

TABLE OF CONTENTS

CHAPTER ONE

INTRODUCTION	1
--------------	---

CHAPTER TWO

THE AI-APPROACH TO LANGUAGE: FRAMES AND CASES	6
---	---

A. The AI Paradigm.....	6
-------------------------	---

B. Case Systems.....	13
----------------------	----

C. Evaluation.....	30
--------------------	----

CHAPTER THREE

AN ALTERNATIVE APPROACH	35
-------------------------	----

CHAPTER FOUR

THE SYNTACTIC COMPONENT	47
-------------------------	----

A. Control Issues.....	47
------------------------	----

B. The Parser and the Recognizer.....	49
---------------------------------------	----

C. Syntactic Utility Functions.....	62
-------------------------------------	----

CHAPTER FIVE

GRAMMATICAL ISSUES	64
--------------------	----

A. Structural Limitations of Context Free Rules.....	64
--	----

B. Grammatical Rules.....	70
---------------------------	----

C. Strengths and Weaknesses of the Grammar.....	76
---	----

CHAPTER SIX

THE SEMANTIC COMPONENT	95
A. Surface Semantics.....	95
B. Word Sense Disambiguation.....	100
C. Semantic Networks and Deductive Retrieval.....	105

CHAPTER SEVEN

FUTURE DIRECTIONS AND CONCLUSION	117
----------------------------------	-----

REFERENCES	122
------------	-----

CHAPTER ONE

INTRODUCTION

This paper describes a natural language processing system that is designed to facilitate the interactive construction of natural language "front ends" and to support experiments in computational linguistics. Since the system builds data structures that are to be executed as programs, LISP (UT-LISP) has been chosen as the implementation language.

The system provides facilities for performing the following tasks:

natural language parsing according to arbitrary context-free and transformational grammars;

disambiguation of word senses using a "pattern-directed inference system";

construction of a semantic network data base from English sentences;

deductive information retrieval to answer simple English questions.

A small English to French translator has been implemented to illustrate the use of the system. (The generated French is, no better than my own, and thus, occasionally, quite poor.)

Chapter One contains introductory material and Chapter Two describes related work in Artificial Intelligence. Chapter Three proposes and argues for a design strategy that differs from the major approaches in both theoretical and computational linguistics. Chapter Four describes

the overall structure of the proposed system and the parsing component. Chapter Five contains a description of the grammar used for the translator and illustrated the syntactic coverage achieved by this grammar. Chapter Six discusses the semantic component, the word sense disambiguation and the deductive retrieval routines. Chapter Seven outlines future work and concludes the paper.

To enhance the usefulness of the proposed system as a tool for building natural language front ends, it is designed so as to allow grammatical and semantic rules to be written with minimal concern for programming matters. The grammar formalism consists of primitive operations for constructing linguistic structures and is almost entirely free of the details normally associated with programming. In contrast to other grammar-specifying programming languages, such as Winograd's PROGRAMMAR (Winograd 72) or Woods' ATN (Woods 72), the user does not even have to worry about specifying the flow of control: the syntactic component of the system takes care of control issues.

This design feature allows the user to concentrate entirely on linguistic problems. For example, if the rule: NP → DET N (NP = noun phrase, DET = determiner, N = noun) is to be used in translating from English to French, the user must decide what information is needed to achieve the agreement in gender (M = male, F = female) and universality (SING = singular, PLUR = plural) between the noun and the article. Since the noun controls the article, the noun must be evaluated first. This is accomplished by initializing two variables, UNIV and GEND, to the default values, say, SING and M. Then the right branch (RB) of the tree whose root is NP is evaluated (EV) before the left branch (LB), and the results are combined.

The evaluation of the noun results in setting UNIV and GEND. A small function attached to "the" uses these values to select the correct French article.

To have these things done in the right order, it suffices to associate a function like

```
(( INITIAL UNIV "SING GEND "M)
   (INITIAL ENVIRON (EV RB))
   (COMBINE (EV LB) ENVIRON))
```

with the rule: NP \rightarrow DET N. Moreover, once the subtree rooted at NP has been traversed, earlier default assignments to the variables, if any, will be restored.

Similar primitive operations are available to write rules for recognizing and removing intrasentential ambiguities. Instead of or in addition to such primitive operations, arbitrary LISP programs can be associated with words and context-free rules.

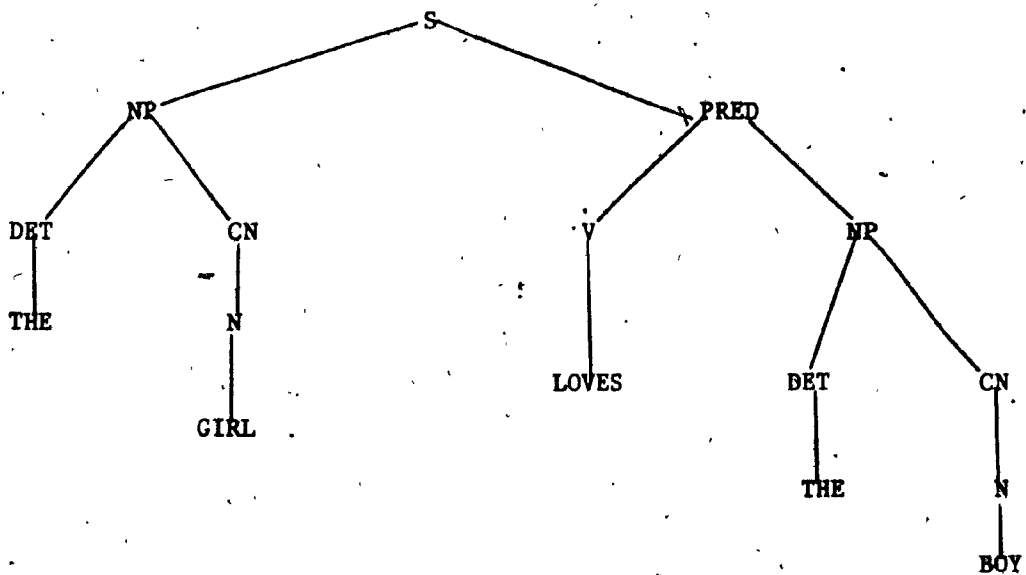
The syntactic component contains a general context-free parser whose recognizer is similar to Earley's algorithm (Earley 70). The parser differs from Earley's, however, in that it allows partially constructed phrase structure trees to be modified by the results of executing two kinds of functions: context-sensitive tests which inspect the immediate context of syntactic constituents, and functions which enable semantic information to affect the parse path taken next.

The syntactic component combines context-free rules and transformational rules as well as semantic rules in the following manner. Each grammar rule written by the user consists of three parts. The first part is a context-free rule in the usual replacement rule form. The second part is a program that acts as a transformational or a semantic rule. For example, such a rule might copy

relevant features from a controlling noun phrase onto a verb to achieve agreement. The third part of a grammar rule is a program that acts as a limited context-sensitive test.

Controlled by context-free rules, context-sensitive tests and, occasionally, semantic information, the parser constructs a phrase structure tree whose interior nodes are occupied by programs corresponding to transformational and semantic rules, and whose leaf nodes are occupied by the words in the surface string. The processing of a sentence amounts to executing the transformational and semantic rules in the order determined by a postorder traversal of the context-free parse tree.

For example, the sentence "The girl loves the boy." might be analyzed as having the following phrase structure (S = sentence, CN = common noun, V = verb, PRED = predicate):



Each interior node in this tree is represented by a LISP node of four elements. The first element of a node points to the current position in the input string,

CHAPTER TWO

THE AI-APPROACH TO LANGUAGE: FRAMES AND CASES

The following paragraphs are not meant to give a survey of the major research programmes in AI nor to describe the work of individual theorists in detail. There are several good general surveys and collections of expository papers available. See (Boden 77); (Charniak & Wilks 76), (Schank & Colby 73), (Wilks 72), (Winograd 72). Instead I shall try to bring out the flavor of linguistic work in AI and to identify several widely shared strands of thought which, together, constitute part of what is sometimes referred to as a "paradigm shift" in AI. The replaced paradigm is, of course, Chomskyan linguistics.

A) The AI Paradigm:

After an early setback of machine translation projects in the late Fifties and early Sixties most computational linguists abandoned belief in the feasibility of predominantly syntactic language processing. The break of the computational linguists with the theoretical linguists' tradition has become almost complete: as far as I know there have been no attempts to implement theories of generative or interpretive semantics, and there are only two parsers based directly on transformational grammar. ((Plath 76), (Walker 67)) The linguistic work that the AI community did exploit does not lie in the mainstream of linguistics. (For instance, Winograd's SHRDLU (Winograd 72) is loosely based on Halliday's systemic grammar (Halliday 70), Schank's conceptual dependency is an outgrowth of Hay's dependency theory (Hays 64), and the semantics of Simmons and his coworkers is based on the case grammars of Fillmore and Celce. (Fillmore 68), (Celce 72). By the way, Fillmore seems to be the only major linguist who acknowledges the stimulating effect of the AI approach on linguistics.

(Fillmore 77, p.58).

The theories propounded by computational linguists have very little in common as far as their basic concepts, terminology and notation, background assumptions, criteria of adequacy and even their purported problem domains are concerned. What all the approaches that fall in the AI paradigm do have in common can be characterized best as a negative reaction to several central aspects of Chomskyan linguistics.

Since AI theories are meant to be executed on computers they have to be what Chomsky calls performance models. Chomsky has often insisted on the importance of the distinction between competence (the grammatical knowledge of an ideal speaker-hearer) and performance (the way in which competence is put to use). This distinction, rigidly maintained, coupled with a belief that linguistics proper should address itself only to matters of competence, has led Chomsky and his followers to ignore some crucially important features of intelligent language use. (For a similar criticism, cp. (Oppacher 81)).

For example, if one wishes to construct a system capable of understanding discourse - where the presence of understanding would be evidenced by an ability to answer questions, to make sense of ambiguous or ill-formed sentences etc. - then a transformational grammar which merely generates or enumerates structural descriptions of well-formed sentences and rejects ill-formed ones is not very helpful. An understanding system ought to be capable of interpreting ill-formed sentences and not just relegate them to one of two sets. In this context it should be noted that Chomsky's theory of degrees of grammaticality constitutes, with the possible exception of his semantics, the least developed part of his theory of language.

Similar remarks apply to transformationalist examples of well-formed but semantically anomalous, i.e. meaningless or uninterpretable sentences. (Consider, e.g., Chomsky's "Colorless green ideas sleep furiously". It is quite easy to make some sense out of this prime example of an "uninterpretable" sentence. ("Boring, half-baked or vaguely formulated ideas reside in the subconscious but have a tendency to rise to awareness unbidden".)) It seems to me that Quine's arguments against the notion of analyticity (Quine 60, chpt. 2) apply with equal strength to the interpretative semanticist's notion of conceptually incoherent sentences: the inability to find interpretations for "semantically anomalous" sentences or counter examples to analytic sentences is due to an unjustifiably strict distinction between matters of fact and matters of meaning and to a refusal to consider larger units of meaning than individual sentences.

The AI approach amounts, thus, to a denial of some of the most basic tenets of Chomskyan semantics: that there is a clear-cut distinction between purely linguistic knowledge and knowledge about the world, that the appropriate unit of linguistics is the individual sentence, and, above all, that only matters of competence are relevant from the point of view of linguistic theorizing.

Another important ingredient of the AI view is its emphasis on semantics. After the rather dismal failure of the early syntactic approaches to machine translation most workers came to believe that improved performance would not result from ever more sophisticated and powerful parsers but from stronger semantic components. The increased interest in semantics led AI workers to tackle problems that the linguists had hardly considered at all. Foremost among these is the problem of meaning representation, i.e.

of devising a representational medium that would be rich enough to encode meanings in such a way as to facilitate the resolution of word sense and referential ambiguities. A related problem concerns the role of inference in understanding.

There are many specifically linguistic tasks such as the identification of referents, the choice of an appropriate structural description for structurally ambiguous sentences, the choice of a particular sense for a semantically ambiguous word etc., which require inference making. Consider the following examples:

- 1) I put the heavy box on the table and it broke.
- 2) I put the light bulb on the table and it broke.
- 3) She pushed the chair.
- 4) Robot, bring me coffee with cream and bread.
- 5) Robot, bring me coffee with cream and sugar.

In the absence of information to the contrary, most people would take the pronoun "it" in 1) to refer to the table and in 2) to the bulb. The identification of the pronoun's referent depends on inferences based on extra-linguistic knowledge of rough generalizations (e.g. that tables do not normally break unless very heavy things are put on them.) 2), it will be noted, presents a further problem. Again, in the absence of information to the contrary, most people would take 3) to imply that the chair had moved to a new location. In 4) one would expect the robot to apply extra-linguistic knowledge to produce a snack whose structure is represented by 6) and not by 7). In the case of 5) the structure should be that of 7), however.

6) (and(with coffee cream) bread)

7) (with coffee (and cream bread))

As can readily be seen from these examples, the intelligent use of language requires frequent inference making. Indeed, most sentences do not state all the information they are meant to convey explicitly: their meaning has to be inferred. The ubiquitous need for inference creates a host of further questions: should inferences be made only when needed, i.e. when a question is being asked or some expression has to be disambiguated, or should they be made unprompted upon the arrival of new information? Which of the indefinitely many possible inferences should actually be made? How should the required common sense knowledge be organized and accessed? Which inference rules should be used? Which mechanism should apply the rules? For some answers to these questions, cp. (Charniak 72, 75, 76), (Charniak, Wilks 76), (Schank, Goldman, Rieger, Riesbeck 75), (Rieger 74, 75).

Let me recapitulate what I take to be some of the important ingredients of the AI paradigm:

Theories of language should be executable competence models which are predominantly semantic in orientation. They should contain strong inferential abilities which are applied to extra-linguistic knowledge in order to resolve various kinds of ambiguities and to render implicit information explicit. Finally, they should construct formal representations of meaning that are useful for question-answering, problem solving and translation.

Most recent theories which address themselves to these problems use as their basic building block one variant or other of the concept of

a frame. In a very influential paper, Minsky describes frames as follows (Minsky 75, p.212):

"A frame is a data-structure for representing a stereotyped situation, like being in a certain kind of living-room, or going to a child's birthday party. Attached to each frame are several kinds of information. Some of this information is about how to use the frame. Some is about what one can expect to happen next. Some is about what to do if these expectations are not confirmed.

We can think of a frame as a network of nodes and relations. The "top levels" of a frame are fixed, and represent things that are always true about the supposed situation. The lower levels have many terminals - "slots" that must be filled by specific instances or data. Each terminal can specify conditions its assignments must meet. (The assignments themselves are usually smaller "subframes".) ...Collections of related frames are linked together into frame systems. The effects of important actions are mirrored by transformations between the frames of a system...Different frames of a system share the same terminals; this is the critical point that makes it possible to coordinate information gathered from different viewpoints... A frame's terminals are normally already filled with "default" assignments. Thus, a frame may contain a great many details whose supposition is not specifically warranted by the situation. These have many uses in representing general information, most-likely cases, techniques for bypassing "logic", and ways to make useful generalizations."

As can be seen from this description, frames are record-like structures - in fact, they are generalizations of LISP property lists - whose fields may have a wide variety of contents; values, default values, restrictions on values, comments on the reliability or source of values, procedures expert at using certain pieces of knowledge, pointers to other frames etc. Frames can be tied together by A-KIND-OF links which point to more general frames and which make it possible for a frame to inherit information from its generic frames. (For a small sample of frame-based work on language, cp. (Bobrow & Winograd 77), (Charniak 72, 75, 76, 77), (Charniak & Wilks 76), (Grosz 77), (Goldman 74), (Hendrix 76, 75), (Schank & Abelson 77), (Schubert 76), (Schubert et al. 79), (Simmons 78, 77), (Wilks 72, 75a, 76, 77, 79), (Winograd 72).)

In order to give a better idea of the application of the frame concept to natural language processing I shall give a very rough sketch of two frame-based theories, the case systems of Fillmore and of Schank. I have chosen to describe these two theories because they are typical of much recent work. Fillmore's theory of case grammar forms the linguistic basis of several programs (e.g. (Simmons 73, 78), (Simmons & Chester 77)) and seems to be the most influential contribution to the AI-paradigm by a linguist. Schank's conceptual dependency theory or some variant of it underlies several powerful understanding programs and has been used to tackle a wide spectrum of language-related problems: inference, paraphrase, translation, building structures to represent an understanding of stories etc. (It is its very generality that makes Schank's work more important, in my opinion, than, say, Winograd's or Wilks'. Wilks' system is entirely dedicated to the resolution of semantic ambiguities, and it is

by far the most expert system for this task, but it is unclear whether or how it could be used for anything else. (Wilks 72, 79). Winograd's famous program SHRDLU (Winograd 72) exploits the constraints provided by its domain, a mini-world of toy blocks manipulated by a simulated robot hand, so heavily that it is doubtful whether it could be made to perform as impressively in more realistic domains.)

B) Case Systems:

Case grammar (CG) is based on the belief that in order to understand natural languages semantics is much more important than syntax. Accordingly, CGs do not generate syntactic deep structures which serve as inputs to interpretive or generative semantic rules. Instead, syntax is used only to provide clues about the relation between surface sentences and deep cases.

The notion of (surface) cases has been employed by grammarians at least since the time of the Stoics. In some strongly inflected languages like Latin or German, affixes of nouns provide information about gender, universality, and the syntactic relation of nouns to other parts of sentences. (Surface) cases refer to the relations, signalled by affixes, between nouns and containing sentences.

The novel aspect of CG lies in the distinction between deep and surface cases and in the insistence that deep cases occur in all languages, even in those languages which, like English, have practically no nominal inflection. (Personal pronouns provide a rare example of case-like nominal inflection in English: he, she, they constitute the nominative case; his, her, their the possessive case and him, her, them the objective

case of the third-person pronoun.)

One of the major proponents of CG, C. Fillmore (Fillmore 68, 71, 77), has argued that concepts like "subject" or "object" are appropriate only to describe surface structures in some but not necessarily all languages and that a theory of language should dispense with such concepts in favor of case structures which represent conceptual events. In the following, I shall briefly describe Fillmore's approach because many computational linguists have found its emphasis on issues of meaning representation and on conceptual rather than syntactic matters very attractive.

CG revolves around the description of events. Most human communications convey information about events. (States and objects can also be construed as special, static events.) Events are frequently described by simple sentences, as in "she chooses", or by noun phrases as in "the choosing" or "the choice".

Consider the event described by 8):

8) Joe surprised Mary with a gift.

The concept of "surprising" at issue in 8) involves the concepts of an agent (A) who does the surprising, a recipient or experiencer (RE) who is being surprised, an instrument (I) used to surprise, and possibly a time and place in which the surprising happens.

This conceptual knowledge about the verb in 1) can be represented as a set of allowable cases, i.e. a case structure and a set of selection restrictions. The case structure for "surprise" could be expressed as $\langle (A), R, (I) \rangle$, with the understanding that each case may occur at most

once, that R must occur in the surface sentence, and that A and I (as indicated by the round parentheses) may but need not occur in the surface sentence. The selection restrictions place further semantic constraints on the objects that fill the case slot. In the present example, both the agent and the recipient must be animate.

Once a case structure has been formulated for a verb, CG parses a sentence by determining whether each case required to occur in the surface structure does indeed occur in the sentence, whether all cases satisfy the imposed selection restrictions, and whether the cases extracted from the surface structure match the case structure.

Consider the following sentences:

- 9) Mary was surprised by the gift. (R,I)
- 10) Joe was surprised with Mary. (R = Joe with Mary)
- 11) Joe surprised himself. (A,R)
- 12) The gift surprised Mary. (I,R)
- 13) Mary was surprised. (R)
- 14) Mary was surprised by Joe. (R,A)
- 15)* Joe surprised.
- 16)* Joe surprised the gift.
- 17)* The gift was surprised by Mary.

The illformedness of 15) - 17) can be accounted for as follows: in 15) the obligatory recipient noun phrase is missing, in 16) and 17) the recipient fails to be animate.

The major remaining problem, especially for weakly inflected languages, is to determine the deep cases for each noun phrase from the surface structure. In English, noun affixes, prepositions, and certain syntactic features such as specific word order and definiteness of noun phrases are used as surface case indicators.

