Natural Language and Representation of Knowledge

by

William A. Martin

CONTENTS

FIGURES

1. Preface and Introduction

1.1 Disclaimer

This book was completed after Bill Martin's death by a number of his friends. The material for this book was taken by a series of class notes that Bill wrote for 6.863, the natural language course that he taught in the spring term of 1980 at MIT. Bill hoped these notes would evolve into a book through the usual process of teaching and revising. Unfortunately, with Bill's sudden illness, it became impossible to realize this plan. At the time of Bill's death, the notes were in very serious need of revision. There was no introduction, conclusion, bibliography, etc. In addition, many sections are not as clearly written as they ought to be. It seemed impractical to rectify all of these problems (as one of the reviewers pointed out). On the other hand, we really wanted to make sure that these notes would become widely available. Therefore, at the suggestion of Bill's family, friends and collegues, we decided to publish the notes with only minor editorial "corrections". We ask the reader to be patient with the unfinished condition of the result. In addition, we apologize for mistakes that may have resulted from the editorial process.[1]

These notes impose a number of difficult demands upon the reader; 6.863 was the third and final course in the MIT natural lanaguage sequence, following a general servey in AI and a more detailed look at representation and expert systems (in education, symbolic manipulation, medicine, business, and natural language). These two courses would generally filter out those students with only a passing interest in Artifical Intelligence. Most 6.863 students would soon be persuing a doctorate in expert systems, if not in natural language. In contrast to these demanding prereqs, (almost) no previous background in philosophy and linguistics are assumed of the student, beyond what one might find in a typical undergraduate education. Admittedly, this is an unusal set of prereqs, both very demanding in AI and yet lax in other areas of cognitive science. We ask the reader to bear with these inconviences; these notes are full of Bill's unique perspective, which is extremely original, and not available in any other form.

1.2 General Background

Bill was an engineer at heart. In an era when many researchers in AI were constructing demonstration systems, Bill liked to build useful programs that really worked. Macsyma [57] is perhaps the best example, a program that he and Joel Moses designed fifteen years ago, and is still important (and potentially profitable) today. The work to be described here on an English Query System also illustrates Bill's practical side.. While most of the other parsers at MIT addressed limited and/or idealized subsets of English (e.g., Winograd [109], Marcus [60], Church [17]), Bill attempted to parse real sentences as they appear in practical situations. The purpose of the EQSP paper (Appendix I), in Bill's words:

1. For instance, there are almost certainly some errors in the references, which were added in the editorial process. We have found that it is very difficult to deduce the intended reference from the text alone. In some cases, we were fairly sure which author Bill was refering to, but not which paper. Even the author in question could not always help us.

r to share our philosophy and experience in adapting a well-known context free parsing algorithm (Earley's algorithm [20, 21] and variations thereof [92, 34, 77, 78]) to the parsing of a difficult and wide ranging corpus of sentences. The sentences were gathered by Malhotra [58] in an experiment which fooled businessmen users into thinking they were interacting with a computer, when they were actually interacting with Malhotra in another room. The sentences are given in Appendix I. The Malhotra corpus is considerably more difficult than a second collection given in Appendix II (originally published in [39]).

In part, his attraction toward experimental methods is an outgrowth of his belief in *small infinities.* In many domains (e.g., symbolic manipulation and natural language), it has been his experience that it is possible (and expedient) to solve typical problems with an exhaustive enumeration of special cases. Most researchers would be reluctant to undertake such an approach, concluding after a quick glance that there were an infinite number of special cases. However, in domain after domain, he found that the "infinity" was considerably smaller than one might think. He had the patience and energy to attack a problem head on where others felt compelled to search for an "unconvential" get-rich-quick solution.

The basic straight-shooter philosophy is that the problem domain and the problem-solving model should be carefully digested as a first step toward implementation of the understanding program. The 'digestion' process consists of thoroughly compartmentalizing the relevant knowledge of the field... 'a place for everything, and everything in its place'. Thus, representational issues become extremely important...

Now this is hardly a new way of doing things. It has been the basis of writing all sorts of 'conventional' computer programs (i.e., almost everything that works)... the straight-shooters can point to 'unconvential' ... problems (like finding the 3-D picture from a set of lines [106] or mathematical integration [76]) which was flailed at for years by fancy 'unconventional' methods and then eventually solved in a trice by stright-shooting." Martin [62, pp. 5-6]

To recapitulate, Bill focused heavily on the importance of a large number of simple experts. He believed that a successful project consisted of a 1000 or more such experts, 200 or so unavoidable engineering decisions, 20-50 great tricks and 10 or fewer powerful ideas [62, p. 9]. An expert might be something like an arc or a part of speech in a grammar. An unavoidable engineering decision in Macsyma is the relationship between binary and unary minus. A great trick, again from Macsyma, is the Henzel lemma which speed up operations like factoring from days to seconds, thus changing the whole nature of what could be done for the user. A powerful idea is the CONS operation in LISP. Powerful ideas are often so elegant that it can take decades for practice to reveal their true role as a basis for "good thinking".

1.3 Motivation for Question Answering Systems

2

The study of English grammar and parsing may be pursued to either scientific or engineering ends. As pure science, it increases our knowledge of how to describe the natural world. Such scientific knowledge finds practical application in the recording and teaching of languages and it may be used as an avenue to scientific theories of human cognition.

