINGUISTICS, Penguin Books, Middlesex, England, 1970.

23. <Marcus 1974> Marcus, Mitchell, "Wait-and-See Strategies for Parsing Natural Language," Working Paper 75, Artificial Intelligence Laboratory, M.I.T., August, 1974.

24. <Martin 1973> "Translation of English into MAPL Using Winograd's Syntax, State Transition Networks, and a Semantic Case Grammar," Internal Memo 11, Automatic Programming Group, Project MAC, M.I.T., April, 1973.

25. <McDermott 1972> McDermott, Drew V. and Sussman Gerald J., "The Conniver Reference Manual," A.I. Memo No. 259a, Artificial Intelligence Laboratory, M.I.T., May, 1972.

26. <McDermott, 1973> McDermott, Drew V., "Assimilation of New Information by a Natural Language-Understanding System," AI TR-291, Artificial Intelligence Laboratory, M.I.T., February, 1974.

27. <Minsky 1973> Minsky, Marvin, "A Framework for Representing Knowledge," AI Memo No. 306, Artificial Intelligence Laboratory, M.I.T., June, 1974.

28. <New 1965> The New Cassell's German Dictionary: German-English English-German, Betteridge, Harold T. (ed.), Funk & Wagnalls, New York, 1965.

29. <Raphael 1964> Raphael, Bertram, "SIR: A Computer Program for Semantic Information Retrieval," in Minsky (ed.), SEMANTIC INFORMATION PROCESSING, M.I.T. Press, Cambridge, Mass., 1968.

30. <Reiners 1963> Reiners, Ludwig, KLEINE STILFIBEL, Deutscher Taschenbuch Verlag, Munich, W.Germany, 1963.

31. <Robins 1964> Robins, R.H., GENERAL LINGUISTICS, Indiana University Press, Bloomington, 1964.

32. <Rubin 1973> Rubin, Andee, "Grammar for the People: Flowcharts of SHRDLU's Grammar," A.I. Memo No. 282, Artificial Intelligence Laboratory, M.I.T., March, 1973.

33. <Rustin 1973> Rustin, Randall (ed.), NATURAL LANGUAGE PROCESSING, Algorithmic Press, New York, 1973.

34. <Sussman 1973> Sussman, Gerald J., "A Computational Model of Skill Aquisition," AI-TR-297, Artificial Intelligence Laboratory, M.I.T, August, 1973.

35. <Vendler 1968> Vendler, Zeno, ADJECTIVES AND NOMINALIZATIONS, Papers on Formal Linguistics, No. 5, Mouton, The Hague, 1968.

36. <Whorf 1956> Whorf, Benjamin Lee, LANGUAGE, THOUGHT, AND REALITY, M.I.T. Press, Cambridge, Mass., 1956.

37. <Wilks 1973> Wilks, Yorick, "The Stanford Machine Translation Project," in Rustin (ed.), NATURAL LANGUAGE PROCESSING, pp. 243-290.

38. <Winograd 1969> Winograd, Terry, "PROGRAMMAR: A Language for Writing Grammars," A.I. Memo No. 181, Artificial Intelligence Laboratory, M.I.T., November, 1969.

39. <Winograd 1972> Winograd, Terry, UNDERSTANDING NATURAL LANGUAGE, Academic Press, New York, 1972.

40. <Woods 1973> Woods, W.A., "An Experimental Parsing System for Transition Network Grammars," in Rustin (ed.), NATURAL LANGUAGE PROCESSING, pp. 111-154.

APPENDIX A. WORD FEATURES

WORD DEFINITIONS FOR THE INTERPRETIVE GRAMMAR REQUIRE THE FOLLOWING SYNTACTIC INFORMATION.

STARRED FEATURES INDICATE REQUIRED MATCHES IN INFLECTION. FEATURES WITH DOTS INDICATE TYPES OF FEATURES:
PERSON= P1ST P2ND-FAM P2ND-POL P3RD
GENDER= MASC FEM NEUT
CASE= NOM GEN DAT ACC
NUMBER= SING PLUR

ADJECTIVE, EITHER ONE THAT CAN BE DECLINED OR NOT,
COMP =FORMS COMPARATIVE
SUP = FORMS SUPERLATIVE
IF A SUPERLATIVE OR COMPARATIVE IS ACTUALLY FOUND THEN THE FEATURES SUPERL AND COMPAR ARE ADDED.