Some tentative rules for the identification of cases might look like this: if A is present in the surface sentence then it is the subject of an active sentence and the object of the preposition "by" in a passive sentence. If A is absent then the subject of an active sentence or the object of "by" or "with" in a passive sentence is I. R must be present (in the above example) and is the object of an active sentence or the subject of a passive sentence.

It is clear from the above example and the tentative rules for the identification of cases that the success of a CG depends crucially a) on the reliability with which cases can be inferred from the surface structure and b) on the appropriateness of the chosen set of deep case structures or conceptual events.

Concerning a), I would like to point out that the identification of deep cases requires a considerable amount of syntactic processing. (Consider, e.g., the above rules, which are no less precise than those proposed by case grammarians, and their application to

18) She was married by a priest.

and

19) She was married by the time her labour began.)

There is a misleading tendency among some computational linguists,

notably Schank, Rieger, and Riesbeck, to claim that in their case-based systems syntax plays practically no role at all.

Concerning b) I should like to point out that there are no theoretically adequate and general criteria for selecting deep cases. A deep case, as the above description has shown, is essentially a binary relation between a predicate (in some formal language like first-order quantification theory) and one of its argument places. The problem of constructing a case structure boils down to the question how many and which arguments a predicate in some formal language should receive. Sometimes linguists appeal to a criterion according to which cases are to be assigned in such a way that different senses of a word can be distinguished. However, it would be quite easy to show (although I do not intend to do so here) that no general criterion for individuating word senses exists (Quin 60). Indeed, it seems the question whether the occurrences of "run" in

- 20) Joe runs.
- 21) Joe runs the machine.
- 22) The machine runs.
- 23) The river runs.

have the same or different meanings can be answered only relative to one or another presupposed case-like system.

Suppose we wish to write a case structure for 'breaking' as in

- 24) Joe broke the window with a hammer.

We would introduce a primitive predicate BREAKING to denote events of a certain kind. (For a good account of the problems involved in specifying

and individuating events, cp. (Davidson 67). Then, quantifying over events, we could write

25) $\forall x (\text{BREAKING}(x) \wedge \text{AGENT}(x, \text{joe}) \wedge \text{OBJECT}(x, \text{window}) \wedge$
 $\text{INSTRUMENT}(x, \text{hammer}) \wedge \text{TIME}(x, \text{past}) \dots)$

The problem here is that there is no general way of deciding how to fill in the ellipsis "...". Should we construe the conceptual event of BREAKING as including a location, the manner in which the agent acted (e.g. deliberately) etc? It seems that a general language understanding program should be capable to select from the indefinitely large number of predicates which are especially important for an event in the light of the current context and purpose of discourse. (cp. (Bruce 75, p.26).) Such a program would have to have a function from concepts (like BREAKING), concept-argument relations (like AGENT (x,y)) and contexts into degrees of relevance or importance such that a deep case for a concept would be any concept-argument relation whose degree of relevance exceeded some pragmatically determined threshold. It goes almost without saying that no existing program employs such an importance measure.

The theoretical difficulties involved in selecting deep cases account for the proliferation of different case systems. To give an idea of the diversity of approaches I shall now briefly sketch Fillmore's and Schank's CG.

Fillmore (Fillmore 68, 71, 77):

The deep structure of a sentence consists of a modality and a proposition:

(In this section, a "+" symbolizes concatenation).

$S \rightarrow M + P.$

The modality component M consists of negation, tense, aspect and mood. The propositional component P is a tenseless structure consisting of a main verb V and one or more case identifiers C each of which occurs exactly once. The case identifiers can be used to derive noun phrases or subsentences.

$$P \rightarrow V + C_1 + C_2 + \dots + C_n.$$

The link between universal cases and their surface indicators in specific languages is accomplished by transformational rules which map deep cases into their surface representations by means of a Kasus element K. K can be used to derive prepositions or particular word orders. A scheme for these rules looks like this:

$$C_i \rightarrow K + NP.$$

Fillmore has presented different case systems in different publications. The following case system is from (Fillmore 71):

- Agent: The instigator of an event.
- Counter-Agent: The force against which the action is carried out.
- Object: The thing that moves or changes or whose existence is considered.
- Result: The thing that comes into existence as a result of the action.
- Instrument: The immediate physical cause of an event.
- Source: The place from which something moves.
- Goal: The place to which something moves.
- Experiencer: The entity which receives or experiences or undergoes the effect of an action.

To these basic cases are added cases required for the specification of verbs of motion, location and duration. Case structures for verbs are constructed (as described above) by specifying which case relations hold between a verb and its associated noun phrases.

Schank (Schank, 72a, 73b, 75, 77):

Schank's version of CG has been partly implemented by (Goldman 74), (Rieger 74) and (Riesbeck 74). Schank has developed an entirely language-free and unambiguous medium for representing the meaning of sentences, the conceptual dependency representation. (CD). Unlike Fillmore's cases, Schank's "conceptualizations" are non-linguistic in the sense that none of their components have to be mentioned in a sentence. Thus, since actions have the conceptual case OBJECT, an object is assumed to be present in the meaning representation of a sentence even if no object is explicitly mentioned. An unfilled object-slot might give rise to an attempt to infer a slot-filler from the context. Accordingly, Schank's arguments for conceptual cases appeal to pragmatic considerations about what kinds of things people typically communicate when describing simple actions. (At times, Schank claims much more for his chosen set of ACTS than that they are sufficient to represent the meanings of a broad range of non-technical sentences. In (Schank 77, p.585), e.g, he states that "...ATRANS is the one ACT ... that is not necessarily universal." It is quite unclear, to say the least, what the implied claim that all the other primitive acts are necessarily universal amounts to. Does he wish to maintain that his set of primitives, or rather the most recently published version of it, enjoys some sort of empirically discoverable objective existence, that the acts denote species-wide brain patterns? Or does he

wish to maintain that no semantic analysis could be both adequate and not based on these acts? I do not want to go into this matter here but I do want to point out that he has not produced any arguments to support such exceedingly strong if not incoherent claims.)

Talk about actions presupposes a conceptual structure consisting of actions and their conceptual cases or role fillers:

ACTORS perform ACTIONS.

ACTIONS have OBJECTS

ACTIONS have INSTRUMENTS

ACTIONS may have RECIPIENTS

ACTIONS may have DIRECTIONS

One type of conceptualization characterizes a verb by providing all of the following: a primitive act, its actor, object, direction or recipient, and instrument.

CD, like other CGs, is verb centered and, unlike other CGs, describes all verbs in terms of a small set of primitive acts. (The primitive acts are frequently changed without warning, and their number is sometimes 10, sometime more. However, the basic idea behind them is always the same.) The primitive acts are meant to represent concepts which all speakers of any natural language learn early in their linguistic careers and out of which all the more complex action-oriented concepts are fashioned. One such set of 12 primitive acts is the following (Schank 77, pp.581 ff).

Physical ACTS:

PROPEL: apply a force to.

MOVE: move a body part.

INGEST: take something inside you.

EXPEL: take something from inside you and force it out.

GRASP: grasp, i.e., what a hand can do.

(Schank (ibid, p.582) claims that these are the only ACTS that can be performed on a physical object.)

GLOBAL ACTS:

PTRANS: change in physical location.

ATRANS: change in ownership.

(Schank (ibid, p.585) claims that ATRANS is the only ACT which is not necessarily universal. It would be absent in a culture lacking a concept of possession.)

Instrumental ACTS:

ATTEND: direct a sense organ to a location.

SPEAK: Schank (ibid, p.586) gives this rather meaningless explanation: "SPEAK is the ACT which actually produces sounds..."

(In fact, his work is replete with such conceptual confusions.. Not only does he allow abstract concepts to produce actual sounds, but he often confounds conceptualizations with the events conceptualized.)

Mental ACTS:

MTRANS: to move information to and from the conscious mind.

MBUILD: to combine thoughts.

In addition to the primitive acts, CD contains unanalysed physical objects, called "picture producers" (PP) and "picture aiders" (PA) that describe PPs. PAs are states like FEAR, HUNGER etc. or relations like PART-OF or PHYSICAL-CONTACT etc. The acts, PPs and PAs are linked into CD-graphs by means of a small number of distinguishable, i.e., labelled edges. Some of these edges and restrictions on their use are shown below (these diagrams could be considered as syntactic rules at the conceptual level):

PP \longleftrightarrow ACT : PP can be the agent of a primitive ACT.

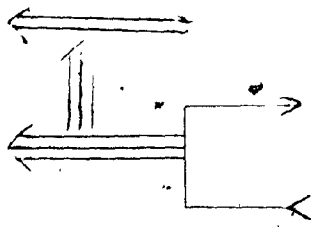
PP \longleftrightarrow PA : a PP can be modified by a PA. (an attribute is predicated of an object).

PP \longleftrightarrow PA : a PP can change from one stage or relation to another.

ACT \xleftarrow{O} PP : an ACT can have a PP as object.

ACT $\xleftarrow{D, R}$ PP : an ACT can have a direction or a recipient.

ACT \xleftarrow{I} PP : an ACT can have an entire conceptualization as its instrument.



: a conceptualization can cause a state or relation to change. (Causal relations hold only between two-way dependencies such as \leftrightarrow , \leftarrow , \rightarrow i.e., only events or states can cause events or states.)

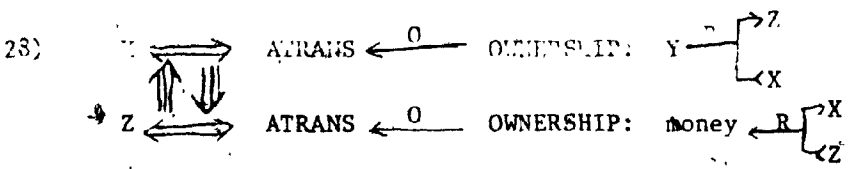
Unlike other forms of CG which are also centered on an actor-action-object framework with cases for the actions (cp. the linguistic system of (Celce 72) and its partial implementation as a program by Simmons and his co-workers (Simmons 73, 78)), CD insists that all actions be expressed in terms of the primitive acts and a few relationships (\leftarrow , \rightarrow , \leftrightarrow , \uparrow , \downarrow). This restriction to a small set of basic ingredient acts and a small number of modes of combining them is similar to the adoption of a meagre set of primitive concepts and axioms in the axiomatic approach to theory construction and enjoys similar benefits of systematization.

The austere, graph-theoretical framework of CD is thus able to capture regularities that lie at a level that is deeper than the one expressed in semantically interpreted Chomskyan deep structures. Consider for example,

- 26) X sells Y to Z.
- 27) Z buys Y from X.

Although 26) and 27) are cognitively synonymous, an interpretive transformational semantics cannot adequately express and explain this fact.

In CD however, the deep similarity of 26) and 27) can be readily exhibited by the representing graph.



This graph is interpreted as follows: selling is at issue when a transfer of an object causes a transfer of money, and buying when a transfer of money causes a transfer of an object. The abstract act ATRANS in 28) has as its object an abstract relationship. The physical acts involved in bringing about the transaction, such as handing over a cash amount or signing a check, constitute the omitted instrument of ATRANS.

The similarity between one sense of "give" and "take" as in 29) and 30) can be exhibited as in 31) and 32).

29) X gave Y to Z.

30) X took Y from Z.

31) X \xrightarrow{P} ATRANS \xleftarrow{O} POSSESSION: Y \xleftarrow{R} $\left\{ \begin{array}{l} Z \\ X \end{array} \right.$

32) X \xleftarrow{P} ATRANS \xleftarrow{O} POSSESSION: Y \xleftarrow{R} $\left\{ \begin{array}{l} X \\ Z \end{array} \right.$

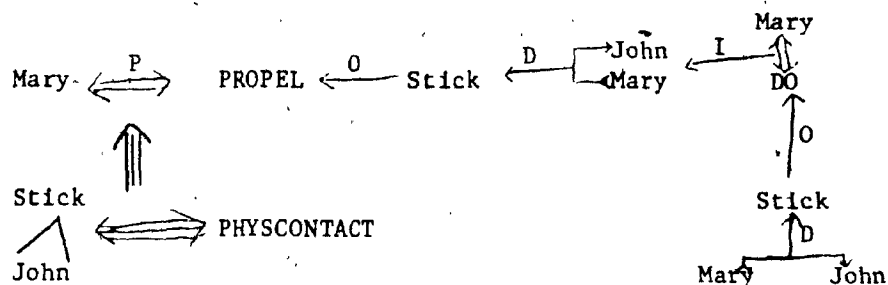
31) and 32) show that givings and takings have the same structure and differ only in the slot-filling agents: if the actor is the source of the transfer he gives something to somebody and if he is the destination then he takes something from somebody. (The "p" stands for "past".)

The requirement to build complex actions out of primitive acts can also reveal hidden ambiguities. In

33) Mary hit John with a stick.

we are only told that Mary brought a stick into sharp physical contact with John but not whether she did so while holding the stick or by throwing it at John. Our ignorance about how she did the hitting is expressed by the presence of an unspecified act DO. A DO in a graph

might invoke a procedure that tries to infer missing information from the context.

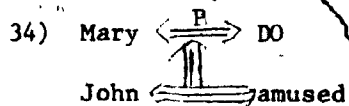


Again, in a linguistically rather than conceptually oriented CG

33) Mary amused John:

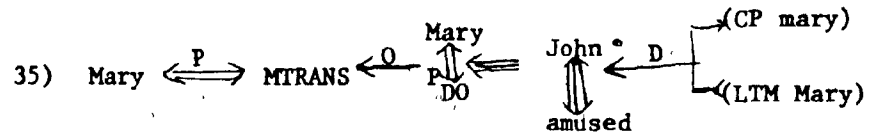
might be given the shallow analysis: (actor : Mary;
 action : amuse;
 object : John;
 tense : past).

From the point of view of semantic theory it seems inappropriate to admit amuse as a basic, unanalyzed action. After all, there is no one thing that anyone could do to amuse everybody. John could have been amused by how she walked or talked or dressed. It seems, then, that amuse refers not to a specific action but to the result some action has on somebody. This is brought out clearly in 34).



As a final example, consider the following used of a mental ACT.

(CP refers to the conscious processor, or to what psychologists might call short-term memory and LTM refers to long-term memory, i.e., the store of beliefs about the world.)



36) might be a representation of

37) Mary remembered that she had amused John.

Schank's system is not good at handling ambiguities of nouns, adjectives or adverbs because it treats them as lacking internal structure. But CG, and Schank's version of it in particular, has some important advantages over interpretive transformational semantics. For example, by defining all actions in terms of a small set of ACTs the problem of writing inference rules becomes tractable. Instead of formulating similar rules for each of a group of similar words, rules are written only for ACTs and are thus accessible to each word in whose definition these ACTs occur. (Inference rules which facilitate understanding express, among other things, very general, often trivial facts about human desires and motives, about background assumptions made by participants of conversations, and about conceptual regularities. "For example, one rule is that if you MTRANS something to your LTM, then it is present there (i.e. you know it). This is true whether the verb of MTRANSing was see, hear, inform, remember or whatever. The inference comes from the ACT rather than the verb." (Schank 74, p.10).)

It should be obvious from the above sketch of CD that it constitutes a powerful approach to semantics which can be readily combined with inferential

abilities. (No provision is made in the Chomskyan model of semantics to include inferential abilities despite their crucial importance for understanding.) However, while CD is among the most powerful and flexible representational media available, it suffers from several serious shortcomings. (Its inability to handle ambiguous nouns etc has already been mentioned.)

The insistence on always decomposing concepts into their lowest-level ingredients leads to very large data structures in which the relationships necessary for inferencing are expressed directly by labelled edges. Thus, the process of drawing inferences always involves pattern matching of very large structures. While there are some linguistic tasks such as paraphrasing sentences which do require an analysis of the fine structure of concepts, there are many other tasks, notably those involved in question-answering and problem-solving, which require the determination and use of relationships between high-level concepts. The latter tasks are actually impeded and slowed down by the irrelevant details of CD-graphs. In systems intended for natural language based problem-solving and question-answering, it has thus been found necessary to use stronger inference rules together with a predicate-logic canonical notation whose component predicates correspond directly to high-level concepts. (See (Schubert 76), (Rieger 79), (Schubert et al, 79).)

But whereas CD is too fine-grained for many linguistic tasks, there are other tasks for which it is not fine-grained enough. For example, walking is defined by Schank as PTRANSing oneself by MOVEing one's feet in a certain direction. This definition, as many other CD conceptualizations, does not even come close to representing what people know about

walking. (For a page-length semantic net for the concept of walking, cp. (Cercone & Schubert 75).) As it stands, the definition does not allow a distinction between walking, running, hopping, skiing etc.

All frame or CG based approaches construe the understanding of the meaning of a sentence as the substitution of particular values such as lexical items or their conceptual counterparts into the slots of a case structure or frame. Frames are usually associated with verbs and are used to impose a fixed structure on the description of events. (Remember the previous example of a commercial transaction. Such an event typically involves two people, the buyer and the seller. Both agents perform two actions, i.e. the buyer takes the goods and gives away the money and the seller takes the money and runs.)

The major advantage of a case or frame based approach lies in the fact that all the relevant information about a certain type of event is contained in the frame definition and does not have to be associated with each vocabulary item involved. Since frames may contain default values, the process of interpretation does not collapse or engage in costly deductive inferences if one or more cases that are not mentioned in an input sentence have to be identified. On the other hand, the very non-redundancy of CD, which is governed by the requirement that intuitively synonymous sentences be mapped into the same representation, will lead to very inefficient processing in those applications which do not presuppose the full range of computer understanding. As I shall try to show, translations of reasonably large subsets of natural languages are precisely tasks which can be accomplished without the strong semantic capabilities of CD-based programs.

C) Evaluation:

It goes without saying that there is no AI-approach that would be clearly preferable to all others. As is often the case in new fields of research, there are many different approaches with idiosyncratic conceptual bases and methods. There is, thus, no widely accepted body of theory upon which to build and little transfer of results between the major schools of thought. However, most theorists agree that semantics is extremely important, that factual knowledge is needed for linguistic tasks and that understanding requires deductive and non-deductive inference. These are profound insights which will have to be taken into account by any attempt to construct general natural language processing systems.

AI-work with its insistence on performance models has also shown more clearly than theoretical linguistics that natural language processing is exceedingly difficult and fraught with nearly intractable problems.

a) Some of these problems concern the representation and use of world knowledge in large data bases, and the inference mechanisms necessary to retrieve assertions from data bases. (The best understood inference mechanism, i.e. the resolution method, gets bogged down by combinatorial explosion in all but toy applications.)

b) Other problems arise in connection with coherent pieces of text, e.g. the problem of determining how currently mentioned items correspond to items mentioned earlier or how the topics of individual sentences contribute to the building of a data structure that represents a paragraph or an entire story. (For some tentative and partial answers, cp. (Bobrow et. al. 77), (Bullwinkle 77), (Grosz 77), (Sidner 79).)

c) As a final example of a problem that appears all but unsolvable in a general setting, consider the task of disambiguation. A program capable of handling sentences like "The box was in the pen", "The pen was in the box", "I saw the Montreal skyline driving across the bridge" etc. as successfully as people would have to have a complexity comparable to that of a human mind.

The novelty and toughness of these and similar problems explain the diversity of styles and approaches in AI. A comparative evaluation of work in this area is made difficult, however, not only by the considerable variety of approaches but also by the fact that the published literature usually does not include programs. Research papers typically contain general claims about the capacities of programs, a few examples of successfully processed sentences, and, sometimes, spotty descriptions of things the programs cannot handle. This can be quite misleading. (For example, several enthusiastic reviewers of Winograd's program SHRDLU (Winograd 72) have described it as solving some of the most vexing problems in language processing. One such problem concerns pronomial disambiguation. It is clear that SHRDLU does not solve this problem at all. It uses a few simple semantic compatibility tests and recency rules which would be inapplicable in less severely restricted domains. It should be pointed out, however, that Winograd himself does not make similarly exaggerated claims. For an attempt to evaluate SHRDLU and some other programs by asking their authors, cp. (Petrick 77). For the claim that sometimes example sentences might not have been processed by an existing program but merely by an imagined one that the author feels capable of writing eventually, cp. (McDermott 77).)