In engineering, there are many potential computer applications which require natural language processing. Some of these require speech processing, some text processing, and some could be done either way. It is generally agreed that speech processing is much more difficult than text processing. This discussion will be limited to text processing. A computer system doing a text processing application may be designed to

2. This section was written by Bill Martin. It originally appeared as the introduction to the second half of the notes.

emulate humans doing that task, in so far as we understand what humans do. However, it need not follow human behavior in detail, as long as it gets needed results in the end. Emulating human behavior is a useful design strategy, and study of human processing may show that humans are exploiting constraints in the language which can be exploited by computer systems as well. But experience has shown that because computer systems have different memory and processing capabilities than humans, they often do best using different techniques than humans. In fact, in an area like computer processing of symbolic mathematical expressions, there has been a gradual evolution away from emulation of humans to the use of methods requiring the manipulation of massive amounts of detail according to complex rules. Only computers can do this quickly and without making a mistake. The two situations are not exactly parallel, since people manipulate large mathematical expressions explicitly, while they process natural language implicitly — they can't explain very well, and aren't completely aware of, what they are doing. Nevertheless, when we consider how observations about human processing might apply to computers, some of the same questions as to what is complex for a given processor will arise. Our goal, then, is engineering applications of text processing.

There are three general forces driving the development of computer text processing. First, many people now have access to, and could profitably read, much more material than they can possibly get through. (For example, the volume of articles in medical journals has reached the point where there are not only journals devoted to abstracts of articles, but journals of abstracts of the abstract journals.) Since much of the information in business, medicine, law, military intelligence, etc., is not easily reducible to numbers or mathematical models, one envisions that natural language will continue to play an important role in its dissemination. A second underlying force is the need for computers to communicate with people in terms they can understand. In any area of endeavor a vocabulary is developed which allows clear and efficient communication among the initiated. If a computer is used in this environment, much difficulty is created if, as is usually the case, it can't use the standard terms. Middle men have to be hired as translators between man and machine. In small organizations this is prohibitively expensive, and in large ones it introduces delays, costs, and misunderstandings. A third, and longstanding force, is the need to translate from one language into another. These three forces (access to large data bases, communication between man and machine, and language translation) have led to the development of a variety of software facilities:

(1a) Database Query: Formal languages for the query of data bases are in wide use. Several systems for English query of databases have been developed and a few have received limited, but practical and successful, use.

(1b) Help Facilities: Help facilities are widespread, but they generally proceed by asking the user multiple choice questions. Limited investigation of natural language help facilities has been undertaken.

(1c) Naive User Programming Systems: Several personal and small computers come with software which leads the user through specification of his algorithms. Very limited research has been done on natural language specification of algorithms.

(1d) Tutoring Programs: Programs have been developed which not only present self paced material to a student but also present him with problems and computer aids to solve them. Some success has been had in developing a natural language front end for such a program.

(1e) Explanation of Program Actions: Programs which can explain their calculations exist only in the research environment. So far, only limited use of natural language has been made.

(1f) Acquisition of User Problems by "Expert" Systems: There are several "model based" systems which acquire a description of a user's problem and then run their model to produce an answer. Very limited use of natural language has been made in these systems.

(1g) Scanning Text on a Particular Topic: Limited research has been done on scanning text such as AP wire service stories on particular topics and automatically building up a database. Research has been done on reading nurse's notes. These systems respond to questions about what they have read.

(1h) Checking System Specifications: A particularly interesting type of text for computers to read is the specifications and documentation of large software systems. Some research has been done on this.

(1i) Document Retrieval: Many systems are available for this. Indexing of documents by keywords is the primary retrieval mechanism. These systems are useful, but they leave much to be desired. No one has attempted this task using natural language.

The above are all examples of how engineering applications of natural language processing can be made in the future. These problem areas serve to delimit the types of natural language processing of interest here. By looking at the current state of the art in some of these areas, we can get a better idea of what a natural language processing system must do.

1.4 Course Outline

6.863 focused on the intersection between Philosophy (ontology, logic, scope) and Linguistics (syntax and semantics) and Artificial Intelligence (representation, natural language and learning). The course was taught in two parts: Representation of Knowledge and Natural Language Understanding. The first part is more philosophical and theoretical; the second is more linguistic and practical. A third part, Learning and the Semantic Base, was planned but was not included in the written notes or in the oral lectures due to time constraints.

1.4.1 Outline of Part I: Knowledge Representation

Part I begins with a philosophical view of representation and moves progressively toward an engineering orientation. Bill introduces basic 'uncontroversial' notions like individuals and sets, which we all think we understand. But, these concepts are not so simple, as philosophers are fond of demonstrating with numerous puzzles. If we have two spheres, identical in every property except for position in space, do we have one or two individuals? If we change an individual just a little bit, do we have one or two individuals? How much do we have to change an individual before we say that we no longer have the same individual? These questions probably do not have very nice elegant solutions. Bill believes the representation should have a place for both answers: 'a place for everything and everthing in its place'. This goal can be accomplished with simple predicate logics and/or semantic networks.