ATTRIBUTIVE ADJECTIVES:
```
(ADJ  ATTR   DECL  COMP SUP)
```

COMPLEMENTS:
```
(ADJ  NODECL  COMP SUP .CASE. )
                       NOOBJ
```

ADJECTIVES THAT MODIFY VERBS OR OTHER ADJECTIVES:
```
(ADJ  RELMOD NODECL COMP SUP )
```

THOSE RELMOD ADJECTIVES THAT MAY NOT APPEAR IN THE FIRST POSITION IN THE SENTENCE:
```
(ADJ  RELMOD NON-FRONTAL)
```

POSTNOMINAL ADJECTIVES:
```
(ADJ  POSTNOM NODECL)
```

WAS FÜR:
```
(ADJ INTER NODECL .CASE.)
```

WO, WARUM, WOHIN, ETC.:
```
(ADJ INTER)
```

WORÜBER, WORAUS, WOZU, ETC.:
```
(ADJ INTER WO-FORM)
```

ADVERB MODIFYING ADJECTIVES AND OTHER ADVERBS:
```
(ADV)
```

BINDERS:
```
(BINDER)
```

COORDINATING CONJUNCTIONS:

(CONJ)

DER DIE DAS ETC.:
(DET DEF .GENDER. .CASE. .NUMBER.)

DIESE JENE ETC.:
(ENDINGS DETERMINE GENDER, CASE & NUMBER)
(DET DEMADJ)

```
EIN KEIN EINIGE:
(ENDINGS DETERMINE GENDER, CASE, & NUMBER)
(DET INDEF SING)
           PLUR
```

WELCHE:
(DET INTER DECL)

WESSEN:
(DET INTER NODECL P.GEN)

MEIN DEIN ETC.:
(ENDINGS DETERMINE GENDER, CASE & NUMBER. SEE POSS-SUBST FOR P.)
(DET POSS P.PERSON. P.GENDER. P.NUMBER.)

JA, NEIN, DANKE, AHA:
(INTERJECTION)

STRONG NOUNS, THAT IS THOSE THAT HAVE A REGULAR DECLENSION:

```
(NOUN STRONG .GENDER. GEN-ES  NOPLUR )
                      GEN-S   PLUR0
                              PLUR-EN
                              PLUR-E
                              PLUR-ER
                              PLUR-N
                              PLUR"E
                              PLUR"ER
                              PLUR"
```

IF THE NOUN MAY TAKE MORE THAN ONE GENITIVE OR PLURAL ENDING, THESE ARE LISTED SIDE BY SIDE IN THE FEATURE LIST, RATHER THAN IN SEPARATE ENTRIES.

WEAK NOUNS, LIKE SOLDAT, MENSCH, ETC.:
(NOUN WEAK .GENDER.)

MIXED NOUNS:
(NOUN MIXED .GENDER.)

NOTE: ALL NOUNS ALSO MAY BE EITHER COUNT, MASS OR PROPN (PROPER NOUN). SHOULD A PARTICULAR NOUN BELONG TO MORE THAN ONE OF THESE CATEGORIES THE FEATURES WILL BE LISTED TOGETHER IN THE SAME WAY THAT THE TRANSITIVITY PROPERTIES ARE FOR THE VERB.
WEAK AND MIXED NOUNS MAY INCLUDE GENITIVE ENDINGS IN IRREGULAR CASES.

CARDINAL NUMBER:

```
(NUM)

DIE MEINIGE, DIE MEINE, ETC.
INFORMATION PREFIXED BY 'P' REFERS TO FEATURES OF THE PRONOUN ITSELF, WHILE
UNPREFIXED INFORMATION (GENDER CASE & NUMBER) WILL BE ADDED WHEN THE ENDING IS
EVALUATED.
(POSS-SUBST  P.GENDER. P.PERSON. P.NUMBER )

PREPOSITION:
(PREP  .CASE.  PRE )
               POST

MAN JEMAND ETC.:
(ENDINGS SUPPLY CASE & NUMBER)
(PRON ABSTRACT  INDEF  )

WER, WAS:
(PRON INTER .CASE.)

MEIN- ETC.
(SEE POSS-SUBST FOR EXPLANATION OF P.  ENDINGS DETERMINE GENDER, CASE, +
NUMBER.)
(PRON POSS  P.GENDER. P.PERSON. P.NUMBER.)

DA-COMPOUNDS:
(PRON PREP RELMOD)

ER SIE ES ETC.:
CASE MAY BE EITHER (NOM GEN DAT ACC REFL WEMF)
(PRON  PERS   DEF  .PERSON.  .GENDER.  .CASE.  .NUMBER. )

DAS, WAS, DA:
(PRON PERS RELMOD)

EIN, KEIN:
(ENDINGS SUPPLY CASE & NUMBER)
(PRON  PERS  INDEF )

DER, DEREN, WELCHE, ETC,:
(PRON REL .CASE. .GENDER. .NUMBER.)

SELBST, ALL, ETC.:
(QUANT    MASS )
          COUNT

PARTICLE USED AS SEPARABLE PREFIX:
(SEPPR)

VERB:
(VERB REG    UML   SEIN   .TRANSITIVITY.      -SEPPR .SEPPR.)
      IRR    ---   HABEN                      +SEPPR
      MIXED                                   INSEPPR
                                              ---
```