This situation: incompatible theories and conceptual systems, few published programs, scant information on syntactic and semantic coverage makes comparative evaluations practically impossible. It would be possible, though useless, to make anecdotal comparisons of reported memory requirements, "average" processing times, admitted gaps in coverage, vocabulary sizes etc. The information one would really like to have, however, is next to impossible to come by. For instance, one would like to know which grammatical constructions a system can cope with, how freely the available constructions can be combined, how easily a system can be made to break down, how gracefully it degrades in case of breakdown, how domain dependent or extensible it is, how "deep" or fine-grained its semantics is, how efficient it is etc.

A rare example of an attempted evaluation and application of a well-known natural language system is given in (Wolfe 75). Wolfe reports some disappointing experiences with one of the most robust systems, LSNLIS. LSNLIS, the Lunar Sciences Natural Language Information System (Woods et. al. 72), was designed to enable lunar geologists to ask English questions about lunar rock samples from the Apollo mission. In (Woods et. al. 72 p.52) the system is reported to be quite successful in satisfying its prospective users: 78% of initial questions posed by geologists at a 1971 conference were processed correctly. (Petrick 77, p.321) notes that Woods has mentioned in several talks that a much smaller percentage of follow-up questions could be processed correctly. Wolfe used the LSNLIS system to produce questions from a programming manual. He reports that 40% of the parsed sentences were assigned incorrect structures. Unfortunately, he gives no examples of parsed sentences

or incorrect parses. He also claims that occasionally the program ran for over one hour before terminating unsuccessfully! All of this sounds quite discouraging, especially when one takes into account that LSNLIS is widely regarded as one of the most powerful systems, as one of few systems with potential serious users, and as the only system so far whose capacity to meet user needs has been assessed.

It seems safe to say, then, that no natural language system has yet proved its mettle in practice. (There are some doubtful exceptions to this claim, e.g. question-answering systems with restricted natural language front-ends like MYCIN, (Davis et al. 77), and the little-known TAUM-METEO system (Chevalier et. al. 78) whose inputs are severely restricted.) The well-known systems are toys, as far as their practical applicability is concerned. Although they have helped uncover many important principles they have not been extended to practically relevant domains. At the moment, the tremendous investment of time and effort in developing understanding programs can be justified only on theoretical grounds, in terms of the insights they have provided us into possible human conceptual mechanisms. (The effort invested in developing those programs is indeed considerable. For example, Wilks, and Simmons and his co-workers began work on their systems in the middle Sixties, Schank and his co-workers in the late Sixties and early Seventies.) Unfortunately this theoretically valuable research has not yet produced any programs which do things that might be of interest to large groups of users.

Moreover, it seems that constructing powerful and useful systems will require rapid access to and manipulation of knowledge bases that are several orders of magnitude larger than the ones presently used. The latter problem of organizing and using vast amounts of knowledge

is not a technological problem that could be solved with larger memories but a theoretical one whose solution depends on the development of powerful heuristics and inference methods.

If one does not wish to work for many man-years on a program that has only slight prospects of becoming practically relevant then one faces a choice: one might attempt to make a purely theoretical contribution to our understanding of understanding, or one might drop the goal of achieving computational text understanding and write a program with more modest aims. I have chosen the latter, more "pragmatic" option.

Accordingly, I have tried to design a system that can produce useful translation of restricted sublanguages of a natural language into corresponding sub-languages of another natural language.

CHAPTER THREE

AN ALTERNATIVE APPROACH

Several shortcomings of current computational theories have been pointed out above, in particular, their unimpressive performance in practical applications despite great efforts by many talented workers. It is tempting to suspect that these shortcomings are not entirely due to the complexities of natural language but, at least in part, to basic theoretical and methodological inadequacies in many of the presently fashionable approaches. This suspicion is strengthened if one considers the ease with which all normal humans acquire mastery of a natural language quite early in their linguistic careers.

In the following, I shall outline an alternative view by contrasting it with two widely shared and - as I shall try to show - wrongheaded theoretical assumptions. (A_1 and A_2). The alternative view amounts to a rejection of both assumptions.

A_1 : Syntax is relatively unimportant. What little syntactic processing has to be done, is to be done under the control of semantic routines.

A_2 : Natural language processing requires a single, homogeneous, "deep", language-free medium of representation.

(It should be noted that I claim that A_1 and A_2 are part of the "folklore" of the field, i.e. that many workers embrace views very similar to A_1 and A_2 , but not that everybody, or even anybody, subscribes to these assumptions exactly as stated.)

A₁ may be partly due to an overreaction to Chomsky's insistence that syntax is more important than semantics and can be pursued without appeal to semantic considerations. For example, in (Chomsky 57, p.108), Chomsky writes 'Description of meaning can profitably refer to this underlying syntactic framework, although systematic semantic considerations are apparently not helpful in determining it in the first place.' And in (Chomsky 65, p.75), in the context of a discussion of subcategorizations such as the classifying of certain nouns as abstract etc., he makes this astounding claim: "A linguist with a serious interest in semantics will presumably attempt to deepen and extend syntactic analysis to the point where it can provide the information concerning subcategorization, instead of relegating this to unanalyzed semantic intuition..." It is strange to be told that a serious interest in semantics leads one to pursue syntactic analysis and that semantic considerations are not helpful in the search for syntactic structures.

It seems clear, on the contrary, that if syntactic constituents were not chosen to correspond closely to carriers of meaning, then syntax would be little more than a description of useless structure. But, while I agree with the proponents of A₁ that syntax cannot stand on its own, I do not take the ultimate dependence of syntax on semantics as showing that the latter is sufficient by itself or that the former is unimportant. It is precisely the dependence of syntax on semantics, i.e. the fact that appropriately chosen syntactic structures embody some semantic information, that makes a strong syntax crucial for language processing.

It also seems quite clear that human understanding of text is greatly

facilitated by syntactic preprocessing, although this is denied by some proponents of A_1 . Schank, for example, (cp. Schank 75, p.12) holds that syntax plays practically no role at all in understanding. This view can be readily refuted by noticing the difficulties people have when presented with text from which some syntactic redundancies have been removed. If if if if syntax is unimportant, semantics is primary, syntax does not facilitate understanding, they should be conflated, Schank is right, only if A_1 is correct. Or is it?

To insist on a strong syntactic parse is not to claim, as does Chomsky, that people complete the syntactic parse before initiating any semantic routines, nor that syntax is somehow more important than semantics, but only that categorizations which result from exploiting syntactic redundancies make the task of semantic interpretation much easier.

The issue is partly a matter of trade-offs: if the semantics is very strong, one can get by with a relatively weak syntax, and, conversely, if the syntax is strong, then a relatively shallow semantics may suffice. Since syntax is better understood than semantics, a stronger syntactic component seems preferable.

A prominent feature of programs based on A_1 is their excessively complicated control structure. This makes such programs unintelligible to almost everybody and prevents their modular extension. Their complex structure is presumably due to the fact that the syntactic expertise is distributed throughout the system. In the interests of improving modularity, for example, to make it possible to add new semantic rules without concern for the control part, it seems indicated to keep syntax and semantics as neatly separate as

possible. Moreover, if it should turn out that A_2 is wrong and that several levels of semantic representation are necessary, then it becomes imperative to keep syntax and semantics separate. Otherwise, syntactic knowledge would have to be duplicated for each semantic level.

To summarize, I propose to replace A_1 by A_1' :

A_1' : In order to make language programs modular and capable of interfacing with several types of semantic representations, the syntactic and semantic components should be kept largely distinct while allowing for some interleaving. A strong syntactic parse is useful because syntactic structure contains much semantic information.

A_2 is a basic tenet of many theories in AI and in cognitive psychology. (For examples from psychology, cp. (Norman & Rumelhart, 75), (Anderson & Bower 73).) According to A_2 , the understanding process involves the extraction, from the surface sentence, of abstract semantic features. The latter are then assigned to slots of pre-existing frame-like patterns. Semantic theories based on A_2 are essentially ways of translating from a natural language to a "conceptual", abstract language, akin to what some philosophers call an internal "language of thought".

Such theories suffer from many philosophical ailments. This might be deemed an excusable shortcoming if they performed well in practice. But they don't. Conceptual carelessness might well be one of the reasons why the study of language is so backward. It should be noted, for example, that the

pertinent literature does not contain a single definition of such concepts as "synonymy" or "meaning" that would meet minimal standards of clarity.

There is a vast body of philosophical literature that sheds grave doubts on the theoretical viability of the so-called "idiom of meaning". For example, Quine, in (Quine 60, chpt.2) and elsewhere, elaborates and defends the "indeterminacy thesis of translation". The thesis is supported by considering the objective evidence that might be used to decide which among infinitely many possible mappings of the sentences of one language onto those of another constitute translations. Put crudely, the thesis says that for any given pair of languages, "translation manuals" can be constructed which are both in perfect agreement with all the available objective evidence and which conflict in that they provide, for infinitely many sentences in one language, translations in the other language that are not equivalent in "any plausible sense of equivalence however loose". (Quine 60, p.27).

The indeterminacy thesis, if correct, could be used to show that theories based on a narrow circle of intensional notions like "synonymy", "ambiguity" etc. are defective in principle, and that there is no fixed vocabulary of primitive concepts which could serve as basic components of an all-purpose conceptual scheme. Quine refers to the view that there is a fixed vocabulary of primitive concepts and that words label fixed entities called meanings, as the "mental museum myth". Something very much like this myth underlies the search, by proponents of A₂, for the best canonical meaning representation. Schank's occasional claims that his primitive acts are universal components of human thought is a similar variant of the mental museum myth.

However, the so-called myth and the associated view of understanding as a matter of translating into a unique, "conceptual" medium of representation is apparently confirmed by a large number of psychological experiments. For example, R.C. Anderson has found that people can often recall learned information on the basis of a cue that paraphrases the item studied. (Anderson 71). This would seem to suggest that meanings are stored in a conceptual medium. If the paraphrased cue is encoded into the conceptual representation, then it will match the representation of the studied information, given that the paraphrase is indeed meaning - preserving. In a similar vein, (Brewer 75) shows that people often remember paraphrases of what they have learned, and not the literal wording. This again would seem to suggest that we remember abstract meanings and not properties of lexical items.

These experiments do not, of course, say anything about any particular conceptual representation. They do not even confirm that there is some such unique representation unless one makes the further assumption that the learned material is recalled as easily with the paraphrased cues as with the literal cues. However, the latter assumption is empirically wrong: although the learned material is often recalled with paraphrased cues, it is recalled more often with literal cues. This fact, which has been established by (Hayes-Roth & Hayes-Roth 77) through an evaluation of many psychological experiments, undermines A_2 and similar views: the postulation of a single, conceptual representation cannot account for the fact that people often employ more superficial representations.

((Hayes-Roth & Hayes-Roth 77) have proposed a theory that seemingly accounts for the experimental data better than the usual view on conceptual

representation. According to their "literal representation theory", people store literal information in networks that contain semantic definitions as well as propositions. Storage of items in the network causes a spreading of "activation" to semantically related information. Activation produces "memory traces". During study, memory traces that represent paraphrases are created together with stronger traces that represent literal cues. The "literal representation theory", thus, holds that the basic units of representation are surface words and not abstract items like the graphs of conceptual dependency theory. Further experimental data supporting this theory can be found in (B. Hayes-Roth 78, pp.336 ff). The author concludes from the empirical success of the theory, together with her belief that theoretical parsimony counsels against postulating more than one representational medium, that all information is stored in surface form. Below, in the context of a discussion of matters of efficiency, I will give arguments against this conclusion and in favor of a view which countenances multiple levels of semantic representation.)

Conceptual representations are often described as offering various kinds of efficiencies. For instance, storage efficiency is achieved by mapping all synonymous expressions into one representation. As has been pointed out in the previous chapter, Schank, among others, claims that conceptual representations facilitate inference making. In his system inference rules are associated with primitive acts. Thus, the rules for a given act have to be written only once and need not be repeated for each word whose definition refers to that act. Moreover, attaching rules to low-level primitives makes inference more efficient, or so it is claimed, because the patterns occasioning inference are fully explicit in the representation.

However, it is patently false that decomposing all concepts into their primitive components increases the efficiency of inferencing. When everything is translated into the lowest level, then drawing inferences requires always that very large data structures are searched and matched. It would seem much more reasonable to initiate matches on lower-level structures only if more "superficial" matches, i.e. matches between high-level concepts or lexemes, fail. This precept is simply an instance of the eminently useful "heuristic": "Don't scratch where it doesn't itch". It applies equally well to the logical analysis of ordinary-language arguments: whereas it is true that longer formalizations of arguments, and, among formalizations of a given length, those with fewer distinct components, stand a better chance of turning out valid, it is clear that too fine a symbolization is useless and cumbersome for inference making. Here, as elsewhere, the depth of analysis should be chosen to depend on the purposes at hand. Quine has put the point nicely (in (Quine 60, p.160)) when he recommends to follow a "maxim of shallow analysis" and to "expose no more logical structure than seems useful for the deduction..."

To avoid scratching where it doesn't itch, i.e. to avoid drawing inferences always in terms of low-level primitives, is not just conducive to greater efficiency. The insistence on a single, primitive representation may well make inferencing programs on larger data structures entirely infeasible. This is so because symbolic pattern matching has the clique, subgraph isomorphism, maximum subgraph and graph homomorphism problems as specific instances. The latter problems are all known to be NP-complete, i.e. their worst case times are at least exponential in the number of nodes. Since

graphs representing the meanings of sentences or entire stories in conceptual representation systems would eventually, in non-toy applications, have to become quite large, pattern-matching and inference making would be impossible because of combinatorial explosion. Since inferencing is essential to natural language processing, and is done by pattern matching, it would seem that language processing programs must deploy more superficial, higher-level representations as well .

The previous discussion makes the simple points that one notational system's labored truth is another's banality, and that the complexity of problems depends to a large extent on the language in which they are couched and on the theory brought to bear upon them. The computational infeasibility of tackling all inference problems in a single, low level representation establishes the need to use a shallower level of semantics. But paraphrasing and related linguistic tasks seem to require deep representations. I conclude, then, that an adequate natural language system should be able to work with several levels of representation.

This conclusion, which contradicts A_2 , is also reinforced by research in other disciplines. Investigators of problem solving in humans and machines have found that a facility to shift problem representations constitutes an important ingredient of successful problem solving behavior. Psychologists and cultural anthropologists studying the process of understanding connected discourse have also found it necessary to postulate several representational layers. (For a survey of many psychological, anthropological and computational investigations of text understanding, cp. (Young 77).) Kintsch, for example in (Kintsch 74), has provided some empirical evidence for the

hypothesis that the sentences in a story are organized into a coherent text by "macro structures". Macro structures are constructed during reading and are, in a loose sense, "implied" by the sentences in a story, much as "x has color" is "implied" by "x is red". They help impose a hierarchic organization on the representation of a text by suppressing or summarizing many details. Summarizing details, in turn, reduces processing loads. The evidence also suggests that inference and recall start with macro structures and that sentences corresponding to the latter are recalled longer than more detailed ones.

If some of the above arguments are sound, then the search, by proponents of A₂, for the best meaning representation is misguided because there is no one, and thus no best one, such representation. Indeed, meaning seems to be a much more multi-faceted affair than is usually recognized.

At the very least, the meaning of a word would have to include:

- procedures for recognizing and identifying referents;
- procedures for accessing world knowledge from dictionary entries;
- specifications of what other words can occur with a given word;
- specifications of semantic "selection restrictions";
- indications of functions and possible uses of referents;
- definitions of intensional relations between concepts at various levels of detail.

Most current AI programs, with the prominent exception of Winograd's SHRDLU, are restricted to purely intensional conceptions of meaning. As a

consequence, such programs could be described as being forever lost in idle conversation, with nothing to talk about. By contrast, logicians and philosophers of language have constructed extensional theories which are elaborations of Tarskian model theory (cp. (Montague 77)). Extensional semantics, however, shows no concern for problems of inference or efficiency and little for problems of context. Although formally much more rigorous than Chomskyan linguistics, they are similarly ill-suited as blue-prints for competence models. (For a comparison of model-theoretic and algorithmic semantics, cp (Oppacher 81).)

To summarize, I propose to replace A_2 with A_2' :

A_2' : - Adequate natural language systems should be capable to deploy several levels of semantic and world knowledge representation. Inference should be done on "surfacy", high-level structures whenever possible.

In the following chapter I shall describe a program that is consistent with the alternative view characterized by A_1' and A_2' . However, the program will take only a few preliminary steps towards implementing the alternative view.

The present initial phase of implementation has two main goals: First, to create a highly modular programming environment to explore the usefulness of different representational media and, second, to study the trade-off between syntax and semantics, i.e., to find out how much mileage can be gotten out of a "surface" semantics, given a fairly strong syntax. These goals will be

realized by writing an almost fully transparent parser that generates context-sensitive parses and executes modular, production-like semantic rules. In this way, some of the complexity of natural language processing is hidden away in the parsing component, which functions essentially as the control structure for the entire system.

For pragmatic reasons, I shall tackle the problem of translating restricted sublanguages of a natural language into corresponding sublanguages of another natural language.

The concept of a sublanguage is intuitively quite clear: the "languages" of car repair manuals, of stock market reports, of group theory, of programming manuals, etc., would count as instances. For many such sublanguages, it seems to be the case that the differences between sublanguages of a given natural language are greater than the differences between corresponding sublanguages in different natural languages. However, since I do not know how to define "sublanguage" precisely, I shall make no important, systematic use of this concept. It serves only as a reminder that there are reasonably large, practically important, and relatively well-knit fragments of languages. Such fragments are characterized by several powerful restrictions which can be exploited so as to make some of the tough problems of language processing quite a bit easier. Such restrictions include not only limits on vocabulary size and on the variety of admitted grammatical constructions but also domain-specific semantic regularities.

In short, the program to be described is not meant to be a computational theory of understanding but a tool to facilitate the processing of restricted sublanguages.

CHAPTER FOUR

THE SYNTACTIC COMPONENT

A. CONTROL ISSUES

In the previous chapter, it was pointed out that most extant natural language processing systems have very complicated control structures and are, consequently, difficult to extend modularly. It was conjectured that this is due to the fact that their syntactic expertise is not conveniently located in one routine but, in effect, distributed throughout the entire program. In this chapter, I shall describe a method to remedy the situation and to design modular language processing systems.

A modular system should be designed in such a way that the question of how the program tackles certain common grammatical tasks may be taken to be settled, so that the programmer can concentrate on the content of semantic rules and need not be concerned with control issues or with the problem of having to identify the syntactic constituents to which the rules are relevant. For example, when writing rules to handle semantic ambiguities or complex grammatical phenomena corresponding to transformational rules, the programmer should not in addition have to deal with such questions as to how to assign a syntactically ambiguous word in a given context to one of several categories.

In the system described below, modularization is achieved by making the control structure largely transparent, i.e. by allowing it to reside in the parser. Put slightly differently, the phrase structure tree produced by the parser for a given sentence serves as the control structure which governs the semantic interpretation of that sentence.

The method, in a nutshell, is this: the parser outputs a phrase marker, or, if the analyzed sentence is structurally ambiguous, a list

of several phrase markers. The latter are trees constructed by means of context-free rules that are similar to the base rules in transformational grammars. The parser differs from the more usual context free parsers, however, in that it allows partially constructed phrase structure trees to be modified by functions that inspect the immediate environment of a node about to be entered into the tree. These functions add a limited form of context sensitivity to the essentially context free parser. The tree for a sentence is generated in top-down fashion under the control of context free rules and restricted context sensitive tests. The interior nodes of the tree are occupied by functions corresponding to transformational and/or semantic rules, and the leaf nodes are occupied by the dictionary entries of the words in the surface string. Since LISP, unlike any other high level language, makes no distinction between programs and data structures, the tree generated by the parser can be immediately executed as a program. The tree, interpreted as program, constitutes the control structure referred to above, and governs the semantic interpretation of the sentence whose structure it reflects. The semantic interpretation of a sentence amounts essentially to an execution of its tree in postorder.