The rest of chapter 2 introduces the tricky notions of meta-description and scope. What is the truth-value of the liar's paradox: *This sentence is false*? What is the scope of the quantifiers *two* and *three* in the sentence: *Two lookouts saw three boats?* Are there two, three or six sightings? Does the sentence *The boys danced with the girls* mean that the boys and the girls danced as couples or altogether, e.g., in one giant circle?

Fig. 1. Course Outline - Spring 1980

February	5	Overview	PART I
	7	General Ontology	
	12	Abstractions	
	14	Strucutural Descriptions - Roles and Contexts	
	21	Reasoning and Deduction in Semantic Nets	
	26	Descriptions of Procedures and Events - and Procedural Descriptions	
	28	Reasoning with and about Procedures	
March	5	An Example Semantic Net - LMS	
	7	LMS	
	12	Question Answering Systems - Overview	PART II
	14	Lexical Syntax	
	19	Lexical Syntax	
	21	Lexical Syntax; System Assessment Paper Due	
April	1	Parsing and Semantic Interpretation	
	3	Parsing and Semantic Interpretation	
	8	Parsing and Semantic Interpretation	
	10	Pragmatic Interpretation	
	15	Pragmatic Interpretation - Discouse Structure and Anaphora	
	17	Pragmatic Interpretation - Logical Form	
	24	Domain Specific Semantic Models	
	29	Answering Questions	
May	1	Story Understanding; Programming Project Due	PART III
	6	Learning	
	8	Deep and Surface Cases	
	13	Primitive Concepts	
	15	Structure of the Lexicon; Term Paper Due	

Most solutions to these puzzles impose an environment structure in order to express the dependency of concepts on other concepts. These solutions will be introduced at the end of the next chapter.

Chapter 3 discusses a number of the more difficult problems in refering to individuals, e.g., intensionality, opaque contexts, and attributive descriptions. Consider for example, the opaque operator *so-called* as in: *Giorgione was so-called because of his size.* From this statement, it is inappropriate to conclude that *Barbarelli is so-called because of his size*, even if *Giorgione* and *Barbarelli* are two names for the same fellow. The principle of substitution is shown to fail in a number of other opaque contexts: names, predicate nominatives and belief contexts. Donnellan's attributive description raises a similar problem. From *The President has lived in the White House since 1800*, it is inappropriate to conclude that *Nixon has lived in the White House since 1800.* On the other hand, substitution is possible in the referential case, e.g., *The president has been married since 1945* has the same truth value as *Nixon has been married since 1945* in the context where *the president* refers to *Nixon.*

How should these opaque contexts be represented inside the computer? Three approaches are suggested: (a) semantic networks, (b) second order predicate calculus, and (c) Montague's intensional logic. Bill favors semantic networks and introduces two types of links, <u>role-in</u> and <u>characterization-of</u> to address some of the puzzles presented thus far. Both links are defined to be intensional relations in order to block inappropriate substitution.

A role-in link would be used, for example, in the sentence *John believes in unicorns* to relate the intensional reference *unicorns* to the belief context, expressing the fact that *unicorns* are individuals <u>relative</u> to the set of John's beliefs, but not necessarily in the larger context. In contrast, in the sentence *John believes in reality*, the extensional reference to *reality* will not be linked to John's beliefs, since *reality* is an individual in the larger context and not dependent on the set of John's beliefs. In short, intensional expressions are linked to a context such as John's beliefs with a role-qin link in order to express the dependency; extensional expressions are free of role-in links, since these contexts are independent of context. As Bill says in Chapter 4 (citing Eaton), "things like unicorns <u>subsist</u> rather than exist."

Role-in links are also used to express Skolem dependencies. A sentence like *Every integer is greater than some integer* is represented by linking the quantified expression to the appropriate context. Thus, if *some integer* is linked to *every integer*, *some* will be interpreted within the scope of *every* (so-called <u>narrow</u> scope). If, on the other hand, *some integer* is not linked to *every iteger*, *some* will be interpreted as outside the scope of *every* (so-called <u>wide</u> scope). Role-in links express contextual dependencies (e.g., narrowly scoped quantifiers, intensional expressions, attributive expressions, opaque contexts). The absence of role-in links indicates contextual independence (e.g., widely scope quantifiers, extensional expressions, referentially transparent expressions).

Chapter 4 shows how semantics networks can be used to implement frame systems and predicate calculus, along the lines discussed in Woods' "What's in a Link" [114]. Section 4.5 introduces the inheritance problem. How does an instance node inherit properties from the generic node. The generic arch, for example, consists of three blocks: two supports and a top. However, it is possible to form an arch out of a single block like the McDonald's Golden Arch. Thus, there will have to be some exception handling mechanism for overriding the generic case.

Sections 4.10-4.12 discuss the representation of frame systems over change and <u>the frame problem</u>. It is not enough to specify what changes as the result of an event; we must also specify what remains the same. A number of proposed solutions to the frame problem are outlined in section 4.10. Section 4.11 elaborates an AI approach: CONNIVER's context hierarchies, an effort to factor situations into changing and constant parts. Section 4.12 reviews a philosphical attack: Karp's attempt to model time by analogy with space. "John, the boy", "the ten year old John" and "the yong John" are modeled as individuals just like "the left part of the arch", "the right part of the arch" and "the top of the arch". Both this philosophical approach and the AI context hierarchy solution are insightful, but not entirely successful.