IF -SEPPR, THEN THE SEPARABLE PREFIX IS GIVEN. FOR SEPARABLE PREFIXES, BOTH PARTS OF THE WORD MUST BE ENTERED SEPARATELY IN THE DICTIONARY, ALTHOUGH SEMANTICS NEED ONLY BE HUNG ON THE COMPOUND. THE COMPOUND ENTRY IS LABELED WITH +SEPPR AND DOESN'T NEED THE INDIVIDUAL SEPPR SPECIFIED, EITHER IN FEATURE LIST OR COLLOCATIONS.

TRANSITIVITIES ARE:
A+D R+A W+A A+G A+A R+D R+G A+P A+E R+E Z A D R G N W P E I

I=INTRANSITIVE A=ACCUSATIVE D=DATIVE G=GENITIVE N=NOMINATIVE R=REFLEXIVE W=(FOR WEMFALL) DATIVE REFLEXIVE P=PREPOSITION AS OBJECT Z=RANKSHIFTED NOUN GROUP E=ANY ADVERBIAL. SOME VERBS HAVE OBLIGATORY LOCATION, TIME, ETC. JUST WHICH CASE APPLIES IS SPECIFIED ROUGHLY IN SEMANTIC RESTRICTIONS.

IF A VERB HAS MORE THAN ONE TRANSITIVITY RELATION, THESE ARE INCLUDED IN PARENTHESES IN ONE DEFINITION, RATHER THAN MAKING SEPARATE LISTS FOR EACH ONE. THESE LISTS ARE THEN EXPANDED AUTOMATICALLY WHEN ENCOUNTERED.

APPENDIX B. SAMPLE PARSE

EIN DEUTLICH SICHTBARES ZEICHEN FÜR DIE IM NERVENSYSTEM VERLAUFENDEN
ERREGUNGEN IST DAS SPIEL DER CHROMATOPHOREN.

```
(((EIN DEUTLICH SICHTBARES ZEICHEN FÜR DIE IM NERVENSYSTEM VERLAUFENDEN
ERREGUNGEN IST DAS SPIEL DER CHROMATOPHOREN )
(CLAUSE MAJOR TOPLEVEL DECLARATIVE REGULAR-ORDER)

((EIN DEUTLICH SICHTBARES ZEICHEN FÜR DIE IM NERVENSYSTEM VERLAUFENDEN
ERREGUNGEN)
(NG NOM FULL NOUN DET INDEF NEUT SING P3RD COUNT)

(EIN (DET INDEF SING NEUT NOM))

((DEUTLICH SICHTBARES) (ADJG ATTR NEUT SING MIXED NONEX)

(DEUTLICH (ADJ RELMOD UNDECL COMP SUP))

(SICHTBARES (ADJ ATTR DECL MIXED NEUT NOM SING COMP SUP)))

(ZEICHEN (NOUN STRONG NEUT NOM SING P3RD GEN-S PLUR0 COUNT))

((FÜR DIE IM NERVENSYSTEM VERLAUFENDEN ERREGUNGEN)
(PREPG SIMPLE)

(FÜR (PREP ACC PRE))

((DIE IM NERVENSYSTEM VERLAUFENDEN ERREGUNGEN)
(NG ACC SIMPLE DET DEF NOUN PLUR MASC P3RD COUNT)

(DIE (DET DEF ACC FEM PLUR))

((IM NERVENSYSTEM VERLAUFENDEN)
(CLAUSE RSQ PRESP PRENOM NONEX SUBORDINATE FEM PLUR ACC WEAK)

((IM NERVENSYSTEM)
(PREPG NO-RSQ ADVERBIAL)

(IM (PREP MIXED PRE))

((IM NERVENSYSTEM)
(NG DAT NO-RSQ DET DEF NOUN NEUT SING P3RD COUNT)

(IM (DET DEF NEUT DAT SING))

(NERVENSYSTEM
(NOUN STRONG NEUT DAT SING P3RD GEN-S PLUR-E COUNT))))

(VERLAUFENDEN
(PART PRESP DECL WEAK FEM ACC PLUR ATTR IRR UML SEIN NO-GE
 MVB PLAIN P AS-VERB)))

(ERREGUNGEN (NOUN STRONG FEM ACC PLUR P3RD PLUR-EN COUNT)))))

(IST (VERB IRR MVB PLAIN SEIN NO-END N PRES INDIC P3RD SING))

((DAS SPIEL DER CHROMATOPHOREN)
(NG NOM FULL DET DEF SING NEUT NOUN P3RD COUNT)

(DAS (DET DEF NOM NEUT SING))

(SPIEL (NOUN STRONG NEUT NOM SING P3RD GEN-S GEN-ES PLUR-E COUNT))

((DER CHROMATOPHOREN)
(NG GEN SIMPLE DET DEF NOUN PLUR MASC P3RD COUNT)

(DER (DET DEF GEN MASC PLUR))

(CHROMATOPHOREN (NOUN PLUR-EN PLUR P3RD GEN GEN-S MASC COUNT))))))
```