As can be seen from this very rough description, control issues involved in processing natural language text are indeed largely taken care of by the syntactic component. However, to eliminate semantically uninterpretable parses and to help resolve subtle syntactic ambiguities, the parser must occasionally communicate with the semantic component. The extent to which this interaction disturbs the highly modular picture just sketched remains to be determined. Very little is known at present about how and to what extent syntax and semantics should cooperate. The goal of

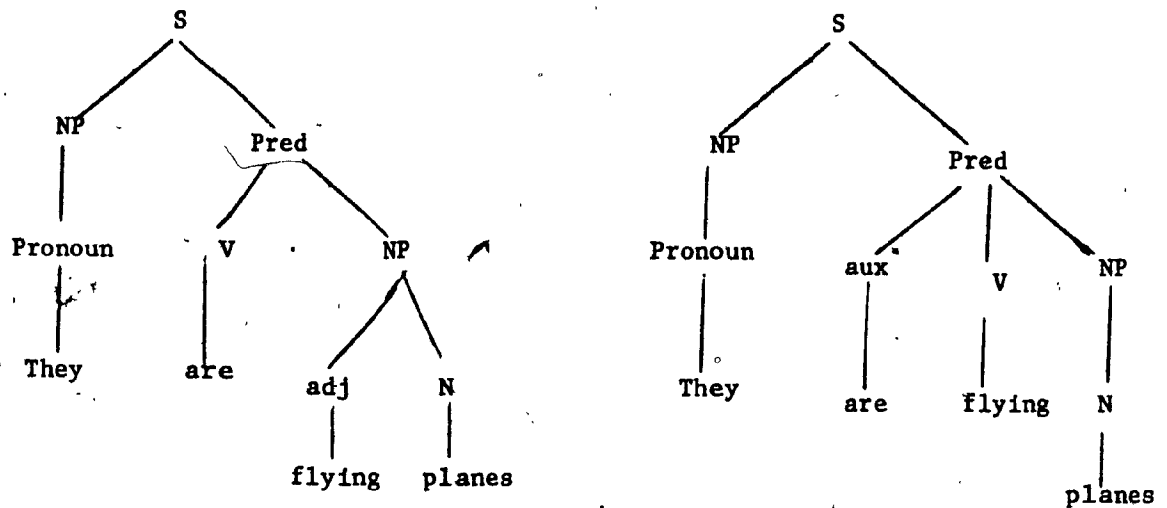
modularity seems best served by keeping syntax and semantics separate while the goal of producing an efficient and linguistically realistic system seems to favour strong cooperation. The system to be described is meant, among other things, to provide an experimental environment for studying this question.

Since the parser does considerably more than just generate derivation trees-- in particular, as will be described below, it initializes variables through which cooperating routines can communicate-- I distinguish it carefully from the recognizer. The recognizer is the context free main loop of the parser.

B. THE PARSER AND THE RECOGNIZER

A natural language has several different kinds of ambiguities. There are word-sense ambiguities (polysemy), referential ambiguities, and structural ambiguities. For example, 1) will be assigned at least the following two derivations by any adequate grammar for English, depending on whether flying is taken to be an adjective or a verb.

1) They are flying planes.



The structural ambiguity of 1) is due to the fact that many words belong to more than one syntactic category. Other types of structural ambiguity are more subtle and harder to detect. In

2) The chicken is ready to eat.

the chicken is the logical subject in one interpretation, and the logical object in another, depending on whether 2) is construed as saying that the chicken is about to eat something or that someone is about to eat the chicken.

Because of the pervasiveness of structural ambiguity in natural languages, any grammar for such a language will have to be ambiguous. A grammar for a language is said to be ambiguous if it generates at least

two distinct (leftmost or rightmost) derivations for at least one sentence of L. A language L is said to be structurally ambiguous if every grammar for L is ambiguous.

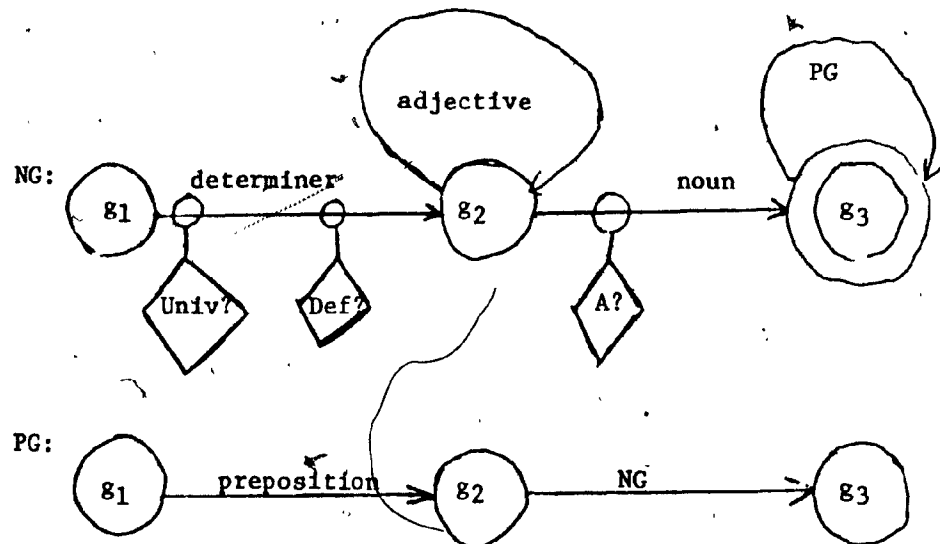
An ambiguous grammar requires a parser that simulates a non-deterministic machine. The usual method of simulating non-deterministic machines on sequential and deterministic devices is to use a backtracking algorithm. Backtracking parsers have mechanisms to make choices based on guesses about probable continuations of a parse and to undo the effects of wrong choices.

A widely used kind of backtracking parser is based on Augmented Transition Network (ATN) grammars. ATN grammars were first described by (Thorne, Bratley, Dewar 68) and by (Bobrow, Fraser 69) and received their currently most popular form in (Woods 70,73). (An extensive survey of works on ATN is given in (Bates 78).)

An ATN is a generalization of a finite state machine. It is a network of nodes and arcs labelled with names of syntactic categories. If an input word belongs to a category that labels an arc, then that arc may be traversed. Several networks can be combined into larger structures by letting arc labels correspond to (possibly recursive) calls to traverse other networks. In addition, arcs may impose restrictions in the form of arbitrary tests on words and may accumulate data gathered by such tests in global registers. If a word in a given state meets the tests on several arcs, the first available arc is traversed. This depth-first regime is usually supported by automatic backtracking.

As an example, consider the simple net for noun groups (NG) whose noun is optionally followed by an arbitrary number of prepositional groups

(PG). Accepting states are represented by double circles and tests by diamond-shaped boxes. The first two tests try to ascertain the universality and definiteness of the article, and the third test checks whether the noun and the article agree in these features.



The (mutual) recursion introduced by the arc labels PG and NG allows for phrases with arbitrarily deep embedding, as in

3) The young girl in the old white house near a blue blue lake....

The tests hung from the arcs would attempt to ascertain whether the determiner is singular or plural and definite or indefinite, and put the results into appropriate registers.

As can be seen from this very brief description, the ATN formalism has the advantage of being very easy to implement. It would be a straightforward task to write an interpreter for rules in ATN form. In such a 'procedural' implementation, the ATN formalism is viewed as a grammar-cum-interpreter, and the interpreter works in the obvious top-down, depth-first, backtracking manner. (For examples of LISP interpreters of ATN

rules, see (Charniak, Riesbeck, McDermott 80, 262 ff) and (Winston 77, 343ff).) However, backtracking parsers have the disadvantage that they may perform a lot of useless work by creating partial parses that do not figure in the final parse. In the worst case, such a parser will take exponential time c^n , where c is a constant depending on the grammar used and n is the length of the input string. If, in order to avoid possibly exponential times, the ATN formalism is viewed simply as a notation to express grammatical rules independently of a particular parsing algorithm, then the advantages of the formalism largely disappear. In the latter case, where a grammatical rule does not itself function as part of a parser, the ATN scheme simply amounts to yet another (and not particularly conspicuous or familiar) notation to write grammatical productions.

For these reasons, I have adopted another method of simulating nondeterminism. This method uses a form of parallelism: the creation of structures corresponding to partial parses which do not belong to the final parse is avoided by pursuing all alternative valid partial parses in parallel, by fusing parse paths whenever possible, and by actually constructing syntactic structures only after one complete parse has been found.

The main loop of the parser used here is an adaptation of Earley's (Earley 70) recognizer. (A recognizer simply accepts or rejects strings whereas a parser constructs derivation trees for well formed strings.) Like Earley's algorithm, it is an entirely general context-free recognizer that works in time n^3 and space n^2 for ambiguous grammars, in time n^2 and space n^2 for nonambiguous but nonlinear grammars and in linear time and space for any grammar that can be parsed with these bounds by any algorithm. The present parser, unlike the recognizer, needs space n^3 to store derivation

trees. It differs from Earley's parser, however, in that it allows partially constructed phrase structure trees to be modified by the results of executing functions which inspect the immediate context of syntactic constituents.

Earley's algorithm has been chosen as the basis of the present parser because it is the most efficient general context-free recognizer known. It has the further advantage of achieving the lower bounds for restricted grammars automatically, i.e. without having to be told what the type of a grammar is that it applies to an input string. A more modest advantage of the algorithm is that it tolerates left recursion and, indeed, that it does not require that productions be written in any particular normal form. The basic idea of the algorithm follows. (To keep the description short, I shall use a very small sample grammar from Earley's paper. A detailed complexity analysis and a correctness proof can be found in (Aho, Ullman 72, 320 ff).)

Let $G = \langle NT, T, S, P \rangle$ be a context-free grammar, with non-terminals NT , terminals T , starting symbol S and productions P . Let $x = x_1 x_2 \dots x_n$ be an input string. The algorithm constructs a list of states S_i for each position $0 \leq i \leq n$ in the input. Each state list S_i represents the state of recognition when the i -th symbol has been scanned. Each state in S_i is a quadruple consisting of

- a) a production whose right hand side has been used to derive the currently examined part of the input,
- b) a dot '.' in the production ($\cdot NT, \cdot T$) which indicates how deep the recognition has been pushed successfully into the right hand side,

- c) an integer to indicate at which position in the input string the search for the present instance of the production was initiated and
- d) a k-symbol look-ahead string which is a valid successor for the present instance of the production.

The look-ahead string can be used in the following manner to cut down on backtracking: no attempt is made to pursue an alternative A, if no prefix of the remaining input is contained in $\text{first}_k(A)$. $\text{first}_k(A)$ is defined as the set of all k-symbol prefixes of terminal strings derivable from A or of shorter terminal strings derivable from A:

$$\text{first}_k(A) = \left\{ t \in T \mid \begin{array}{l} (A \xRightarrow{*} tB, \text{ for some } B \text{ and } \text{length}(t) = k) \text{ or} \\ (A \xRightarrow{*} t \text{ and } \text{length}(t) < k) \end{array} \right\}.$$

Thus, a state $S = \langle A \rightarrow \alpha \cdot \beta, j, l_k \rangle$ with look-ahead string l_k is in state list S_1 if there is some derivation $S \xRightarrow{*} \delta A \psi$, for some δ and ψ , such that $\delta \xRightarrow{*} X_1 \dots X_j$, $\alpha \xRightarrow{*} X_{j+1} \dots X_i$, $X_{i+1} \dots X_{i+k} \in \text{first}_k(\beta)$. In other words, the portion of X derived from α is bracketed by i of S_1 and j in S. $X \in L(G)$ iff some state $\langle S \rightarrow \lambda \cdot 0 \rangle \in S_n$.

To fix our ideas, let

$$G = \left\langle \left\{ E', E, T, P \right\}, \left\{ \text{id}, +, *, \dagger \right\}, E, \left\{ \begin{array}{l} E \rightarrow T, E \rightarrow E + T, \\ T \rightarrow P, T \rightarrow T * P, P \rightarrow \text{id} \end{array} \right\} \right\rangle \quad \text{and let}$$

$$X = \text{id} + \text{id} * \text{id}.$$

E' is used only to initialize the algorithm in state list S_0 . In the following, k is assumed to be 1 and ' \dagger ' functions as a terminator. S_0 originally contains only the state

$$(1) \quad \langle E \rightarrow \cdot E \mid , \mid , 0 \rangle$$

If $S \in S_1$ has a non-terminal to the right of the dot, we add to S_1 a new state for each alternative of the non-terminal. In each such case, the dot is put before the newly added production's right hand side to indicate that we have not yet seen any strings which might start at position $i + 1$. The pointer of each new state is set to i to indicate that we began looking for an instance of its production in state list S_1 . This operation is called, accordingly, the predictor. Thus we add to S_0

$$(2) \quad \langle E \rightarrow \cdot T, \mid 0 \rangle$$

$$(3) \quad \langle E \rightarrow \cdot E + T, \mid 0 \rangle$$


The same operation applied to (2) and (3) yields

$$(4) \quad \langle T \rightarrow \cdot \cdot P, \mid \cdot \cdot 0 \rangle$$

$$(5) \quad \langle T \rightarrow \cdot T * P, \mid , 0 \rangle$$

$$(6) \quad \langle E \rightarrow \cdot E + T, +, 0 \rangle$$

$$(7) \quad \langle E \rightarrow \cdot T, +, 0 \rangle$$

The look-ahead symbol is ' \mid ' in (2) and (3) because it occurred after E in (1), and it is again ' \mid ' in (4) and (5) because it occurred after T in (2). It is '+' in (6) and (7) because the E in (3) is followed by '+'.


Next we add

$$(8) \quad \langle T \rightarrow \cdot P, +, 0 \rangle$$

$$(9) \quad \langle T \rightarrow \cdot T * P, +, 0 \rangle$$

$$(10) \quad \langle T \rightarrow \cdot P, *, 0 \rangle$$

$$(11) \quad \langle T \rightarrow \cdot T * P, *, 0 \rangle$$

(12) $\langle P \longrightarrow . id, +, 0 \rangle$

(13) $\langle P \longrightarrow . id, +, 0 \rangle$

(14) $\langle P \longrightarrow . id, *, 0 \rangle$

No more states can be added to S_0 by the predictor. A new operation, the scanner, is now applicable to (12) - (14). The scanner operation takes a state in S_1 with a terminal to the right of the dot and compares the terminal with X_{i+1} . If there is a match, the state, but with the dot moved over the terminal, is added to S_{i+1} . In the example id is scanned first and, thus, S_1 will contain

(15) $\langle P \longrightarrow id ., +, 0 \rangle$

(16) $\langle P \longrightarrow id ., +, 0 \rangle$

(17) $\langle P \longrightarrow id ., *, 0 \rangle$

To S_1 a third operation, the completer, is applicable. The completer takes a state S in S_1 whose dot is at the end of the production and compares that state's look-ahead string with $X_{i+1} \dots X_{i+k}$. If there is a match, the completer returns to the state list indicated by the integer in S , and adds to S_1 all states from the indicated earlier state list that have the current state's left-hand side to the right of the dot, moving the dot over. Thus, the completer adds to S_1

(18) $\langle T \longrightarrow P ., +, 0 \rangle$

(19) $\langle T \longrightarrow P ., *, 0 \rangle$

(20) $\langle T \longrightarrow P ., +, 0 \rangle$

In the example, $X_2 = +$ and the completer is applied to (18), yielding

$$(21) \quad \langle E \longrightarrow T \cdot, +, 0 \rangle$$

$$(22) \quad \langle E \longrightarrow T \cdot, \uparrow, 0 \rangle$$

$$(23) \quad \langle T \longrightarrow T \cdot * P, +, 0 \rangle$$

$$(24) \quad \langle T \longrightarrow T \cdot * P, \uparrow, 0 \rangle$$

$$(25) \quad \langle T \longrightarrow T \cdot * P, \downarrow, 0 \rangle$$

Only (21) leads anywhere and the completer adds

$$(26) \quad \langle E \longrightarrow E \cdot + T, +, 0 \rangle$$

$$(27) \quad \langle E \longrightarrow E \cdot + T, \uparrow, 0 \rangle$$

$$(28) \quad \langle E' \longrightarrow E \cdot \uparrow, \uparrow, 0 \rangle$$

The scanner is now applied to (26) and (27) and adds

$$(29) \quad \langle E \longrightarrow E + \cdot T, +, 0 \rangle$$

$$(30) \quad \langle E \longrightarrow E + \cdot T, \uparrow, 0 \rangle$$

to S_2 . And so it goes.

If the algorithm produces a state list S_{n+1} whose only element is

$$\langle E' \longrightarrow E \cdot \uparrow, \uparrow, 0 \rangle, \text{ then the scanned string is an element of}$$

$L(G)$, otherwise the string is rejected.

To describe the recognition algorithm more precisely, a few notational conventions are needed. Let the productions of G be arbitrarily enumerated from 1 to $p-1$. The 0-th production has as its left side a non-terminal L_0 that occurs in no other production and as its right hand side, the starting symbol of G followed by the terminator $\uparrow \in T$: $L_0 \longrightarrow S \uparrow$. Each production of G is of the form $L_\ell \longrightarrow R_{\ell 1} \dots R_{\ell k_\ell}$, $1 \leq \ell \leq p-1$, where k_ℓ is the number of symbols on the right side of the ℓ -th production. Each state is of the form $\langle \ell, j, b, \alpha \rangle$ where ℓ ($1 \leq \ell \leq p-1$) is the production number, j ($0 \leq j \leq k_\ell$) is the number of the symbol on the right side of the

ℓ -th production which has already been recognized (i.e. the dot would go to the right of the j -th symbol), b ($0 \leq b \leq n+1$) is the back pointer to the state list S_b , where we began to look for the current instance of the ℓ -th production, and α is a k -symbol look-ahead string.

Given G , k , $X_1 \dots X_n$, the algorithm is initialized as follows:

$$X_{n+1} := \neg, 1 \leq i \leq k+1$$

$$S_1 := \emptyset, 0 \leq i \leq n+1$$

$$S_0 := \langle \langle 0, 0, 0, \neg \rangle \rangle$$

It is assumed that state lists are maintained as queues and that a new state is inserted at the end of a list, if it is not already present. X is accepted iff S_{n+1} contains as its only element $\langle L_0 \rightarrow S \neg, 0, \neg \rangle$ or, rather $\langle 0, 2, 0, \neg \rangle$.

for $i := 0$ to n do

begin for all $S = \langle \ell, j, b, \alpha \rangle \in S_i$ do

begin if $(j \neq \ell$ and $R_{\ell, j+1} \in NT)$ then

begin for all q such that $R_{\ell, j+1} = L_q$ and

for all $\beta \in \text{first}_k (R_{\ell, j+2} \dots R_{\ell, \ell})$ do

$$S_i := S_i \cup \langle \langle \ell, 0, 1, \beta \rangle \rangle \text{ end}$$

else if $(j = \ell$ and $R_{\ell, j+1} \in T)$ then

begin if $R_{\ell, j+1} = X_{i+1}$ then $S_{i+1} := S_{i+1} \cup \langle \langle \ell, j+1, b, \alpha \rangle \rangle$ end

else if $(j = \ell$ and $\alpha = X_{i+1} \dots X_{i+k})$ then

for all $S' = \langle q, m, g, \beta \rangle \in S_b$ such that $R_{q, m+1} = L_\ell$ do

$$S_i := S_i \cup \langle \langle q, m+1, g, \beta \rangle \rangle$$

end;

if $S_{i+1} = \emptyset$ then reject the input string

else if $i = n$ and $S_{i+1} = \langle \langle 0, 2, 0, \neg \rangle \rangle$ then accept the input string

end.