In chapter 5, Bill attempts to relate the structural notions introduced in chapter 4 with <u>Ontology</u>, the branch of meta-physics dealing with the philosophical theory of reality [30, p. 944]. He reviews some work by Eaton and Lyons and outlines how these positions can be formulated within his semantic network formalism. Like so many of the philosophical questions encountered thus far, ontology is laden with open problems. Bill mentions Winograd's observation that most concepts are extremely difficult to define. Consider the term *bachelor* which might seem to be fairly straightforward; it might seem that a bachelor is simply an unmarried man. But this definition turns out to be grossly inadequate, as Winograd shows with seven counter-examples. Bill concludes the chapter with an outline of a theory of description. Descriptions will be classified into five types for linguistic reasons:

(2a) Attributes: *health, color, length*
(2b) States / Processes
(2c) Relational Characterizations: *top of*
(2d) Attributive Characterizations: *bachelor, teenager*
(2e) Other kinds

Chapter 6 deals with word senses. Bill loved to form long lists of all the ways that a particular word like *run* or *near* could be used. He believed that word senses can be interpreted in three ways: by analogy, through abstraction, according to a taxonomy, etc. Bill discusses a few examples of each of these. *Dog house*, he suggests, is understood by analogy with a house for people. President Johnson's term *transmote* illustrates the abstraction mechanism; the term is interpreted by abstracting the meaning of the morphemes *trans-* and *-mote*. (Transmotion is a mechanism invented by President Johnson for getting rid of someone that cannot be promoted or demoted.) The taxonomy system is the subject of much discussion.

How should the taxonomy be organized? The organization could reflect universal truths: e.g., *the natural numbers are a subset of the integers* or stereo-typical generalizations: e.g., *men are heavier than women.* Bill favors the latter approach since exceptions are so common in natural language. For example, it might be said that *all* dogs have four legs. But if a dog should lose a leg in a car accident, would we say that it is no longer a dog? Bill would prefer to say that the the prototypical dog has four legs, but it is possible under exceptional circumstances to find a particular dog with some other number of legs.

The generalizations in the taxonomy should also be relevant. For example, it is silly to say that *ideas are colorless*, even though it may be literally correct. It is a completely open question how this relevance condition should be formulated. It appears that relevance is a matter of degree. For instance, it is very reasonable to say that *lettuce is edible*, somewhat less reasonable to say that *grass is edible*, and very odd to say that *poison is edible.* Bill discusses a number of different ways that one might want to address this condition.

Chapter 6 is concluded with a number of examples illustrating how abstraction, analogy and taxonomies could be used in order to understand word senses. This discussion reveals Bill's tendency toward straight-shooting by suggesting a society of agents (special case experts) to disambiguate among various competing interpretations.

Chapter 7 ends the discussion of knowledge representation with a review of procedural interpretions of semantic networks. Most traditional logicians focus on the declarative representation of knowledge, rather than the procedure for interpreting the knowledge. As a reaction to this declarative tradition, procedural semantics emerged in the early 1970's, arguing that knowledge should be formulated in the form of procedures, rather than data structures. For example, the knowledge that every boy wants a lion could be cast in the form "if one were to check every boy in question and count those who want lions, then the count of these who want lions would be equal to the count of the boys checked." The debate over procedural/declarative semantics in AI has been heated and inconclusive. There is probably merit in both positions. Procedural formulations shed more light on processing issues, especially time and space efficiency, whereas declarative perspectives are probably more suitable for addressing certain classic philosphical and linguistic questions such as scope relations, intensionality, meta-description and so forth. In addition, it is probably useful to try to factor declarative knowledge (the *what*) from the flow of control (the *how*), just as Chomsky factors linguistics into competence and performance [aspects, §2], and as Marr distinguishes the implementation level from theory representation in his three levels of vision [61, p. 25].

The emphasis on procedural semantics has been insightful. It has shown us that there are many different ways to interpret a concept, each with its advantages and disadvantages. For instance, there are efficiency trade-offs between different ways of thinking about quantifier scope. One could use Skolem dependencies, as Bill does [63] or one could use Woods' iterative formulation. These two schemes index the representation in reverse directions.[3] Similarly, there are quite a number of different ways to bind a variable, e.g., shallow binding, deep binding, pattern matching, etc. In addition, there are quite a number of different ways to invoke a procedure: e.g., call by name, call by reference, pattern directed invocation, etc. And there are different ways to preprocess (compile) a procedure. Bill will discuss a number of these issues, with examples taken from Fahlman's NETL [24].

1.4.2 Outline of Part II: Natural Language Understanding

Chapter 8 discusses the apportionment of the grammar. The line between syntax and semantics is not always so clear. Some people (e.g., Schank) have proposed systems that combine syntactic and semantic processing into a single step. However, Bill believes in the Autonomy Thesis, that syntax and semantics should be kept distinct. The autonomy thesis is most popular among linguists, especially MIT-style syntacticans. Some proponents of the autonomy thesis, e.g., Chomsky and Bresnan, believes that syntax is further subdivided into a number of levels of representation. Chomsky discusses deep structure (the input to the transformational component), surface structure (the output from the transformational component), phonetic representation, θ-roles (thematic roles such as agent, goal, receipient), and so forth. Bill tends to favor a framework more like Bresnan's where it is assumed that syntax[4] is divided into constituent structure (e.g., noun phrases, verb phrases), grammatical relations (e.g., subject, object) and thematic relations (e.g., agent, goal, receipient).