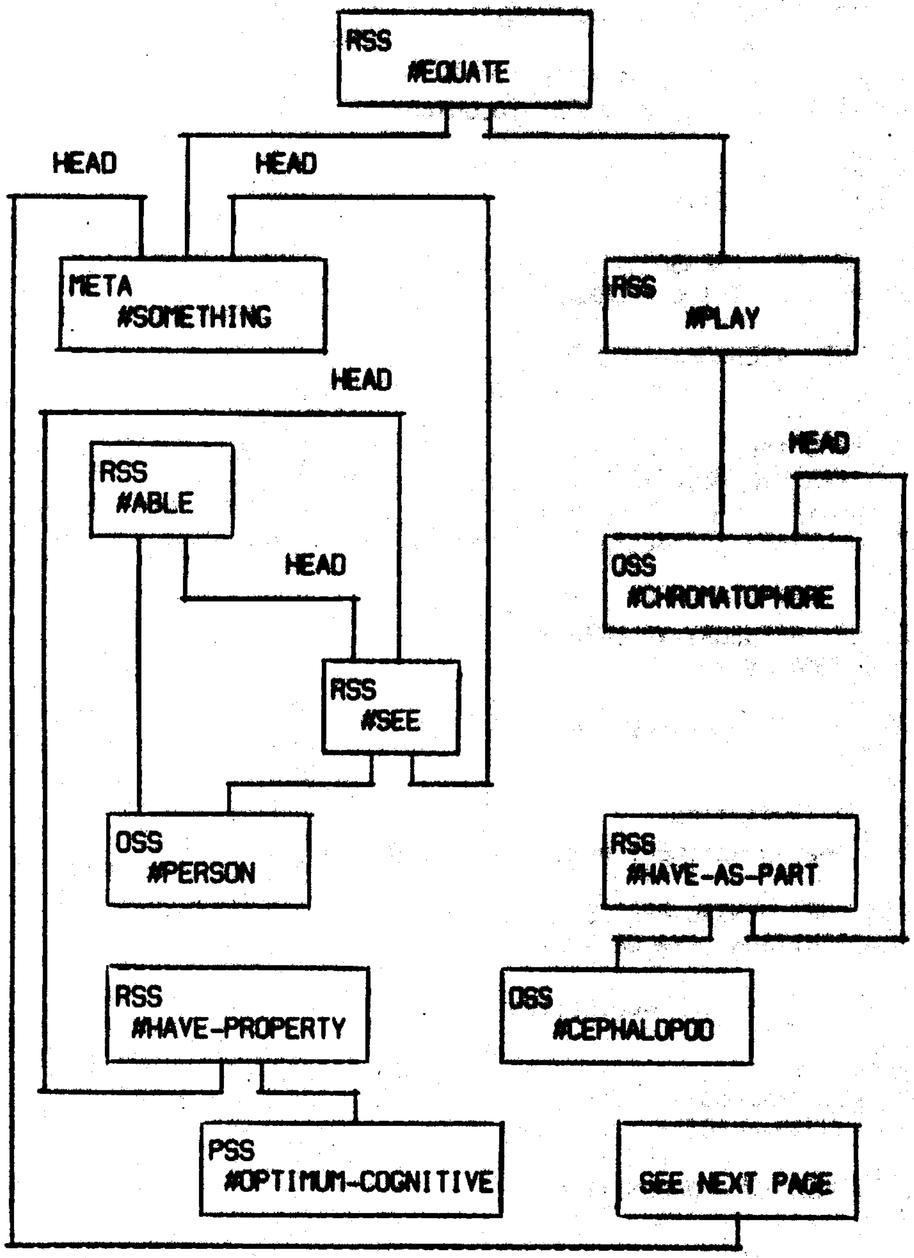
APPENDIX C. A SECTION OF THE CONCEPT MARKER TREE