Pratt describes (Pratt 75, 423 ff) a parser that combines the Earley and the Cocke-Kasami-Younger algorithms and claims it to be considerably faster in practice than the Earley algorithm. But Pratt does not claim to have reduced the worst case $O(n^3)$ bound. Since I have not yet been able to implement his parser, I don't know whether it would process my grammar faster and if so by how much than the Earley algorithm. Pratt's system, however, has greatly influenced the design of my own. In particular, I have adopted his proposed solution to the 'marker' problem. Using his examples:

- 1) No mother of such twins has time to relax.
- 2) The mother of no such twins has time to relax.
- 3) The mother of such twins does not have time to relax.
- 4) The mother of such twins has no time to relax.
- 5) The mother who has no twins has time to relax.

the marker problem can be characterized as follows (Pratt 73,376ff): in each of the sentences 1)-4) above, the negation applies to the entire sentence although the 'negation marker' occurs in different places. An unappealing way to handle this situation would be to write different rules for all possible marker positions. If one were to try to avoid this proliferation of rules by using a global negation variable that could be set by any occurrence of a negation marker, then 5) would be analyzed incorrectly (because it makes a positive claim). Similar difficulties arise with respect to determiner-noun or subject-verb agreements. In all these cases, the necessary information cannot be held in global variables because different sub-clauses need different marker information. Embedded sentences, for instance, have non-global scopes for various kinds of agreements.

Consider

6) The girls whose father is not young **are** pretty.

Not only is the negation in 6) confined to the sub-sentence, but the subject-verb agreement markers have to be different for the main noun and the noun in the relative clause.

Pratt's solution is remarkably simple and elegant: markers are to be treated as LISP LAMBDA and PROG variables local to the clause that needs the information! Adopting this solution to the marker problem, however, requires a slight change in the LISP interpreter function EVAL. It will be remembered from the discussion in section A that the phrase structure tree generated by the parser is to be executed 'essentially' in postorder. Suppose, then, that a sub-tree root is occupied by function f_1 and its children by functions f_2 and f_3 . Suppose further that f_1 has two LAMBDA variables to be bound to the results of evaluating f_2 and f_3 , and that the latter should set two markers m_1 and m_2 whose values should be known to f_1 . That is, we assume that f_1 should combine the results of f_2 and f_3 depending on the values of m_1 and m_2 . Clearly, m_1 and m_2 cannot be further LAMBDA variables of f_1 .^{*} Because of scope problems, we also do not want them to be global to f_1 . We would like m_1 and m_2 to be local to f_1 but global to all functions below it in the tree. Unfortunately, it would not do to have m_1 and m_2 simply as PROG variables of f_1 because they would be initialized to NIL as soon as f_1 was applied to the results of its children. The solution I have chosen is to treat m_1 , m_2 as PROG variables of f_1 but to initialize them before f_2 and f_3 are executed in postorder. The tree is processed by performing a preorder initialization of PROG variables and postorder EVALuation. The modified EVAL function looks somewhat like this:

```

(eval' (tree)
  (cond (tree (initialize prog variables of tree)
    (eval' (leftchild tree))
    (eval' (rightchild tree))
    (eval tree ))))

```

C. SYNTACTIC 'UTILITY' FUNCTIONS

In addition to the parser, the syntactic component contains a number of utility functions to ease the task of writing grammatical and semantic modules.

Task-Specific Functions

This group includes functions that deal with the details of French word endings, liaison, conjugation and so forth. For example, when the parser has determined that a given verb is active, in present tense, third person, plural, then a conjugation function will get the verb's conjugation class from the dictionary and add the appropriate ending to the word stem. Similar functions construct the plural forms of nouns and the feminine and plural forms of adjectives. Another function skims through the French sentence just before it is printed and tidies it up. For instance, 'de le' is replaced by 'du' unless the next word begins with an 'h' or a vowel; 'de les' is replaced by 'des', 'le', 'la', 'me', 'que' etc. are apostrophized when followed by a word beginning with a vowel; 'ma', 'ta', 'sa' are replaced by 'mon', 'ton', 'son' when the next word begins with a vowel and so on.

General Utility Functions

This group includes functions that are useful no matter what the target language might be. There are, for instance, functions to interactively enter

new grammatical rules and dictionary entries and to perform limited work space editing of poorly written entries.

Dictionary entries are stored as property lists on word stems and are thus retrieved by the usual LISP hashing method from the OBLIST in roughly constant time.

A very powerful suffix removal algorithm, CHOPSUFFIX, is used to cut down on the number of words that have to be stored in the dictionary. CHOPSUFFIX works for any English suffix. (To prevent acute boredom I shall not describe the fairly long algorithm in detail. A few examples may suffice.)

CHOPSUFFIX gets as input a word and tries to look it up in the dictionary. If the word is not there, then the algorithm checks if the word has one of six possible suffix-ending last letters: 'S', 'G', 'D', 'R', 'Y', 'T'. Suppose the last letter is 'S'. 'S' is chopped off and the dictionary is checked again. If there was no success, then the currently last letter is inspected. If it is an 'E', it is chopped. If now there is an 'I', it is changed to 'Y' and the dictionary is searched for the last time. If after chopping the 'S', the last three letters were 'ING', 'ING' is chopped and an 'E' is added at the end and the dictionary used once again. If there was no success, the 'E' is chopped again. If now there are two identical letters at the end, the dictionary is searched for the final time. And so on. The rules just sketched would reduce 'boys' to 'boy', 'cries' to 'cry', 'undertakings' to 'undertake', 'hittings' to 'hit'. CHOPSUFFIX handles even such outlandish cases as 'truly-true', 'wholly-whole' etc. in a fairly general manner. After having reduced, say, 'loveliest' to 'love', the suffix of 'love' would indicate that the word was a superlative.

CHAPTER FIVE

GRAMMATICAL ISSUES

A. STRUCTURAL LIMITATIONS OF CONTEXT-FREE RULES

In the first pass, before more sophisticated processing takes place, an input sentence is parsed with a context-free grammar. (As mentioned earlier, this is not entirely correct because the parse can be made to depend on the outcome of context sensitive tests.) There are several good reasons for starting like this: context-free grammars are understood far more deeply than context sensitive ones, and context-free parsers are more efficient than context sensitive ones.

To avoid being in the situation of the drunk who looked for his key in a well lit area and not in the dark alley where he dropped it - presumably on the grounds that looking in the light is more efficient than looking in the dark - a common objection to the use of context-free grammars in natural language processing needs to be overcome. The objection is based on the obviously correct observation that many important aspects of a natural language cannot be captured in a context-free grammar. While this is perfectly true, it overlooks the possibility that the crude and relatively cheaply available context-free approximation to the structure of a sentence might contain enough information to tackle context-sensitive and semantic problems successfully. I happen to believe that a context-free parse does indeed elicit enough structure to greatly facilitate answering further, more complicated questions. For instance, I believe that it is cheaper in terms of both processing time and programming time to address semantic questions with the knowledge derived from a context-free analysis, than to try to get by without such knowledge. Whether this belief is true or not depends on the number and variety of linguistic phenomena that the present approach can treat efficiently and without heroic programming efforts.

Here, as elsewhere, the proof of the pudding is in the computing.

The limitations of context free grammars are best illustrated by some examples. Suppose we have the following grammar (NP=noun phrase, N=noun, PN=proper noun, DET=determiner, VP=verb phrase, V=verb, S=sentence):

- 1) S \longrightarrow NP + VP
- 2) NP \longrightarrow DET + N
- 3) NP \longrightarrow N
- 4) VP \longrightarrow V + NP
- 5) DET \longrightarrow the/a
- 6) N \longrightarrow boy|girl|boys|girls|Jane
- 7) V \longrightarrow hit|hits|loves|love|picks|pick...

This grammar will generate sentences like

- 8) The boy loves the girl.
- 9) The boy loves Jane

but also sentences like

- *10) The boy love the Jane.
- *11) The Jane hits boy.

To prevent noun phrases like the Jane we simply introduce a new category PN of proper nouns and replace 3) and 6) by

- 12) NP \longrightarrow PN
- 13) N \longrightarrow boy|girl|boys|girls...

and add

- 14) PN \longrightarrow Jane.....

Since the modified grammar still allows sentences like

*15) The boy love Jane.

*16) The boys loves a girls.

we might wish to add rules to control N-V and DET-N agreement. This might lead to the following grammar:

- 1') S → NP^SPLUR + VPPLUR
 1'') S → NPSING + VPSING
 2') NPPLUR → DETPLUR + NPLUR
 2'') NPSING → DETSING + NSING
 3') NPSING → PN
 4') VPPLUR → VPLUR + NPPLUR
 4'') VPSING → VSING + NPSING
 5') DETPLUR → the
 5'') DETSING → a|the
 6') NPLUR → boys|girls...
 6'') NSING → boy|girl...
 7') VSING → loves|hits|picks
 7'') VPLUR → love|hit|pick

To handle a simple form of agreement, we had to introduce many new nonterminals and productions. If we were to try to cope with additional types of agreement, tenses, sentence modalities etc. in a similar manner, we would find that our grammar grows multiplicatively. Such a proliferation of nonterminals and rules would soon render our grammar practically useless. It would be theoretically inadequate as well because the unanalyzable new nonterminals would prevent us from formulating important linguistic

generalizations. For example, 2') and 2'') have, as far as our grammar is concerned, nothing at all in common. Some linguists might even object to the facile introduction of the category PN (in 12) and 14) above) on the grounds that the modified grammar does not allow us to show how the single category of nouns is represented by different sub-categories in different contexts. This objection could be easily met by adding these rules to 1) to 5), 7) and 8):

17) DET + N \longrightarrow DET + COMMONNOUN

18) V + N \longrightarrow V + PN

and changing 6) to

19) COMMONNOUN \longrightarrow boy|girl...

Similar context sensitive rules could be written to achieve N-V agreement without unduly inflating the set of productions and nonterminals. However, the introduction of context sensitive rules would make parsing much more time consuming. Both theory and programming technology are very much less developed for context sensitive grammars than for context free ones. But even if one were prepared to overlook this disadvantage, there are numerous syntactic phenomena that cannot be adequately captured by any phrase structure grammars. Consider, for example:

20) The girl picks up the boy.

21) The girl picks the boy up.

Given a new category PART (particle), we might try to account for the synonymy of 20) and 21) as follows (round parentheses in rules indicate optionality):

22) PART → up

23) VP → V + (PART) + (NP)

24) VP → V + (NP) + (PART)

The problem with these rules is that whenever the object NP is a pronoun,

24) must be used:

25) The girl picks him up.

*26) The girl picks up him.

Since 23) or 24) must be chosen before NP is expanded, it would be futile to consider adding a rule like:

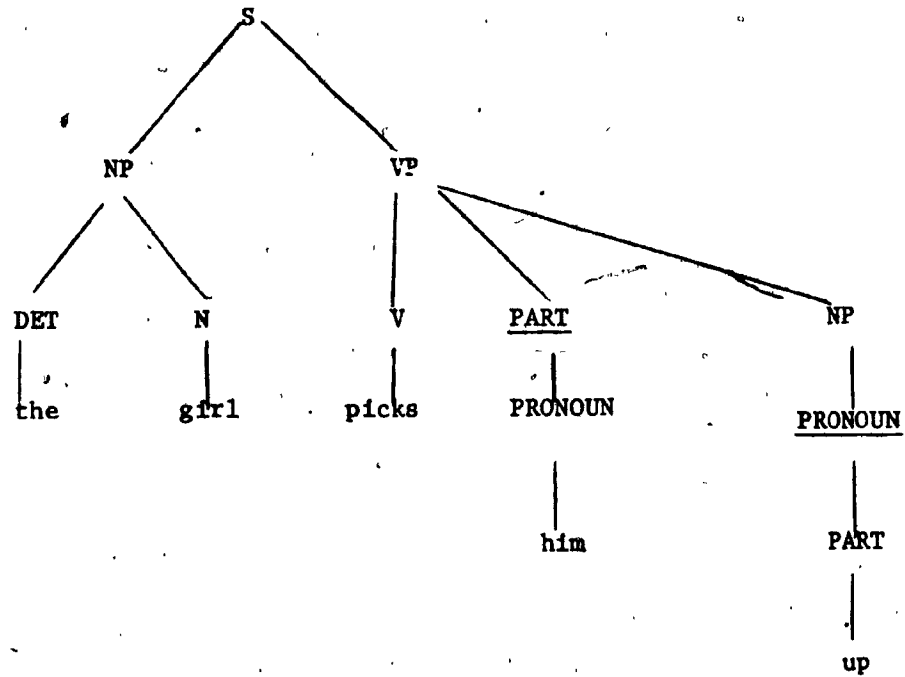
27) NP → PRONOUN

But we could add to 22), 23) 27)

28) PART + PRONOUN → PRONOUN + PART

29) PRONOUN → him|it...

A glance at the derivation tree for 25) shows that this solution is still inadequate.



For one thing, this tree would commit us to the silly claim that particles are pronouns and pronouns, particles. Worse still, our grammar would now accept sentences like

*30) The girl loves up the boy.

What has gone wrong is, of course, that verbs and particles are selected in context free manner. To fix all of this, we need a transformation to express the fact that when a V and a PART are followed by a PRONOUN, the PART must go to the right of the PRONOUN.

Instead of trying to control agreements, tenses etc. by means of

an unwieldy rapidly expanding context free grammar or an inefficient and occasionally inadequate context sensitive one, the present system uses very simple context free rules for the first pass and subjects the resulting structure to procedural versions of transformational rules. Since the current implementation translates from English into French, the first pass produces a structure akin to an English phrase marker. The latter is then operated upon by programs whose effects are roughly comparable to French transformational rules. ('Roughly' because there really are no known transformation rules to map an English phrase marker into a French surface sentence. The process will be described in some detail below.)

B. GRAMMATICAL RULES

Each grammatical rule used here is a list of three elements:

(CF CS TS).

The first element, CF, is a context free rule with at least one and at most two nonterminal symbols at the right. Rules whose right hand sides are terminals are parts of dictionary entries. The restriction to at most two nonterminals at the right of CF is inessential, i.e. it is not required for the parser but it simplifies some of the tree-modifying functions by guaranteeing that all trees are binary. If a rule starts out with $n > 2$ right nonterminals, it is replaced by $n-1$ rules involving $n-2$ new nonterminals. For example, the rule

$$X \longrightarrow A + B + C + D$$

is replaced by

$$X \longrightarrow A + Y$$

$$Y \longrightarrow B + Z$$

$$Z \longrightarrow C + D.$$

Since the newly introduced nonterminals make the rules unpleasant to read, they and the rules requiring them are omitted below.

The second element of a grammatical rule, CS, is optional and consists of a LISP function to perform a context sensitive test on two adjacent constituents. This feature is rarely needed and will not be further described.

The third element, TS, consists of an arbitrarily complex LISP program corresponding to a transformational or semantic rule. TS is executed at the time when the parser, in its postorder evaluation of the tree, processes the node corresponding to the left terminal of the associated CF. The model here is simply that of a syntax directed translation scheme. (Aho, Ullman 72, Vol,2).

The TS component of a grammatical rule may call routines that modify or interrogate a data base, disambiguate word senses or assign a translation to a node. In the latter case, the assigned translation is a function of the translation(s) assigned to the child(ren) of the node in question.

The following are some examples of rules that appear in the CF component of grammatical rules (NG=noun group, PRED=predicate, CONP=conjoined NP, CONJ=conjunction PREPG=prepositional group, PREF=preposition, RELG=relational group, RELPRON=relative pronoun):

- 1) S → NG + PRED
- 2) NG → NP
- 3) NG → NG + CONF
- 4) CONF → CONJ + NG
- 5) NG → NG + PREPG
- 6) PREPG → PREP + NG
- 7) NG → NP + RELG
- 8) RELG → RELPRON + PRED
- 9) RELG → RELPRONG + SS
- 10) SS → REL1 + PRED
- 11) REL1 → NP + RELG
- 12) REL1 → NP
- 13) NG → THATCL + S
- 14) NP → DET + NP
- 15) NP → ADJ + NP
- 16) PRED → V + NG
- 17) NP → N

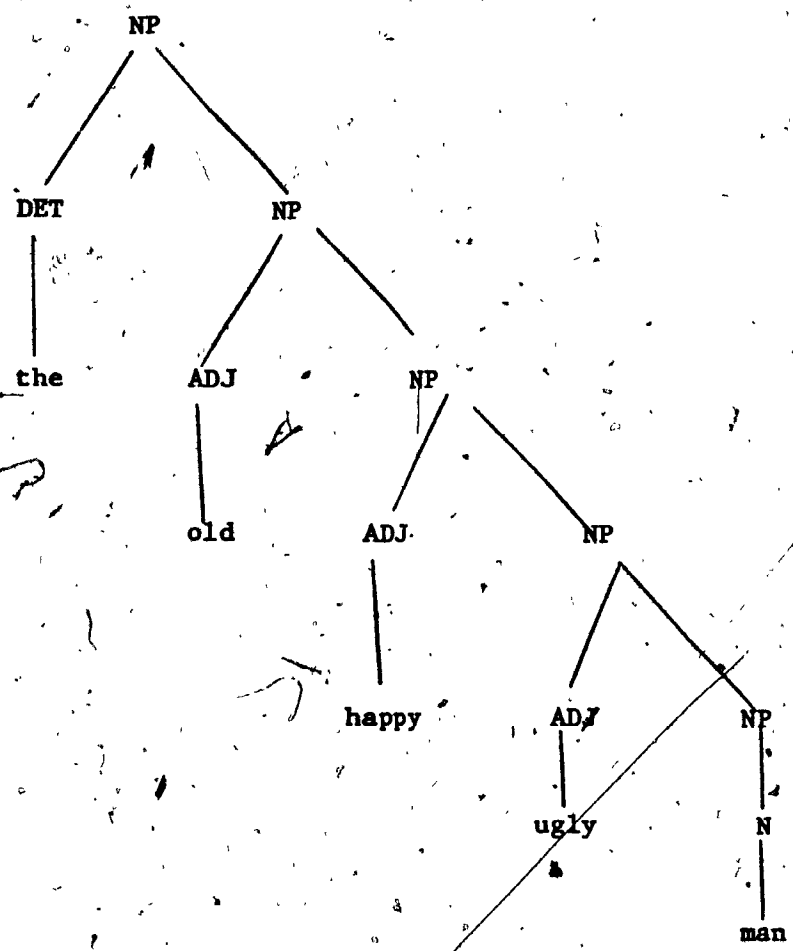
Other rules, not shown here, deal with verb phrases, transitive and intransitive verbs, adverbials, auxiliaries and quantifiers.

A few examples will now be given to illustrate the functioning of the nonsemantic parts of TS components. Suppose we wish to translate the noun phrase

18) The old, happy, ugly man.

Using 14), 15) and 17), the following is constructed:

19)



The TS component associated with 14), i.e. with the root of the tree, is essentially a program that concatenates the list resulting from the evaluation of the left branch (LB) with the list resulting from the evaluation of the right branch (RB). However, in French, the determiner must agree with the noun phrase in gender (GEND=M,F) and universality (UNIV=SING,PLUR). To achieve this agreement, the TS component of 14) evaluates the RB first, saves the results, and concatenates the list resulting from the evaluation of the LB

with the saved result. The evaluation of the RB has, of course, the desired side effects of setting GEND ← M and UNIV ← SING. The dictionary entry for the has a little program to check UNIV and GEND: if UNIV=PLUR then the list (les) is returned else, depending on GEND, (le) or (la) is returned. In the course of evaluating the RB of 19) the TS component associated with 15) is evaluated. This function does the following: to achieve adjective-noun agreement, the noun phrase is evaluated first and concatenated with the result of evaluating the adjectives. Eventually, this produces a list of a noun phrase consisting of a noun followed by one or more adjectives. This list is given to the function ARRADJ:

```
(ARRADJ(LIS)
(PROG(TEMP)
(COND((NULL(CDR LIS))(RETURN LIS))
((EQUAL(LENGTH LIS)2)(COND((GET(CADR LIS)"PRE)
(RETURN(CONS(CADR LIS)(LIST(CAR LIS))))))
(T(RETURN LIS))))))
(SETQ TEMP(REVERSE(CDR LIS)))
(COND((GET(CAR TEMP)"PRE)(SETQ LIS(CONS(CAR TEMP)(LIST(CAR LIS))))
(SETQ TEMP(CDR TEMP)))
(T(SETQ LIS(LIST(CAR LIS))))))
(RETURN(APPEND LIS(BETWEEN TEMP"ET)

BETWEEN(LAT ARG)
(COND((NULL(CDR LAT))LAT)
(T(APEND(CONS(CAR LAT)(LIST ARG))(BETWEEN(CDR LAT) ARG))
```

As can be seen, ARRADJ checks if the first adjective after the noun has the marker PRE and if so, puts the adjective before the noun. Any remaining adjectives after the noun are separated by 'et'. In the case of 18) this gives the result

19) le vieux homme heureux et laid.