Chomsky and Bresnan do not agree on the status of grammatical relations. Chomsky does not believe that grammatical relations should be granted a separate level of representation in the grammar. He believes that the grammatical relation *subject*, for example, is always the noun phrase immediately dominated by a sentence node. Since the subject relation can be expressed in terms of the surface structure *with no loss of generalization*, the grammatical relation should be eliminated from the grammar as superfluous. Bresnan

3. For instance, given the sentence *Some man likes himself*, in Bill's system, *himself* would be linked to *some* with a Skolem dependency whereas in Woods' system, *some* would be scoped over *himself*.
4. Bill often uses the term syntax in a very narrow sense, referring only to the constituent structure and apportioning the rest of the grammar to semantics.

believes that grammatical relations capture crucial generalizations.

This disagreement manifests itself in numerous places in the grammar. For instance, consider the passive operation which relates sentences like:

(3a) Sincerity frightens John. *active*
(3b) John is frightened by sincerity. *passive*

Chomsky accounts for this relationship by performing transformations on the parse tree, so that both sentences have the same underlying deep structure. Bresnan accounts for this relationship by assuming a passive lexical entry *be frightened* which is like the active entry *entry* except for the specification of grammatical roles. The active entry takes a subject (a thing doing the frightening) and an object (a recepient of the action). The passive entry takes a subject (which is identical to the object in the active case) and an optional by-phrase (which is idental to the subject in the active case). Bresnan takes a <u>lexicalist</u> approaching by account for the active-passive relationship in the lexicon; Chomsky takes a <u>structuralist</u> approach by accounting for the relationship by transforming the parse tree.

The remainder of Chapter 8 discusses four major components to a syntactic parser: ambiguity, agreement features, wh-movement and conjunction. Ambiguity comes in many varieties. Bill is particularly interested in lexical ambiguity (a word has two or more parts of speech) and structural ambiguity (a sentence has two or more parse trees).

(4a) I went to work$_{N, V}$.[5] *lexical ambiguity*
(4b) Put the block [in the box] [on the table].[6] *structural ambiguity*

In many cases, possible sources of ambiguity can be resolved by context. For example, the lexically ambiguous word *schedule*$_{N, V}$ can be disambiguated in the following two sentences.

(5a) I lost my <u>schedule</u>$_N$ in the lecture. *noun*
(5b) I plan to <u>schedule</u>$_V$ the lecture you want. *verb*

In these sentences, it is advantageous to determine the part of speech of *schedule* after the surrounding context. Such a strategy is consistent with the "wait and see" philosophy, where the parser makes the easy decisions first, and defers the harder ones until there is more information on which to base the decision. This "wait and see" philosophy is known by many other names: e.g., delayed binding, lazy binding, principle of least commitment.

5. *To work*$_N$ is interpretated as a destination; *to work*$_V$ is a purpose construction.
6. The phrase *in the box* can be attached to the noun phrase *the block* or the verb phrase. If it is attached to the noun phrase, the sentence is interpreted as a command to move the block which is <u>already</u> in the box and place it on the table. Otherwise, the sentence is interpreted as a command to move the block <u>into</u> the box that is on the table.

There are many different types of contextual cues that can be used to disambiguate potential sources of ambiguity. Agreement constraints are particularly effective. The following sentences illustrate the usefulness of number features on the verb. Note that sentences (6a-b) are unambiguous, in constrast to (6c), where the verb *can* takes both singular and plural subjects.

(6a)	$[_{VP}$ Flying planes] are dangerous.	*plural*
(6b)	$[_{NP}$ Flying planes] is dangerous.	*singular*
(6c)	$[_{NP,\ VP}$ Flying planes] can be dangerous.	*ambiguous*

grammatical roles and thematic relations. The active form assigns transformation that wait and see
lexical ambiguity
aux-inversion features attachment ambiguity wh-movement conjunction

generative capacity questions

CHAPTER 9: parsing top-down / bottom-up Earley Systems
Semantic Grammars
Determinism Conjunction

CHAPTER 10: lexical syntax and word senses
X-bar theory
grammatical frames
senses of run
historical derivations
syntactic frames
lexical rules
space and time
partial and complete conversion
noun phrases
noun-noun modification

2. Philosophical Foundations

2.1 Introduction

In surveying the history of knowledge representation, one sees a small number of basic notions combined in a variety of ways to produce different representation schemes.

• individuals	• predicates	• description
• sets	• relations	• meta-description
• structure	• denotation	• logic

Most schemes use only a subset of the notions, and so different schemes usually lead not only to different data structures, but also to different philosophical claims about the fundamental nature of reality and its representation. This chapter will present some of these notions, discussing each, and then show some ways they have been combined.