EACH CONCEPT MARKER IS LINKED TO ITS PARENT BY THE UP PROPERTY AND TO ITS DAUGHTERS, IF ANY, BY DOWN. THE ORDERING OF RESTRICTION LISTS CORRESPONDS TO THE ORDER OF THE SEMANTIC ARGUMENTS OF THE RELATION.

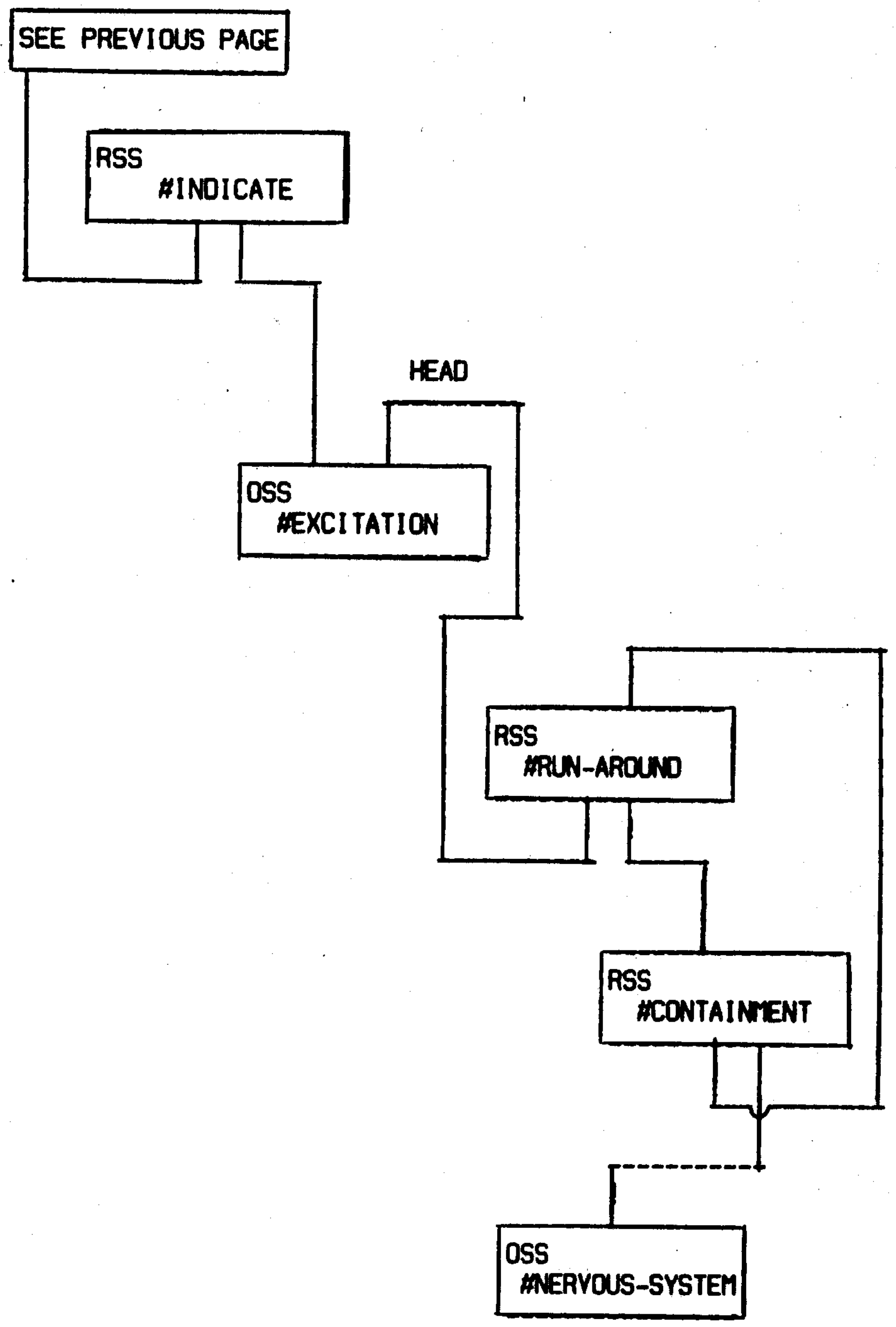
```
(DEFS #MENTAL-PROCESS  UP  #RELATION
 DOWN  (#PERCEPTION #REACTION #COGNITION))

(DEFS #PERCEPTION  UP  #MENTAL-PROCESS
 DOWN  (#SENSORY-INVOLUNTARY  #SENSORY-VOLUNTARY))

(DEFS #SENSORY-INVOLUNTARY  UP  #PERCEPTION
 DOWN (#DISTINGUISH #PERCEIVE))

(DEFS #DISTINGUISH  UP  #SENSORY-INVOLUNTARY
 RESTRICTIONS: (#LIVING-THING #CONCRETE #CONCRETE))

(DEFS #PERCEIVE  UP  #SENSORY-INVOLUNTARY
 DOWN  (#SEE))

(DEFS #SEE UP  #PERCEIVE
 RESTRICTIONS:  (#ANIMAL #CONCRETE))

(DEFS #SENSORY-VOLUNTARY (#PERCEPTION)
 DOWN (#OBSERVE))

(DEFS #OBSERVE  UP  #SENSORY-VOLUNTARY
 RESTRICTIONS: (#HUMAN #OBJECT))

(DEFS #REACTION  UP  #MENTAL-PROCESS
 DOWN  (#WISH-FOR))

(DEFS #WISH-FOR UP  #REACTION
  RESTRICTIONS: (#HUMAN (EITHER: #RELATION #OBJECT)))

(DEFS #COGNITION  UP  MENTAL-PROCESS
 DOWN (#NEUTRAL-COGNITION  #VALUE-ASSIGNED))

(DEFS #NEUTRAL-COGNITION  UP  #COGNITION
 DOWN  (#KNOW))

(DEFS #KNOW UP  #NEUTRAL-COGNITION
 RESTRICTIONS: (#HUMAN #FACT))

(DEFS #VALUE-ASSIGNED  UP  #COGNITION
 DOWN (#ASSUME))
```