Adjectives that are commonly used before nouns in French are marked 'PRE' in the dictionary. All adjectives are provided with the appropriate endings by programs in their dictionary entries.

Suppose we wish to translate

20) The girls did not remember the man.

and we have already evaluated the LB of 1), i.e. the first noun phrase in 20). In the course of working on the RB of 1) we must evaluate the TS component associated with 16) which looks like this:

(DUDE(NEGATE(TIMEIT(APPEND(EVALUATE LB)

((LAMBDA(PERS UNIV GEND) (EVALUATE RB)) 3"SING"M)

Evaluating V first, we find that the tense is past and the negation marker NEG is to be set to true. From the dictionary, we find that remember is translated as rappeler, that rappeler is reflexive and requires être in its past form and that it is in the conjugation class REG1. GEND, UNIV and PERS are set to 'F', PLUR and 3, respectively, from the previous evaluation of the noun group the girls. This information is attached to the word stem of souvenir and returned as the result of invoking (EVALUATE LB).

Before RB, i.e. the noun group of the predicate, is evaluated the PERS, UNIV and GEND markers are reset to their default values. The resulting list is now given to the function TIMEIT. This function tries to determine the tense of the verb and conjugates it accordingly. NEGATE checks if the NEG marker is true and, if so, inserts ne and pas at the proper places, i.e. before and after the first occurrence of an auxiliary or modal auxiliary or main verb. Finally, DUDE inserts de, de la, de le (which later may become du) or des in the right place. 20) is eventually translated as

21) les filles ne se sont pas rappelées de l'homme.

It should be emphasized that the previous paragraph gives only a bare sketch of the action of DUDE, NEGATE and TIMEIT. The definition of each of these functions amounts to about one page of fairly hairy LISP code to cope with numerous details of word order, exceptions and the like. But enough has been said, hopefully, to illustrate the nature of the nonsemantic part of TS. The semantic part will be described in the next chapter.

C. STRENGTHS AND WEAKNESSES OF THE GRAMMAR

Each dictionary entry is a list of two elements: (TP TR). TP is a terminal production, i.e. a production whose right side is a single terminal. TR is an arbitrarily complex LISP program which may set markers or attempt semantic disambiguation but primarily returns a list consisting of the French translation of an English word stem. Let's have a look at some examples.

- 1) (V DISTRUST (CT(REG"MEFI)"(V INFIN MEFIER IMPF MEFI EPASSE
MEFIE PART DE REFLEX)))

2) (ADJ OLD (CT(SMFPMF"VIEUX "VIEILLE "VIEUX "VIEILLES)"(ADJ PRE)))

3) (DET A (CT(CDR(SASSOC GEND'((M UN)(F UNE))NIL))"DET))).

When distrusts appears in the input, the suffix is removed and the stem is marked as having an s - ending. At any rate, by the time the entry for distrust is executed, PERS, UNIV, GEND, and TENSE are known. The function REG in 1) checks TENSE and conjugates the stem accordingly. If TENSE is PRESENT, for example, then REG will use the information provided by UNIV and PERS to select one of the endings e, es, e, ons, ez, ent and attach it to méfi. The result, say, méfi, is now given to the categorizing function CT which attaches to the word the data listed in the second argument. In this case, V is the value of the indicator CAT, MEFIER is the value of the indicator INFIN etc. The indicator EPASSE with its non-nil value MEFIE shows that the past tense of the verb requires être rather than avoir. The flag REFLEX shows that the verb is reflexive. (The indicators and their values may be listed in any order.) Finally, CT returns a list of the value of its first argument. This list, (méfi) in our example, is now used by the TS components of nodes higher up in the tree.

In 2), the function SMFPMF returns one of its arguments depending on whether UNIV is SING or PLUR and GEND is M or F. For regular adjectives like laid, rouge, heureux etc. a function REGADJ is used and is only given the singular, masculine form of the adjective. The French adjective is then categorized as ADJ and, in the case of old, as PRE to indicate that it should precede its controlling noun.

In 3), (un) or (une) will be returned depending on the value to which GEND has been set during the prior evaluation of the controlling noun.

It should be noted that the writing of dictionary entries is

heavily standardized. The addition of new entries, therefore, is a trivial task. Nevertheless, or, maybe partly because of that, I have chosen to keep the dictionary very small. (It contains about 100 entries.) There are several reasons for this decision. For one, I have no particular sub-language in mind for which I should like to achieve reasonably broad expressive coverage. For another, the effort of enlarging the dictionary is not rewarded by new insights. Finally, since the goal of this work is to study design issues in natural language processing and not to create a showy system, it would be a waste of my own and the machine's resources to experiment with more than a handful of instances of each of the categories handled by the grammar.

Most of the syntactic phenomena that the current grammar treats successfully, as well as some important ones it cannot yet handle, will be illustrated in the next few paragraphs.

Since French is much more highly inflected than English, many types of agreements have to be established: agreements between nouns and determiners, between nouns and verbs, between auxiliaries and modals (like must, can, may) and verbs, between nouns and adjectives. Since the scopes of some of the agreements may extend over entire sub-sentences with different agreement features, care must be exercised to prevent scope interferences.

Adjectives

Adjectives can occur in verb phrases as in 'is happy', 'are white' or as noun modifiers. Adjectives themselves can be modified by adjective modifiers like 'very'. A noun can have an arbitrary number of adjectives. There seems to be very little systematic rhyme or reason as to when an

adjective precedes or follows its noun. Some, like color adjectives, typically follow the noun; others, like 'petit', 'grand', 'bon' etc. usually precede it. The policy followed here is this: if an adjective commonly precedes the noun, it is marked PRE in the dictionary and is inserted before the noun if it is the first adjective. Thus,

4) the big, white house

becomes

4') la grande maison blanche

but

5) the white, big house

becomes

5') la maison blanche et grande

Whenever several adjectives follow a noun, 'et' is inserted between any two of them.

Verbs and Auxiliaries

The verbs used fall into seven of the French conjugation classes. At the moment, only present, past, imparfait, plusqueparfait and future tense are recognized. The available auxiliaries (and models) are: have, has, had, am, are, is, was, been, won't, did, didn't, does, doesn't, don't, can, can't, will, must. For example,

'I love the girl.'

'I have loved the girl.'

'I will love the girl.'

'I loved the girl.'

'I won't love the girl.'

'I don't love the girl.'

'I can't love the girl.'

'I did love the girl.'

'I didn't love the girl.'

'I had loved the girl.'

are translated properly. GEND and UNIV agreements for verbs whose past tense requires être are also taken care of:

6) The girls loved the boy.

7) The girls remembered the boy.

8) The boys remembered the girls.

9) The boy and the girl remembered the man.

become respectively;

6') Les filles ont aimé le garçon.

7') Les filles se sont rappelées du garçon.

8') Les garçons se sont rappelés des filles.

9') Le garçon et la fille se sont rappelés de l'homme.

As can be seen from 9'), if a noun phrase contains at least one noun of male gender, then this feature controls the following verb. If there are several nouns, even if they are all singular, the controlled verb is plural.

10) I have the car.

11) I have had the car.

12) I had had the car!

become respectively

10') J'ai la voiture.

11') J'avais la voiture.

12') J'avais eu la voiture.

but both

13) I had the car.

and

14) I did have the car.

go into

13') J'ai eu la voiture.

Similarly,

15) I will not buy the car.

and

16) I won't buy the car.

both become

15') Je n'achèterai pas la voiture.

Relexive verbs are marked as such in the dictionary. Thus,

17) We distrust John.

becomes

17') Nous nous méfions de Jean.

Have and be and their variants are classified both as verbs and as auxiliaries. The grammar contains the rule: $V \longrightarrow AUX + V$. This simple approach in parsing leads sometimes to slight complications in the translation phase. For example, have had in

18) I have had five dollars.

becomes first ai avais, and only after an additional test, ai eu.

Negations

Negations are recognized by the presence of such auxiliaries as doesn't, don't, won't, can't, cannot, didn't or not. Some of them set tense markers, all of them set NEG to true, and none returns anything but the empty list. The actual insertion of ne and pas or aucun happens higher up in the tree when the entire verb phrase is known. Occasionally, as when 19) is translated into 19'), ne and pas do not immediately flank the first verb or auxiliary in a verb phrase:

19) We don't distrust John.

19') Nous ne nous méfions pas de Jean.

Negations also have the effect of changing des to de:

20) I have friends.

20') J'ai des amis.

21) I don't have friends.

21') Je n'ai pas d'amis.

Both of the following

22) She had no friends.

23) She didn't have any friends.

become

22') Elle n'avait pas d'amis.

De, de la, du, des

de, in one of its senses, is used before mass or bulk nouns. In such cases, the program inserts de and the appropriate determiner before the noun. De le and de les are later changed to du or des respectively, unless they occur in the scope of a negation in which case the article is deleted.

Accordingly,

24) The cat drinks milk.

25) The cat doesn't drink milk.

become

24') Le chat boit du lait.

25') Le chat ne boit pas de lait.

However, de is not inserted before a noun that begins a sentence or a subsentence.

becomes first

52) All the girls are beautiful.

and then

51') Toutes les filles sont belles.

Some gives rise to further problems: if some occurs at the beginning of a sentence and the next noun is not a mass term, then it is translated as il y a; if some is at the beginning and the following noun is a mass term then it becomes un peu de; if some is not at the beginning it becomes quelque.

53) Some dogs are happy.

53') Il y a des chiens qui sont heureux.

54) Some wine is good for you.

54') Un peu de vin est bon pour toi.

55) Some girls have some friends.

55') Il y a des filles qui ont quelques amis.

A similar construction employs there as a noun phrase, as in

56) There are friendly dogs.

Unfortunately, there also functions as a pronoun in

57) The dog is there.

Conjunctions

Conjunctions of entire sentences such as

58) The girl loves the cat and the boy loves the dog.

are trivial. It is also easy to process conjoined noun phrases. For example,

59) The ugly boy and the beautiful girl loved the dog.

becomes

59') Le garçon laid et la belle fille ont aimé le chien.

Such cases are parsed according to the rules

60) NG \rightarrow NG + CONP

61) CONP \rightarrow CONJ + NG

and the UNIV marker is set to PLUR. ('Loved', by itself, reveals neither UNIV nor GEND.) However, rules similar to 60) and 61) offer no help with many frequently encountered types of conjunctions. The problem with

62) He played with, teased, and then walked away with his dog.

is that the incomplete conjunctions in this sentence have different structures.

A similar difficulty arises in the case of 63) and 64).

63) He usually plays with and walks Fido.

64) He usually plays with Fido and he usually walks Fido.

It would be inappropriate to analyze 63) as having the structure of 64)

because the two sentences are not equivalent: 63) implies 64) but not vice-versa.

As can be seen from the above examples, the present system can treat a fairly wide range of syntactic phenomena reasonably well. Certain extensions would be straightforward. For example, possessive adjectives could be introduced immediately by simply marking them as PRE. Additional tenses and subjunctives could be introduced as soon as new conjugation functions became available. Questions and commands could be handled through minor extensions of the grammar, and so on. There are also a number of phenomena, like those exemplified in 62) and 63), that would seem to require major changes. (But the really tough problems are semantic in nature. See chapter 6.)

The grammatical rules that guide the parser incorporate, in effect, two different grammars. One is a phrase structure grammar for the source language and the other is (akin to) a transformational grammar for the target language. The use of a phrase structure grammar for the source language has two consequences: the construction of parse trees is fast and simple but the system accepts ill-formed source language sentences. The grammar applied to English sentences 'covers' English in the sense that it accepts all English sentences (ideally, not in its present form) but not only English sentences.

Since my goal is to design a system to translate from English into French, I have assumed that input sentences are usually correct. The system, accordingly, does not interpret well-formedness as a necessary condition for acceptability and discovers only gross mistakes. For instance,

65) Boy girl.

would be rejected but

66) Boy loves girl.

is accepted.

The assumption that inputs are usually correct does not only make the original parse much more efficient but it also makes it possible to process ill-formed sentences that most people would consider perfectly intelligible.

Newspaper headlines, messages in telegram style, many utterances by children, and a surprisingly large percentage of 'ordinary' sentences are all ill-formed in the strict linguistic sense. An intelligent system should be able to interpret even seriously ill-formed sentences and should not be stumped by a sentence like

67) There was two boys in the car.

Such a system should also 'degrade gracefully': it is better to get strictly ungrammatical output for ungrammatical input than no output at all. However, had my purposes been different, then I would have chosen a stronger grammar for the first parse. For instance, a CAI system designed to teach English would need an English grammar that is powerful enough to guard against all sorts of mistakes. The present system could, of course, be equipped with such a grammar but only at the price of considerably slowing down its operation.

CHAPTER SIX

THE SEMANTIC COMPONENT

A. SURFACE SEMANTICS

An important issue in semantics concerns the extent to which natural language processing has to rely on 'world knowledge'. Although I do not intend to discuss this very complex issue at length, a few remarks are in order. Proponents of 'deep', frame-based meaning representations typically do not attempt to distinguish factual and linguistic knowledge. They treat both kinds of knowledge uniformly. (A well known example of this policy is the knowledge representation language KRL (Bobrow, Winograd 1977).)

In chapter 3, it was argued that uniform deep representations are inefficient and perhaps psychologically inadequate and therefore high level representations should be used whenever possible. It was also pointed out that many tasks, like deductive inference, that are commonly thought to require deep representations can often be better accomplished with 'shallow' representations i.e. with representations near or at the level of the lexemes themselves. The refusal to distinguish factual and linguistic knowledge also leads proponents of frame-based representations to include knowledge in an understanding system that seems far too strong. Winograd, for example, has argued that a powerful understanding system should know enough to find out that 'they' refers to 'the City Council' in 1) and to 'the women' in 2).

- 1) The City Council refused the women a permit because they feared violence.
- 2) The City Council refused the women a permit because they were communists.

But, as Wilks notes, whatever determines the antecedents of 'they' in 1) and 2) has nothing to do with natural language understanding: "...it all

depends on whether the City Council is in Washington or Peking, so that an intelligent system might be perfectly right to refuse to assign the anaphora in such trick examples..." (Wilks 73, p.117).

It is admittedly difficult -- and if philosophers like Quine are right (Quine 60), impossible -- to distinguish sharply between linguistic and factual knowledge. It may also be the case that some knowledge that is commonly thought to be factual is needed for tasks that are commonly thought to be linguistic. But neither of these two facts justifies the currently fashionable and thorough integration of linguistic and factual knowledge. Moreover, in order to enhance the transportability of natural language processing systems, it seems clearly desirable to keep their linguistic expertise as separate as possible from the knowledge modules needed for question answering and problem solving in specific domains.

A tentative implementation of surface semantics will be illustrated below by two applications: word sense disambiguation and deductive information retrieval. It should be kept in mind however, that the present system is in no way committed to the use of a surface semantics. The system can handle frame-like structures, and the available graphprocessing routines are general enough to build and manipulate semantic networks of arbitrary complexity.

The system is simply designed to operate 'superficially' as long as possible, for the previously mentioned reasons of efficiency, modularity and transportability.

In its current stage of development, the semantics can resolve only the very simplest kinds of ambiguities. Before giving some examples of ambiguities that the semantics can handle and some that it could be made to handle without much additional effort, a few things should be said about

the theoretical status of the concept of ambiguity.

As far as I know; no acceptably clear definition of the concept of semantic ambiguity is available. (I consider a definition in a theory of language 'acceptably clear' if it is couched in an extensional meta-language and if its definiens does not contain any undefined member of a small family of interdefinable 'intensional notions'. The latter family, 'the idiom of intentionality', includes some concepts that have been used informally throughout this paper: meaning, translation, synonymy, conceptual or analytic truth, necessity, possibility, ambiguity. Given any one of these concepts, the others are readily definable. For example, given an acceptably clear concept of synonymy, it is easy to define 'meaning', 'translation', 'conceptual truth' etc:

The meaning of a sentence S is the set of all sentences synonymous with S; S translates S' if and only if S and S' are interlinguistically synonymous; S is conceptually true if and only if either S is logically true or S becomes a logical truth when extralogical predicates in S are replaced by synonymous predicates; S is ambiguous if and only if S has more than one meaning. The obscurity of these and similar definitions is deplored in (Quine 53, 60) and (White 56).

The theoretical intractability of 'ambiguity' is not at all mitigated by the work of lexicographers. A clear definition provides identity criteria for its definiendum and, in particular, makes it possible to

individuate and count the referents of the latter. Lacking such a definition, it is impossible to decide whether two different paraphrases constitute one or two senses of a given word. The inability to individuate senses is brought out clearly by the astonishing fact that the number of distinct entries for ambiguous words -- and most words are ambiguous -- seems to be roughly proportional to the size of the dictionary! A similar, unsolved problem concerns the distinction between ambiguous, fuzzy and metaphorically used concepts. (For a proposed computational analysis of metaphors, See (Winston 79)).

The general problem of disambiguation is, strictly, unsolvable because several answers may appear equally correct in some cases and because different answers may appear correct when larger and larger contexts are considered. For example, we would assume that 'them' in 3) refers back to 'the women'.

- 3) The soldiers fired at the women, and some of them fell down.

However, we might drop this assumption when 3) is preceded by 4).

- 4) When I fired a shotgun for the first time I fell flat on my back.

To be as good as people are at disambiguation, a program would need potentially all pieces of linguistic and world knowledge, all modes of inference that people use, in short, the full powers of a human mind replete with facts and heuristics. (The implementation of a 'mind' is left as an

exercise for the reader. Hint: revamp epistemology, the psychology of cognition and perception. learning theory, linguistics, neurophysiology, AI, design new ways of representing knowledge etc. and combine all of these ingredients into one system.)

Leaving the hopelessly difficult general problem of disambiguation aside, I shall now consider a very simple type of referential ambiguity.

A simple recency rule is sufficient to find the referents of pronouns in cases exemplified by 5)

5) I ordered a hamburger, sat down in my car, and ate it.

It can be bound to the hamburger rather than the car by the following process: the case structure for eating requires as object an item with the semantic marker FOODSTUFF. The pronoun should therefore be bound to an antecedent that is similarly marked. A relatively straightforward way of achieving this binding correctly in many cases is to maintain a fixed-sized list of prior noun phrases and to check for the most recently entered noun phrase that answers the description required by the pronoun. In 5), car would be skipped because it is not a food item but hamburger qualifies and is chosen as the referent. If the list is about to exceed its chosen maximum size, the first item entered is deleted. Ties between equally acceptable antecedents are broken in favor of the most recently entered item. Unfortunately, it is easy to think of examples for which this method fails to work. By the way, Winograd's SHRDLU seems to use a similar method of referential disambiguation.

B. WORD SENSE DISAMBIGUATION

The word sense disambiguation routines that are now implemented take care of the following kinds of situations.

6) The old man takes off his shoes.

7) The old man takes his shoes off.

6) and 7) are both translated as

6') Le vieux homme enlève ses souliers.

In general, particles like off, up etc. may be separated from their verbs by arbitrarily complex noun phrases:

8) The plane will take off in an hour.

9) He takes the book off the table.

10) He takes off in a car.

11) She takes up a new occupation.

12) He takes his opinion back.

13) She takes a chance with the boy.

14) The trip will take one hour.

15) He picks up a beautiful girl.

16) He picks ideas up in the school.

17) He picks a green apple.

18) He picks up a green apple.

19) I draw a conclusion.

20) I draw a picture.

8) to 20) are translated by 8') to 20') respectively, as follows:

- 8') L'avion s'envolera dans une heure .
- 9') Il enlève le livre de la table.
- 10') Il part dans une voiture.
- 11') Elle prend une nouvelle occupation.
- 12') Il renonce à son opinion.
- 13') Elle prend une chance avec le garçon.
- 14') Le voyage durera une heure.
- 15') Il prend une belle fille.
- 16') Il apprend des idées dans l'école.
- 17') Il choisit une pomme verte.
- 18') Il ramasse une pomme verte.
- 19') Je tire une conclusion.
- 20') Je dessine une image.

The mechanism that achieves this type of disambiguation can be applied to words of all categories and can be repeatedly invoked in any given sentence. The method works as follows.