The goal of this chapter is not just to introduce these notions, but also to bring out the relationship between the idea that a description is true to-matches-reality and the idea that a logical proposition is true. We hold that truth of propositions is probably the simplest way of describing this relationship. There is often said to be a trade-off between the simplicity of a language (which facilitates its study as an abstract object) and the simplicity of its use. This point of view would hold that truth theory is probably too simple a model of the correspondence to reality. Many agree, but claim that it is a useful starting point in avoiding fallacious deduction, and can be augmented with other description for computational purposes. The research question then becomes: Is this true, and if so, what form of logic can be most naturally augmented for use in computational linguistics?

2.2 Individuals

That people think of the world as containing individuals is probably the most uncontroversial assumption with which to start. As Strawson [99] puts it:

> We think of the world as containing particular things some of which are independent of ourselves; we think of the world's history as made up of particular episodes in which we may or may not have a part; and we think of these particular things and events as included in the topics of our common discourse, as things about which we can talk to each other. These are remarks about the way we think about the world, about our conceptual scheme. A more recognizably philosophical, though no clearer, way of expressing them would be to say that our ontology comprises objective particulars.

As examples of objective particulars Strawson lists historical occurrences, material objects, people, and their shadows.

The idea of individuals seems quite natural, but there are circumstances which show more clearly how the existence of individuals must be viewed as an assumption people make. Consider, for example, a world consisting of nothing but two spheres rotating around their common center of mass, Black [5]. Since the two spheres are indeccernable, can it make any sense to talk about separate individuals? People are prepared to do it; typically a person looks at the problem as one of developing a test which will let the two individuals be distinguished.

As another example, consider an individual place, like one's favorite spot in a forest. What grounds have we for considering this an individual? Couldn't we also have another place just slightly shifted from this, thought to include or leave out an additional tree? The forest has as many individual places as we wish to conceive of. There is nothing innate about the forest which determines which individual places we will choose to identify. We take a substructure of a structure like a forest to be an individual whenever it is an aid to thought.

It may be argued that, indeed places are not individuals, even though we sometimes speak of them as such. This is the position taken in Lyons [55] and one can see there the complexities introduced because for example *this is a nice place* and *this is a nice apple* are not semantically parallel. We can certainly question the exclusion of places as individuals given our criterion of providing English with as straightforward a semantic interpretation as possible.

The issue of whether places are individuals can also be considered from a more fundamental perspective. What is the operational reason for assuming the existence of individuals? We think it to be that the individual has a certain structure which gives it useful and interesting properties and that the structure of an individual is predictable over time to a significant degree. Predictable enough so that the individual can be re-recognized at succeeding points in time and so that plans can be formulated based on the assumed persistence of an individual and its properties. If it were the rule of the day for frogs to turn into princes and such, one would have to live very much more on a moment to moment basis and individuals would surely play much less of a role in our thought. Under this criterion places are individuals because they have an interesting configuration of properties that persist through time.

A similar line of thought may be found in Quine [82]. He argues

> The totality of our so-called knowledge or beliefs, from the most casual matters of geography and history to the profoundest laws of atomic physics or even of pure mathematics and logic, is a man-made fabric which impinges on experience only along the edges. Or, to change the figure, total science is like a field of force whose boundary conditions are expensive. A conflict with experience at the periphery occasions readjustments in the interior of the field ...
>
> As an empiricist I continue to think of the conceptual scheme of science as a tool, ultimately, for predicting future experience in the light of past experience. Physical objects are conceptually imported into the situation as convenient intermediaries - not by definition in terms of experience, but simply as irreducible posits comparable, epistemologically, to the gods of Homer. For my part I do, qua lay physicist, believe in physical objects and not in Homer's gods; and I consider it a scientific error to believe otherwise. But in point of epistemological footing the physical objects and the gods differ only in degree and not in kind. Both sorts of entities enter our conception only as cultural posits. The myth of physical objects is epistemologically superior to most in that it has proved more efficacious than other myths as a device for working a manageable structure into the flux of experience.

It is probably going too far to say that physical objects are a cultural posit if this implies that the notion comes only from interacting with other people. Many animals rely heavily on instinctive behavior and in man a substantial part of the genetic code is devoted to the nervous system. There is no reason to believe a species can't pass components of a successful conceptual strategy by heredity. Since Quine wrote the above quote, psychologists have come to believe much more in inherited perceptual and conceptual strategies. While our our conceptual structure may be a man made fabric, it is perhaps made by man as a species, and passed down both genetically and socially.

Recall that Strawson referred only to "objective particulars." examples of which were historical occurances, material objects, people, and their shadows. Should one admit "subjective particulars" in a representation scheme. e.g., particular concepts or particular heights? This question is more controversial than whether any individuals exist. The chapters to follow will bring out some of the representational issues involved in this decision.

2.3 The Ship of Theseus

People often get a better idea of how they themselves think about individuals by considering the story of the Ship of Theseus. A nice version of this story is told by Chisholm [14].