APPENDIX D. SAMPLE SEMANTIC REPRESENTATION

A sample representation of the sentence, "Ein deutlich sichtbares Zeichen für die im Nervensystem verlaufenden Erregungen ist das Spiel der Chromatophoren der Cephalopoden." Semantic structures produced for tenses and those produced by determiners have been omitted for clarity.



APPENDIX D. (COND.)



BIBLIOGRAPHIC DATA SHEET	1. Report No. MAC TR-142	2.	3. Recipient's Accession No.
4. Title and Subtitle Some Problems in German to English Machine Translation			5. Report Date: Issued December 1974
			6.
7. Author(s) Gretchen Purkhiser Brown			8. Performing Organization Rept. No. MAC TR-142
9. Performing Organization Name and Address PROJECT MAC; MASSACHUSETTS INSTITUTE OF TECHNOLOGY: 545 Technology Square, Cambridge, Massachusetts 02139			10. Project/Task/Work Unit No.
			11. Contract/Grant No. N00041-70-A-0362-0006
12. Sponsoring Organization Name and Address Office of Naval Research Department of the Navy Information Systems Program Arlington, Va 22217			13. Type of Report & Period Covered: Interim Scientific Report
			14.
15. Supplementary Notes			

16. Abstracts

This paper discusses some problems in the machine translation of natural language, in particular, for translation of German into English. An implementation of some parts of the translating process has been built. The system consists of a German interpretive grammar, to take in German text and output a set of semantic representations, and a generator, to produce English sentences from single semantic representations. Special attention is paid to questions of semantic representation in a multi-language setting and to stylistic issues in English generation.

17. Key Words and Document Analysis. 17a. Descriptors

Natural language
Machine translation
Linguistics
German grammar
English grammar
Semantic representation
Computational linguistics
Parsing
Generation

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement Approved for Public Release; Distribution Unlimited	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 190
	20. Security Class (This Page UNCLASSIFIED	22. Price