Some verbs (eventually all of them), have Fillmore-type case structures on their property lists. In addition, ambiguous words of any category have a list of disambiguation rules on their property lists. (LISP creates a property list for each symbolic atom as soon as the latter is read for the first time. A property list consists of a set of dotted pairs. The first element of each such pair is an atomic indicator or property name and the second element is the corresponding property value. The value can be any S-expression. Several system functions are available to manipulate property

lists. For instance, (PUT "AT "INDIC "VAL) puts the value VAL on the property list of the atom AT under the name "INDIC. (GET "AT "INDIC retrieves VAL).

Each disambiguation rule is of the form

((pattern)(tests)(actions)(substitute)).

Some of the rules for take look like this:

- 21) ((*X TAKE OFF *Y) (H* C*) (EFFPART REPLACE) (DOFF))
- 22) (SAME (TR* TI*) (EFFPART SUBST1) (DEPART))
- 23) ((*X TAKE *Y OFF *Z) (H* PO* PO*) (SUBST1 SUBST2) (REMOVE))
- 24). ((*X TAKE *Y) (E* TI*) (SUBST1 EFFPART) (LAST))

When the dictionary entry for take calls for a translation of the word, these rules are submitted to the function DISAMBIG. DISAMBIG attempts to match the pattern, i.e. the first part of a rule, against the input sentence until a match is achieved or all rules are exhausted. If no rule matches, the word is translated by the default word listed in the dictionary.

If a match succeeds, then the matching items, i.e. the parts of the sentence bound to the star-prefixed pattern variables are subjected to tests. There is one test for each pattern variable. For example, the value of the pattern variable *X in 21) is tested by the function H*, and *Y is tested by C*. If all tests are successful, the actions are executed and the substitute is substituted for the ambiguous words. In 21), take off is replaced by doff. The latter's translation, i.e. enlever is now conjugated according to the GEND, UNIV, PERS, TENSE information that is available about take off. If at least one test fails, the next

rule is tried. If the pattern of the next rule is an atom rather than a list, as in 22), the same pattern is assumed present. In this way, different batteries of tests can be applied to one and the same pattern without the need for rematching.

The tests are functions that might check for the presence of certain semantic markers on the main nouns of noun phrases or for the semantic type of a case on the case structure of a verb. Thus, H* and PO* in 23) search a list of words for a main noun and, if they find one, check whether it has the marker HUMAN or PHYSICALOBJECT.

The actions of a disambiguation rule erase particles or verbs, insert particles needed for the French output and replace some words by others.

The semantic markers used in the present implementation are chosen merely on the basis of their potential contribution to the task of disambiguation. They are not meant to bear much theoretical weight. More specifically, they are not put forward as candidates for the status of primitive universals of human thought. Neither are they chosen so as to constitute a minimal and complete set.

Many words have several markers, for example, a hammer is both a PHYSICALOBJECT and a TOOL, and a woman is both HUMAN and ANIMATE. The hierarchical relationships between non-disjoint markers could be easily expressed by rules that allow inferences from, say, TOOL to PHYSICALOBJECT or from HUMAN to ANIMATE. For the time being, I have chosen to express hierarchical relationships implicitly through an ordering, according to decreasing marker specificity, of the disambiguation rules themselves. Each rule can thus be seen as attempting to find the most specific

communicative pattern such as (HUMAN use TOOL), (PROCESS take TIME), that might be embedded in a given sentence.

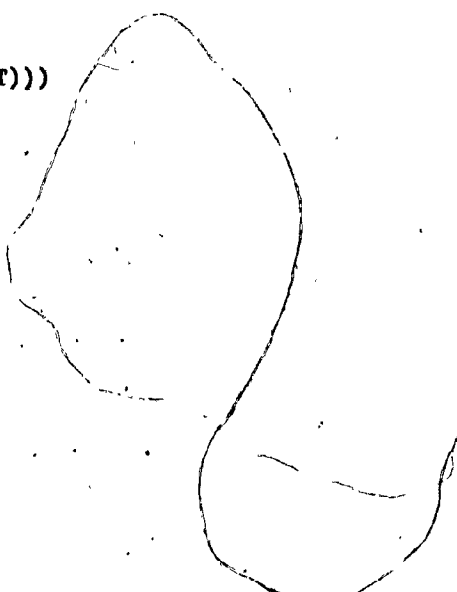
The extraction of communicative patterns is the task of the symbolic pattern matcher. This algorithm compares a sentence with a rule pattern. A pattern is a list consisting of an arbitrary number of atomic key words and variables. The latter are atoms prefixed with asterisks. For a match to succeed, a sentence must contain key words in the same order as the pattern, and the end of the sentence and the pattern must be reached simultaneously. Each pattern variable can match zero or more words in the sentence and, if the match succeeds, gets as its value the list of matching words.

Thus, if the sentence and the pattern are exhausted simultaneously, the algorithm MATCH returns T, and if only one of them is empty it returns NIL. If the CAR of the pattern is a variable, MATCH calls itself recursively to find out which of the following cases obtains: the variable matches nothing in which case MATCH works on the rest of the pattern; the variable matches one atom in the sentence in which case MATCH works on the rest of the sentence and of the pattern; the variable matches several atoms in which case MATCH works recursively on the rest of the sentence. Otherwise, that is, if the CAR of the pattern is not a variable, then MATCH checks whether the CAR of the pattern equals the CAR of the sentence. When this last test succeeds, MATCH works recursively on the rest of the sentence and of the pattern, otherwise it returns NIL. (In the algorithm below, the function (\$NOC LIS S) CONSES the expression S onto the list LIS.)

```

(MATCH (SENT PAT)
(COND ((NULL SENT)(NULL PAT))
  ((NULL PAT) NIL)
  ((VARIABLE (CAR PAT))
    (COND ((NULL (CDR PAT))
      (SET (CAR PAT)
        (APPEND (EVAL (CAR PAT)) SENT)))
      ((EQ (CAR SENT)(CADR PAT))
        (MATCH (CDR SENT)(CDDR PAT)))
      (T (SET (CAR PAT)
        (SNOC (EVAL (CAR PAT))
          (CAR SENT)))
        (MATCH (CDR SENT) PAT))))))
  ((EQ (CAR SENT)(CAR PAT))
    (MATCH (CDR SENT)(CDR PAT)))
  (T NIL) ))

```



C. SEMANTIC NETWORKS AND DEDUCTIVE RETRIEVAL

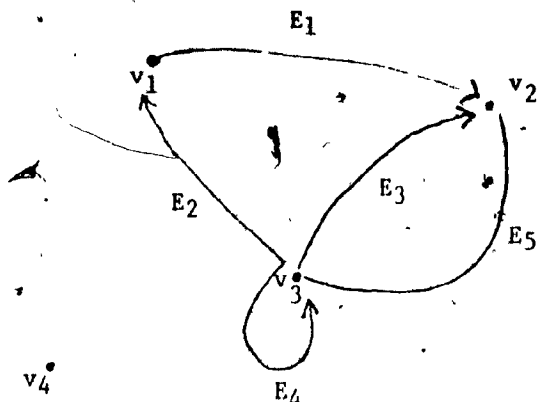
In this section, I shall describe a set of routines to construct and manipulate arbitrarily complex semantic networks and to retrieve information deductively from such networks. In accordance with the 'alternative view' outlined in Chapter 3, the networks are high level case structures in which case relations are attached to the lexemes themselves. However, the same set of network functions can be used to construct low level, conceptual meaning representations. (The design and implementation of a conceptual meaning representation that is both subtle enough to

capture slight meaning nuances and deductively at least as powerful as predicate logic is a major undertaking for future research. Some very sophisticated networks are described in (Brachman 77, 79), (Hendrix 75), (Schubert, Goebel, Cercone 79).)

LISP's property list facility makes it easy to implement graphs and thus, semantic networks. A semantic network is a directed graph, that is, an ordered pair $G = \langle V, E \rangle$, where V is a set of vertices and E is a binary relation on V . It is convenient to admit multiple, differently labelled edges and to allow nodes to participate in several graphs, that is, to treat only the edges as local to a given graph. (Pratt, Friedman 71) have described a graph processing programming language extension GRASPE which is used in present system. GRASPE is a complete system of abstract, semantic definitions of primitive operations on digraphs. Before describing the GRASPE definitions and their LISP counterparts, I shall briefly explain how graphs are represented by property lists.

Nodes are represented by atoms. If a node v occurs in graph G , then the property list of v contains, under the indicator G , a dotted pair of two lists: (INPAIRS . OUTPAIRS). Each element of INPAIRS is, in turn, a dotted pair consisting of the label of the node at which the edge begins that terminates at v and of the label of that edge. Analogously, each element of OUTPAIRS consists of a node label and an edge label. The node label refers to the destination node that can be reached from v along the labelled edge. A graph G is represented by an atom whose property list contains, under the indicator NS (Node set), a list of all nodes of G . For example, the graph G can be represented by the diagram below.

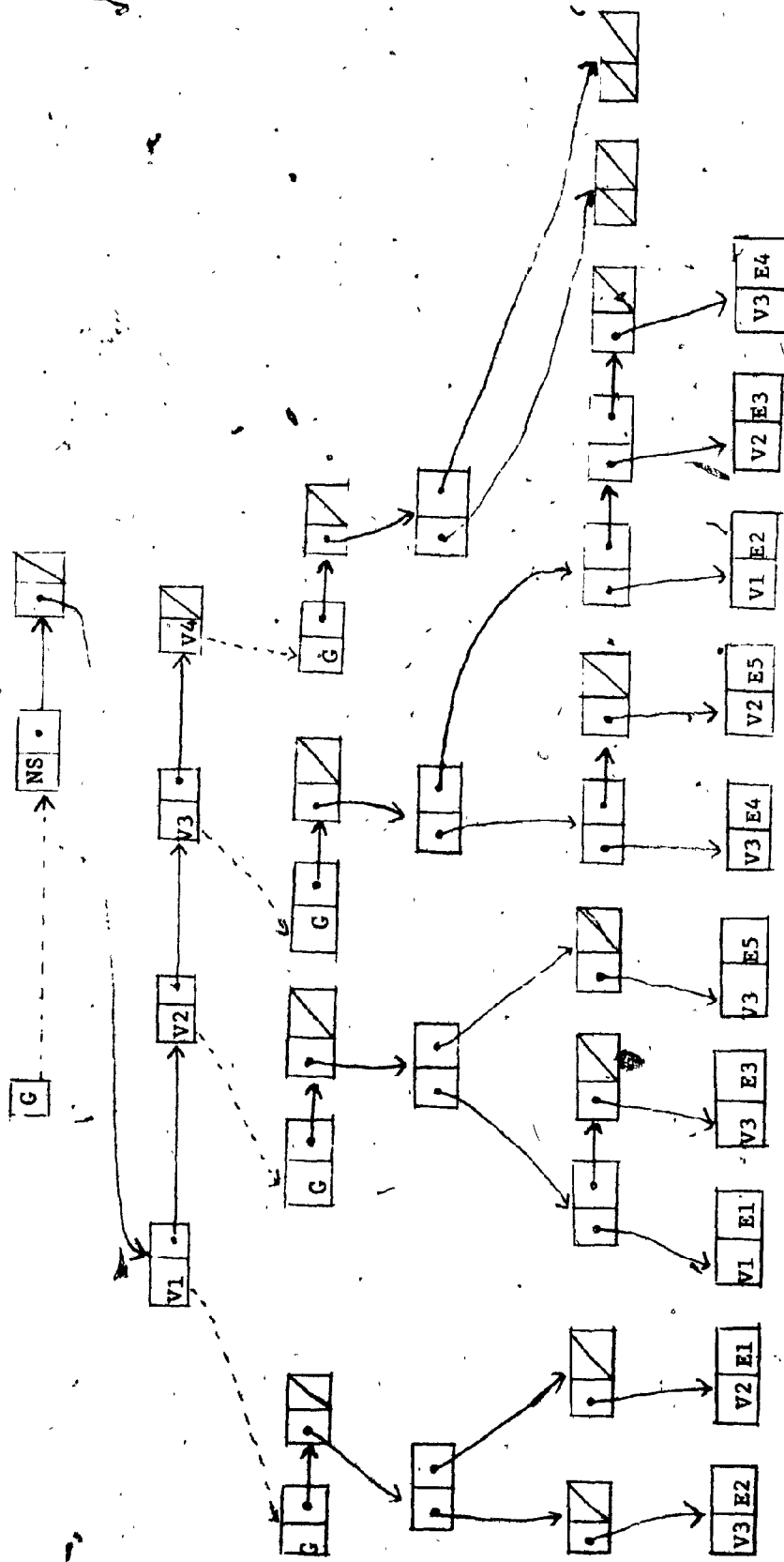
G:



To reduce clutter, several simplifying conventions are made in the diagram: most of the structure of atom and list cells is suppressed; a pointer to an atom is represented by writing the name of the atom into the pointer field; the property list of an atom is drawn with a dashed arrow '----->'.

(In reality, the property list of an atom is pointed at by the CSR field of the atom header, and each atom has at least two properties which are omitted here: PNAME which contains the atom's print image and the INFO property whose value controls the printing of special literal atoms.)

Let DG, N, E be the sets of digraphs, nodes and edges, respectively. Let $g \in DG$ and let N_g and E_g be the nodes and edges of g respectively. Let $n, m \in N_g, e \in E_g$ and $\text{arc}: DG \rightarrow 2^{N_g \times E_g \times N_g}$. When $(n, e, m) \in \text{arc}(g)$, we shall say that there is an edge e from n to m in g .



The first five functions below create graphs, the next nine access them, and the last eight delete components and destroy graphs.

Creators

makeemptygraph (g) = df $N_g \longleftarrow E_g \longleftarrow \emptyset$

(MAKEEMPTYGRAPH (G))

(OR(NODESET G) (PUT G "NS NIL])

(NODESET(G) (GET G "NS))

makeonenode (n,g) = df T, with side effect : $N_g \longrightarrow N_g \cup \{n\}$

(MAKEONENODE (NG))

(OR(GET N G) (AND (PUT N G "(NIL.NIL))

(PUT G "NS (UNION(LIST N) (NODESET G])

(UNION (SET1 SET2)

(COND((NULL SET1)SET2)

((MEMBER(CAR SET1)SET2)(UNION(CDR SET1)SET2))

(T(CONS(CAR SET1) (UNION(CDR SET1)SET2])

makeoutpair (n,e,m,g) = df T, with side effect: $N_g \longrightarrow N_g \cup \{n,m\}$

$E_g \longleftarrow E_g \cup \{(n,e,m)\}$

(MAKEOUTPAIR(N E M G))

(AND(MAKEONENODE N G) (MAKEONENODE M G))

(PUT N G (CONS(INPAIRS N G) (UNION (OUTPAIRS N G)

(LIST(CONS M E)))))

(PUT M G (CONS(UNION(INPAIRS M G) (LIST(CONS N E)))

(OUTPAIRS M G])

makeinpair (n,e,m,g) = df makeoutpair (m,e,n,g)

makeadjacentpair (n,e,m,g) = df makeoutpair (n,e,m,g) and makeinpair(n,e,m,g).

Selectors

outpairs (n,g) = df $\{(m,e)/(n,e,m) \in \text{arc}(g)\}$

(OUTPAIRS(N G) (CDR(GET N G)])

inpairs (n,g) = df $\{(m,e)/(m,e,n) \in \text{arc}(g)\}$

(INPAIRS(N G) (CAR(GET N G)])

outarcs (n,g) = df $\{e/V_m ((n,e,m) \in \text{arc}(g))\}$

(OUTARCS (N G)

(MAKEASET(MAPCAR(OUTPAIRS N G) "CDR])

inarcs (n,g) = df $\{e/V_m ((m,e,n) \in \text{arc}(g))\}$

(INARCS (N G)

(MAKEASET(MAPCAR(INPAIRS N G) "CDR])

(MAKEASET(SET1)

(MAPPENDLIS SET1 (FUNCTION(LAMBDA (SETCDR)

(COND((MEMBER(CAR SETCDR)(CDR SETCDR))())

(T(LIST(CAR SETCDR)])

(MAPPENDLIS(LIS FUN)

(COND((NULL LIS)())

(T(APPEND(FUN LIS) (MAPPENDLIS(CDR LIS)FUN)])

outnodes (n,g) = df $\{m/V_e ((n,e,m) \in \text{arc}(g))\}$

(OUTNODES(N G)

(MAKEASET(MAPCAR(OUTPAIRS N G) "CAR)])

innodes (n,g) = df $\{m/V_e ((m,e,n) \in \text{arc}(g))\}$

(INNODES (N G)

(MAKEASET(MAPCAR(INPAIRS N G) "CAR)])

adjacentpairs (n,g) = df outpairs (n,g) v inpairs (n,g)

adjacentarcs (n,g) = df outarcs (n,g) v inarcs (n,g)

adjacentnodes (n,g) = df outnodes (n,g) v innodes (n,g)

Destroyers

deloutpair (n,e,m,g) = df T, with side effect: $E_g \leftarrow E_g - \{(n,e,m)\}$

DELOUTPAIR (N E M G)

(AND(PUT N G (CONS(INPAIRS N G)

(COMPLEMENT(OUTPAIRS N G) (LIST(CONS M E))))))

(PUT M G (CONS(COMPLEMENT(INPAIRS M G) (LIST(CONS N E))))

(OUTPAIRS M G))]

delinpair (n,e,m,g) = dg T, with side effect: $E_g \leftarrow E_g - \{(m,e,n)\}$

(DELINPAIR(N E M G)(DELOUTPAIR M E N G)]

deloutarcs (n,g) = df T, with side effects: $E_g \leftarrow E_g - \{(n,e,m)/Ve,m$
 $((n,e,m) \in \text{arc}(g))\}$

DELOUTARCS (N G)

(NOT(MAPC(OUTPAIRS N G) (FUNCTION (LAMBDA(P)

(DELOUTPAIR N(CDR P)(CAR P)G))

delinarcs (n,g) = df T, with side effect: $E_g \leftarrow E_g - \{(m,e,n)/Ve,m$
 $((m,e,n) \in \text{arc}(g))\}$

(DELINARCS (N G)

(NOT(MAPC(INPAIRS N G) (FUNCTION(LAMBDA(P)

(DELINPAIR N(CDR P) (CAR P) G))

deladjacentpair (n,e,m,g) = df deloutpair (n,e,m,g) and delinpair (n,e,m,g)

deladjacentarc (n,g) = df deloutarcs (n,g) and delinarcs (n,g)

delsinglenode (n,g) = df if adjacentarcs (n,g) = \emptyset then $N_g \leftarrow N_g - \{n\}$

(DELSINGLENODE (N G)

(OR(OUTPAIRS N G) (INPAIRS N G)

(AND(PUT N G NIL) (PUT G 'NS(COMPLEMENT

(NODESET G) (LIST N))]

delgraph (g) =df if nodeset (g) = \emptyset then g is erased and T is returned.