> Let us imagine a ship—the Ship of Theseus—that was made entirely of wood when it came into being. One day a wooden plank is cast off and replaced by an aluminum one. Since the change is only slight. there is no question as to the survival of the Ship of Theseus. We still have the ship we had before: that is to say, the ship that we have now is identical with the ship we had before. On another day, another wooden plank is cast off and also replaced by an aluminum one. Still the same ship, [CHECK THIS QUOTATION] since, as before, the change is only slight. The changes continue in a similar way and finally the new Ship of Theseus is made entirely of aluminum. The aluminum ship, one may well argue, is the wooden ship we started with, for the ship we started with survived each particular change, and identity, after all is transitive.
>
> And what happened to the discarded wooden planks? Consider this possibility suggested by Thomas Hobbes: 'If some man had kept the old planks as they were taken out, and by putting them afterwards together in the same order, had again made a ship of them, this, without doubt, had also been the same numerical ship with that which was at the beginning; and so there would have been two ships numerically the same, which is absurd.' Assuming, as perhaps one has no right to do, that each of the wooden planks survived intact throughout these changes, one might well argue that the reassembled wooden ship *is* the ship we started with. 'After all, it is made up of the very same parts, standing in the very same relations, whereas that ugly aluminum object doesn't have a single part in common with our original ship.'
>
> To compound the problem still further, let us suppose that the captain of the original ship had solemnly taken the vow that. if his ship were ever to go down, he would go down with it. What, now, if the two ships collide at sea and he sees them start to sink together? Where does his duty lie—with the aluminum ship or with the reassembled wooden ship?

This story shows that there can be different strategies for the identification of individuals through time. A strategy which leads to the choice of the wooden ship can be described as follows:p

(7a) Identify an individual from its structural description, and from the known identity of the individuals which are its components.

A strategy which leads to the choice of the aluminum ship is:

(7b) Identify an individual from its structural description, but don't be concerned with whether its components are the same individuals as long as they meet the same description. In fact, don't require them to be exactly the same (wood $\rightarrow$ aluminum is OK). Allow some changes as long as the whole individual maintains continuity in time and space.

Presumably continuity in time and space is determined by the individual's spatial relationship to other individuals. Thus this second method implicitly assumes that we know the identity of some individuals in environs of the given individual, or that we treat its whole environment as a persistent individual. Methods (a) and (b) contrast in that (a) looks inward to the individuals in a given individual's structure, while method (b) looks outward to individuals in its context or containing it.

The Ship of Theseus presents many people with a problem because they don't, in fact, have a unique method of identifying an individual through time. People need not be aware of or concerned by this as long as all their methods give the same answer. It is the business of philosophers to find the trick cases and then argue whether there is one correct method. We are interested in the concepts needed to describe how people might be doing it. Here it seems we need the notions of individuals and structural descriptions.

It might be well to point out that the ship's captain could deal with this problem without having any identifying description of the Ship of Theseus. He would just ask someone *Where is the Ship of Theseus?* He believes there is one ship and he calls it by name. This makes the method of identifying the ship someone else's problem. It is an important strategy for computers, blind and deaf as they are.

2.4 Notation for Individuals

It is convenient to introduce two types of notation for describing individuals; constants and variables. Constants are names. They will be written in italics and will designate a particular individual. Variables will be written in capital letters and will match, or be satisfied by any individual.

2.5 Representation of Structure

Formal theories of semantics generally include some concept of individuals. In addition, it is recognized that the world has structure—that is— that individuals may be related in various ways. For example, some individuals are parts of others or an individual may be in a particular spatial relationship to some others. We saw the use of this in the last section. A formal theory has to have some way of representing this. To do this requires at least one fundamental notation for structure.

Suppose an individual is in some relation to a second individual. A theory may choose to call the relation itself an individual, or it may not. For example, suppose Bob loves Mary, a theory may choose whether or not to take Bob's love of Mary as an individual, an individual love.

But given only a notation for denoting individuals, there is no way to describe the fact that an individual x is in relation R to an individual y, whether or not R itself is taken to be an individual. We must have some notation which shows that x and y are in R. Note that this notation must involve the notation for x and y, for example, xRy. Therefore, we see that at a very basic level there must be a structural correspondence between structures in the world and the notation for these structures. However, because given a general notation for structure, it is possible to simulate all kinds of structures, the exact nature of the correspondence between the structure of a notation and the structure of what it denotes is something we must investigate.

Traditional formal semantics is based on Zermelo set theory. Set inclusion is often used as the basic structural concept. Set inclusion, $\in$, is an irreflexive, non-transitive relation. Using $\in$, we might represent *Bob loves Mary* as:

(8) *Bob* $\in$ *lovers* & *Mary* $\in$ *lovees* & *Bob* $\in$ *love-1* & *Mary* $\in$ *love-1*

where *Bob* and *Mary* denote individuals which are atomic and *lovers, lovees* and *love-1* denote individuals which are sets.

This representation requires us to postulate the existence of the sets *lovers, love-1* and *lovers*, which may seem unnatural. It also leads us into some technical issues. First it is important not to confuse a description of structural relations between denoted individuals with a description which is uniquely satisfied by certain individuals. Suppose that if instead of Bob loves Mary; Mary loves Bob and Harry loves Mary and Bob loves Sue, then the above structural relations between people and loves will still hold, but they won't be the same ones. We have named a certain love, *love-1*, and it is an instance of *Bob loves Mary*, not *Mary loves Bob*. Second, recall that in standard set theory, an individual set is identified only by its members, and not at all by the sets it is a member of. Thus, if everyone who is a lover is also a lovee, standard set theory will not allow us to represent distinct individual sets *lovers*, containing exactly the lovers, and *lovees*, containing exactly the lovees, as members.