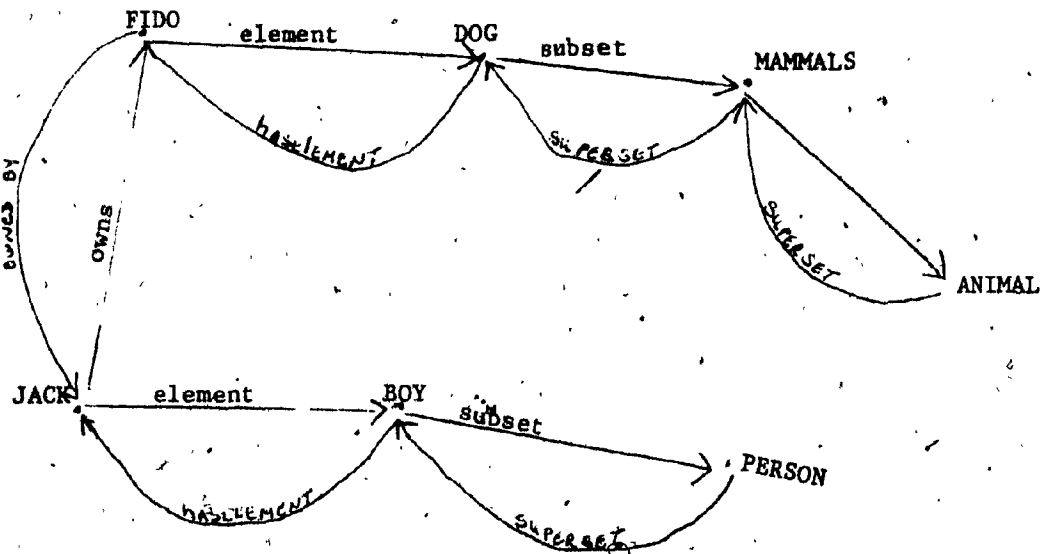
(DELGRAPH(G) (OR(NODESET G) (REMPROP G "NS) T))

The above GRASPE functions have much more power than is required for the rather trivial tasks to which they are put in the present system. They help, for example, construct high level networks in which nouns and other verb complements are represented by nodes and two-place verbs by edges. Such networks can then be processed by a simple question answering algorithm QA. The latter has been implemented to illustrate the feasibility -- and, as urged in chapter 3, the desirability -- of attempting logical deductions in terms of high level predicates before working on their fine-grained conceptual representations. Since the current implementation contains only shallow networks, its deductive power is quite limited. (When QA does not succeed with a deduction, it simply gives up. It is hoped that in future implementations such logical failures of nerve can be converted into calls to stronger deduction algorithms that are capable of working with more detailed networks.)

QA is modelled after Raphael's system for semantic information retrieval SIR (Raphael 68) and works as follows: whenever a sentence is input, a corresponding network is built. If new sentences share network-represented words with old sentences, the appropriate networks are automatically extended to represent the fresh information. In this manner, data bases can be built automatically by simply typing in English sentences.

For example, after sentences 1) - 6) have been read, the network 7) exists in memory.

- 1) Fido is a dog.
- 2) All dogs are mammals.
- 3) A mammal is an animal.
- 4) Jack is a boy.
- 5) Jack owns Fido.
- 6) Each boy is a person.



The questions 8) - 16) are answered by 8') - 16').

- 8) Is Fido a dog?
 8') Yes.
- 9) Is Fido a mammal?
 9') Yes.
- 10) Is Fido an animal?
 10') Yes.
- 11) Is Jack a person?
 11') Yes.
- 12) Is every mammal a dog?
 12') Not enough information.
- 13) Are mammals dogs?
 13') Sometimes.
- 14) Does a person own a dog?
 14') Sometimes.
- 15) Does Fido own Fido?
 15') No, they are identical.
- 16) Is each person an animal?
 16') Not enough information.

If we now type in

- 17) Each person is an animal.

two new links will be set up in 7) and 16) is answered affirmatively.

The basic idea behind QA is this: QA consists of a number of link

specialists'. Each link specialist is a graph searching routine that embodies procedural knowledge about the logically relevant properties of the predicate (s) that is (are) represented by the link.

For instance, the question whether A is a subset of B triggers a procedure that tries to reach B from A along zero or more subset links. If it fails, it tries to reach A from B along one or more subset links. Success in the latter situation gives rise to the answer 'Sometimes'. The two tests for this question are expressed in LISP like this:

(PATH A (SUBSET KS) B)

(PATH B (SUBSET KP) A).

The question whether B owns an element of A triggers a procedure that tries to reach A from B either along zero or more equivalence-links followed by an owns-link, followed by zero or more subset-links, or along zero or more equivalence-links, followed by an element-link, followed by zero or more subset links, followed by an ownedby-link, followed by zero or more subset-links. The two tests are written like this:

(PATH B (EQUIV KS OWNS SUBSET KS) A)

(PATH B (EQUIV KS ELEMENT SUBSET KS OWNEDBY SUBSET KS) A).

(KS in the path description functions like the Kleene star i.e. it means that the preceding link is to be traversed zero or more times. KP means 'one or more times'.)

The true work horse of QA is the function PATH which is defined as a FEXPR. (A FEXPR function takes an arbitrary number of arguments that are not evaluated before the function is evaluated. Instead, its only formal variable is bound to the list of the unevaluated arguments.) PATH

returns an appropriate answer according to whether or not the destination node can be reached from the source node via the list of links that is its middle argument. Since PATH is a rather lengthy function, I refrain from reproducing it here.

CHAPTER SEVEN

FUTURE DIRECTIONS AND CONCLUSION

Needless to say, a great deal remains to be done: the vocabulary is tiny, the grammar is fairly small, contextual and disambiguation mechanisms are rudimentary, and the deductive methods are weak. But the most serious shortcoming is the inadequate network formalism.

The design and implementation of a semantic network language is a major task which I hope to tackle in another paper. Here I shall only outline a few properties that a more adequate network formalism should have.

- 1) Domain-specific relations like OWNS, HAS etc. should be represented by nodes and not by links. This would make it possible to attach information and not just labels to domain-specific relations. The only links left in the system, the "system-links", would be used to impose structure on concept-nodes, i.e. to express hierarchical relationships, logical connectives and modalities, quantifiers and the scopes of quantified variables, and groupings of nodes to represent contexts.
- 2) Given a fixed set of system-links, it should be possible to provide a formal semantics of semantic nets in terms of an interpreter that processes these links.
- 3) The network should be capable to express the distinction between definitions and facts and the internal structure of concepts.

- 4) The network should have the expressive power of predicate logic with modal operators and identity; in particular, it should have perspicuous ways of indicating the scopes of nested quantifiers.
- 5) The available system-links should make it possible to solve the "symbol-mapping problem" (McDermott 75), (Fahlman 77), i.e. the problem of inheritance of properties.
- 6) Finally, the formalism should facilitate the use of inference rules and should be rich enough to represent natural language meanings at several levels.

Recent work on semantic nets (See (Woods 75), (Boley 77), (Brachman 77,79), (Fahlman 77), (Hendrix 79), (Schank, Abelson 77), (Shapiro 79), (Schubert, Goebel, Cercone 79), (Wilks 79), among others) has not yet converged on a formalism that would meet the above desiderata. But one thing seems clear from previous discussions (particularly in chapter 3): an adequate network formalism must make it possible to represent information at different levels of detail. Consequently, the concept of a directed graph must be extended to include structures in which single nodes can represent and provide entry points to entire subgraphs. Such structures might be defined along the lines indicated below. G below is similar to the graphs that (Pratt 69) used to formalize the semantics of programming languages. It will be remembered that the graph processing routines described in the previous chapter view a digraph $G = \langle N, A, E \rangle$ as consisting of the nonempty, disjoint sets N and A of

nodes and arc labels, respectively, and of edges $E \subseteq NXAXN$. Let $H = \langle N, A, E, n \rangle$, $n \in N$, be a digraph with entry point n (DGE). Then it is possible to define a new kind of digraph G' in which each node can represent either an atom or a DGE:

$$G' = \langle AT, N, A, VAL \rangle \text{ where } AT \text{ is a nonempty set of atoms, } AT \cap N = AT \cap A = \emptyset, VAL: N \rightarrow AT \cup DGE, \text{ and}$$

$$DGE = \left\{ H \mid H = \langle N', A', E', n' \rangle \wedge N' \subseteq N \wedge A' \subseteq A \right\}.$$

Let $G' = \langle AT, N, A, VAL \rangle$ be a digraph each of whose nodes has been assigned an atom or a digraph by the function VAL . Let $S \subseteq N$, $n \in N$.

Then the function $f: 2^N \rightarrow 2^N$ is defined as follows:

$$f(S) = \begin{cases} \emptyset & \text{if } S = \emptyset \text{ or } (S = \{n\} \text{ and } VAL(n) \in AT); \\ N' & \text{if } S = \{n\} \text{ and } VAL(n) = \langle N', A', E', n' \rangle; \\ \bigcup \{f(\{n\}) \mid n \in S\} & \text{if } |S| \geq 2. \end{cases}$$

$f^k: N \rightarrow 2^N$, $k \geq 1$, is defined inductively, thus:

$$f^1(n) = f(\{n\}); \quad f^{k+1}(n) = f(f^k(n)).$$

In terms of f^k it is now easy to express the formal conditions that a graph must meet in order to allow the representation of information at various levels of detail. Such a "layered" graph, or L-graph, is a graph $G' = \langle AT, N, A, VAL \rangle$ that meets the following two conditions:

- (i) $\forall n (n \in N \wedge \bigcup \{f^k(n) \mid k \geq 1\} = N - \{n\})$ and
- (ii) $\bigwedge n (n \in N \rightarrow \forall k (f^k(n) \neq \emptyset))$.

Conditions (i) and (ii) together imply that each L-graph has exactly one entry point.)

Although many things remain to be done to create a flexible natural language processing system with a semblance of comprehension, this paper has achieved the goals set at the outset:

A general context-free parser that is more efficient than ATN parsers has been developed. The generated parse tree, whose construction is influenced by restricted context-sensitive tests, makes it possible to apply transformational and semantic rules without concern for control issues.

The parser provides the control structure of a modular programming environment for experimenting with different representational and inferential strategies. The resulting system can also be looked at as an interpreter for a programming language whose programs consist of largely independently writeable dictionary entries, context-free and transformational rules, semantic rules, and deductive retrieval rules.

The system constitutes a theoretical alternative to both Chomskyan linguistics and the main trends in computational linguistics. But it also serves as a tool for writing natural language front ends quickly and cheaply.

As an illustration, a small English to French translator has been written. The translator itself is extremely small: while its syntactic and semantic coverage is greater than that of some other systems, it uses fewer than 100 rules.

The system includes algorithms to extract basic communicative patterns from sentences. These patterns are used in the process of word sense disambiguation.

The system uses a complete set of graph-processing routines to create network data bases from natural language input.

The resulting semantic networks are processed by a deductive question - answering algorithm.

It is possible to bypass detailed parsing and linguistic processing and to do key-word-based parsing instead.

REFERENCES

- A.V. Aho, J.D. Ullman, The Theory of Parsing, Translation and Compiling, Vol. I: Parsing. Prentice Hall, Englewood Cliffs, N.J., 1972.
- J.R. Anderson, G.H. Bower, Human Associative Memory, Winston, Washington D.C., 1975.
- R.C. Anderson, Encoding processes in the storage and retrieval of sentences. *J. Exp. Psych.*, 2, 1971, 338-340
- M. Bates, The theory and practice of augmented transition network grammars. In (Bolc 78), 1978, 191-259.
- D.G. Bobrow, J.B. Fraser, An augmented state transition network analysis procedure. *Proc. IJCAI*, 1969, 557-567.
- D. Bobrow, R. Kaplan, M. Kay, D. Norman, H. Thompson, T. Winograd, GUS, a frame-driven dialogue system. *AI*, Vol.8, 2, 1977.
- D. Bobrow, T. Winograd, An overview of KRL, a knowledge representation language. *Cogn. Sci.* Vol. 1, 1. 1977, 3-46
- M. Boden, Artificial Intelligence and Natural Man. Harvester Press, Haddocks, England, 1977.
- L. Bolc, ed. Natural Language Communication with Computers. Springer, N.J., 1978.
- H. Boley, Directed recursive labelnode hypergraphs: a new representation language. *AI*, vol. 9, 1977, 49-85.
- R. Brachman, What's in a concept: structural foundations for semantic networks. *Int. J. for Man-Machine Studies*, Vol. 9, 1977, 127-152.
- R. Brachman, On the epistemological status of semantic networks. In (Findler 79), 1979, 3-50.
- W.F. Brewer, Memory for ideas: synonym substitution. *Memory and Cognition*, Vol. 3, 1975, 458-464.
- B. Bruce, Case Systems for Natural Language. AI Report 23, Bolt Beranek and Newman Inc., 1975.
- C. Bullwinkle, The semantic component of PAL: The personal assistant language understanding program. MIT AI Lab Working Paper 141, 1977.
- M. Celce, Paradigms for sentence recognition. SDC Final Report HRT-15092|7907, 1972.

- N.J. Cercone, L.K. Schubert, Toward a state-based conceptual representation.
Adv. papers of the 4. IJCAI, 1975, 83-90.
- E. Charniak, Towards a model of children's story Comprehension.
AITR-266 MIT AI Lab, 1972.
- E. Charniak, Organization and Inference in a frame-like system of Common sense knowledge. In (Schank, Nash-Webber 75).
- E. Charniak, Inference and Knowledge in language Comprehension.
AI, Vol. 2, 1976, 541 ff.
- E. Charniak, A framed PAINTING: The representation of a Common sense knowledge fragment. Cogn. Sci., Vol. 1, 1977, 355-394.
- E. Charniak, Y. Wilks, ed., Computational Semantics. North-Holland, Amsterdam 1976.
- E. Charniak, C.K. Riesbeck, D.V. McDermott, Artificial Intelligence Programming. Lawrence Erlbaum Ass., Hillsdale, N.J., 1980.
- M. Chevalier, J. Dansereau, G. Poulin, TAUM-METEQ. Univ. de Montreal, 1978
- N. Chomsky, Aspects of the Theory of Syntax. MIT Press, Cambridge, Mass., 1965.
- N. Chomsky, H. Lasnik, Filters and control.
Linguistic Inquiry, vol. 1, 1977, 425 - 504.
- D. Davidson, The logical form of action sentences.
In N. Rescher, ed, The Logic of action and preference. Pittsburgh Univ. Press, 1967.
- R. Davis, B. Buchanan, E. Shortliffe, Production rules as a representation for a knowledge - based consultation program.
AI, vol.8, 1977.
- J. Earley, An efficient Context-free parsing algorithm.
CACM, vol.13., 1970, 94 - 102.
- S.E. Fahlman, A system for representing and using real-world knowledge.
AI Lab, MIT, Cambridge, Mass., 1977.
- C. Fillmore, The Case for case. In E. Bach, ed. Universals in linguistic theory. Holt, Rinehart and Winston, N.Y., 1968.
- C. Fillmore, Types of lexical information. In Steinberg, Jakobovits, ed., Semantics: An interdisciplinary reader. Cambridge Univ. Press, 1971.

- C. Fillmore, Scenes - and - frames semantics. In (Zampolli 77), 1977.
- N.V. Findler, ed., Associative Networks. Academic Press, N.Y., 1979.
- N.M. Goldman, Computer generation of natural language from a deep conceptual base. Memoranda from the Instit. Stud. Gen. Cogn., Castagnola, 1974.
- B. Grosz, The representation and use of focus in dialogue understanding. Techn. Note 1S1, SRI, Menlo Park, Cal., 1977.
- M.A.K. Halliday, Functional diversity in language as seen from a consideration of modality and mood in English. Foundations of Language, Vol.6, 1970.
- B. Hayes-Roth, F. Hayes-Roth, The prominence of lexical information in memory representations of meanings. J. Verbal Learning and Verbal behavior, vol. 16, 1977, 119 - 136.
- B. Hayes-Roth, Implications of human pattern processing for the design of artificial knowledge systems. In (Waterman, Hayes-Roth 78), 1978.
- D. Hays, Dependency theory: a formalism and some observations. Language, vol.40, 1964.
- G.G. Hendrix, Expanding the utility of semantic networks through partitioning. Proc. 4. IJCAI, 1975, 115 - 121.
- G.G. Hendrix, Encoding knowledge in partitioned networks. In (Findler 79), 1979, 51 - 92.
- R.M. Kaplan, Augmented transition networks as psychological models of sentence comprehension. AI, vol.3, 1972, 77 - 100.
- J. Katz, Semantic Theory. Harper and Row, 1972.
- D.T. Langendoen, Essentials of English Grammar. Holt, Rinehart and Winston, N.Y., 1970.
- W.G. Lehnert, The Process of Question Answering. Lawrence Erlbaum Ass., Hillsdale, N.Y., 1978.
- D. McDermott, Symbol - mapping : a technical problem in PLANNER-like systems. ACM SIGART, Newsletter 51, vol.4, 1975.
- D. McDermott, Artificial intelligence meets natural stupidity. ACM SIGART, Newsletter 57, 1976, 4 - 9.
- M. Minsky, ed., Semantic Information Processing. MIT Press, Cambridge, Mass., 1968.

- M. Minsky, A framework for representing knowledge.
In P. Winston, ed., The psychology of computer vision,
McGraw-Hill, N.Y., 1975, 211 - 277.
- B. Nash-Webber, R. Reiter, Anaphora and logical form ; on formal meaning
representations for natural language, Proc. 5
IJCAI, 1975, 121 - 131.
- D.L.F. Nilsen, The Instrumental Case in English. Mouton, The Hague, 1973.
- D.A. Norman, D.E. Rumelhart, eds., Explorations in Cognition.
Freeman, San Francisco, 1973.
- F. Oppacher, Algorithmic versus procedural semantics.
In T.W. Simon, M. Ringle, eds., Language, mind, and brain.
Lawrence Erlbaum Ass., Hillsdale, N.J., 1981.
- S.R. Petrick, On natural language based computer systems.
In (Zampolli 77), 1977, 325 - 342.
- W.J. Plath, REQUEST: A natural language question - answering system.
IBM J. Res. Develop. XX, 4, 1976, 326.
- T. Pratt, A hierarchical graph model of the semantics of programs.
Proc. AFIPS Spring J. Conf. 1969, Vol. 34, 1969, 813-825.
- T.W. Pratt, D.P. Friedman, A language for graph processing and its formal
semantics. CACM, Vol. 14, 7, 1971, 460-467.
- W.R. Pratt, A linguistics oriented programming language.
Proc. 3 IJCAI, 1973, 372-381.
- V.R. Pratt, LINGOL - A progress report.
Proc. IJCAI, 1975, 422-428.
- W.V.O. Quine, From a Logical Point of View. Harvard Univ. Press,
Cambridge, Mass., 1953.
- W.V.O. Quine, Word and Object. MIT Press, Cambridge, Mass., 1960.
- B. Raphael, SIR: semantic information retrieval.
In (Minsky 68), 1968, 33-134.
- C.J. Rieger, Conceptual memory. In (Schank 75), 1975.
- C.J. Rieger, Five aspects of a full-scale story comprehension model.
In (Findler 79), 1979, 425-462.
- C.K. Riesbeck, Computational understanding: analysis of sentences and
context. Memoranda from the Inst. Stud. Sem. Cogn.
Castagnola, 1974.

- R.B. Roberts, I.P. Goldstein, The FRL manual. MIT AI Lab memo 409, 1977.
- R. Schank, Identification of conceptualizations underlying natural language. In (Schank, Colby 73), 1973.
- R. Schank, Causality and reasoning. Memoranda of the Inst. Stud. Sem. Cogn., Castagnola, 1974.
- R. Schank, ed., Conceptual Information Processing. North-Holland, Amsterdam, 1975
- R. Schank, Representation and understanding of text. AI, 1976, 575-619.
- R. Schank, K. Colby, eds., Computer Models of Thought and Language. Freeman, San Francisco, 1973.
- R. Schank, B. Nash-Webber, eds., Theoretical Issues in Natural Language Processing. Assoc. for Computational Linguistics, 1975.
- R. Schank, R.P. Abelson. Scripts, Goals, Plans and Understanding. Lawrence Erlbaum Ass, Hillsdale, N.J., 1977.
- R. Schank, N. Goldman, C. Rieger, C. Riesbeck, Inference and paraphrase by computer. JACM, Vol. 22, 1975, 309-328.
- L.K. Schubert, Extending the expressive power of semantic networks. AI, Vol. 1, 1976, 163-198.
- L.K. Schubert, R.G. Goebel, N.J. Cercone, The structure and organization of a semantic net for comprehension and inference. In (Findler 79), 1979, 141-175.
- S.C. Shapiro, The SNePS semantic network processing system. In (Findler 79), 1979, 199-204.
- C. Sidner, Disambiguating references and interpreting sentence purpose in discourse. In (Winston, Brown 79), 1979, 233-254.
- R.F. Simmons, Semantic networks: Their computation and use for understanding English sentences. In (Schank, Colby 73), 1973, 63-113.
- R.F. Simmons, Rule-based computations on English. In (Waterman, Hayes - Roth 78), 1978, 435-468
- R.F. Simmons, J. Slocum, Generating English discourse from semantic nets. CACM 1972.
- R.F. Simmons, D. Chester, Inferences in quantified semantic networks. Proc. 5 JFCAI, 1977, 267-273.