One way to solve this problem would be to include in every set an individual which is a member only of that set and the set, *self. Lovers* and *lovees* would then have different elements which were members of *self.* It seems unlikely that people believe in the self individual of a set.

Perhaps a more conceptually natural solution is to assume that it is a mistake to represent structures by set inclusion, since sets are uniquely determined by their members, but structures are not uniquely determined by their members unless we elaborate their structural descriptions just for the purpose of making them unique (this claim is discussed further in Chapter 4). On this view it would be more natural to retain $\in$ for set membership, but to model structure with a irreflexive, intransitive relation, S. If xSy we will say that x is a component of y. As opposed to sets and their members, two distinct individuals can have the same components.

This formulation removes the technical problem, but it still leaves us with these queer individuals which have all lovers on lovees as components. These can be gotton rid of with the introduction of predicates in Section 2.7.

2.6 Comparison Made Important by Redundancy in the World

Besides the existence of individuals and structure, another important characteristic of the world is that it is repetitive. There is more than one tiger, elm tree leaf, or rising of the sun. Since the behavior of one tiger, etc., is much like that of another, an important mode of thinking is to assume that the properties and behavior of the next tiger will be the same as those of tigers previously encountered. This is a complex problem, but one thing it definitely requires is the ability to recognize when one individual is similar to another. This in turn requires that one can tell when two individuals are identical,[7] because even schemes which compute partial matches require identity between some components of the structures matched.

7. "Identical" is used here in the sense of the LISP programming language EQUAL, not EQ.

Given only the notations for individuals and structure introduced so far, the only way that individuals can be compared is by comparison of structure. One can compare both (a) the structures of the two individuals, and (b) the structures containing the two individuals and their component structures. For example, suppose we want to compare *Bob loves Mary* with *Harry loves Betsy* using the representation of the last section. These could be compared as shown diagrammatically in Figure 2 (but note that we have not yet discussed the problem of representing a comparison of representations). Note that when we take into account the structures which contain their components, there is only one way to match *love-1* and *love-2*. However, looking at only their internal structure, they would match in two ways.

Let us define a <u>structural description</u> as a set of expressions of the form *xSy*, where *x* and *y* are either constants or variables. (This definition will be expanded and refined later.) The structural description of *love-1* might be (8). Building on our notion of satisfaction of a variable we can seek a definition of the satisfaction of a structural description containing variables. The proper definition is a question for research if we wish to allow underspecification as is suggested in Section 2.11.

2.7 Predicates

Given only individuals and structure, our only method of describing similarity is to create an individual having all the individuals which are similar in a certain way in its structure. This was what was done with *lovers* and *lovees*. In a similar way, the color of an individual could be represented as *red* by putting him in the structure of *red-individuals*. At the expense of another notion, one can get rid of individuals like *red-individuals*.

The idea of predicates arises naturally from the point of view that given an objective individual, one can test whether he is red or not by looking at him. One way of formalizing the representation of the knowledge gained from such tests is to let our notation consist of some finite stock of symbols, called predicate symbols, each corresponding to a test and for each of which, we have notation indicating that the test yielded a yes answer for certain individuals.

There is no compelling technical reason to associate predicates only with tests conceivably performed by the senses. Thus, using predicates we can reformulate the example in (8) as

(9) *Bob* $\in$ *love-1* & *Mary* $\in$ *love-1* & <u>lover</u>(*Bob*) & <u>lovee</u>(*Mary*)

where *Bob*, *Mary* and *love-1* are individuals as before, and <u>lover</u> and <u>lovee</u> are predicates.

Note that when we know nothing about predicates but some individuals for which they are true, then a predicate which is true for only one individual tells us nothing. The use of predicates is to model the redundancy of the world. The same predicate will be true for more than one individual. It indicates they have something in common. It gives us an additional basis for comparison, and thus prediction.

Often predicates are defined not just on single individuals, but on tuples of individuals. For example, a predicate on two individuals might test if one is on top of the other. Given a method of modeling structure one can always define an individual having the individuals in the tuple in its structure. A one place predicate

Fig. 2. Matching Structures
<==<fig1-2.press<

Love-1 and *love-2* can be matched by matching their structures and the structures containing them and their components.

can then be applied to this new individual. Thus given a notation for structure, only one place predicates are needed.

2.8 Negation as Meta-Description

By meta-description we will mean a notation which denotes other notation. There are two general reasons for meta-describing a description of reality:

(10a) To talk about the correspondence between a description and reality. (Knowledge about such correspondences is traditionally referred to as synthetic knowledge).

(10b) To talk about properties of the description itself, such as the number of symbols involved or which descriptions were computed from others. This knowledge is useful in planning computational strategies. (Knowledge about statements which follow logically from others is termed analytic knowledge).

Suppose a dog, Fido, is born who does not have a tail. By matching we somehow determine that our description of Fido is very similar to that of other dogs we know, except that Fido doesn't have a tail. One way to record this is to create a symbol denoting Fido's tail and then meta-describe this as non-existent, e.g. not corresponding to the world. Note that almost everything we could imagine doesn't exist. It's only necessary to have the knowledge that something is non-existent if there is, loosely speaking, reason to believe it would exist. In this interpretation <u>not</u> is a meta-description which indicates no match with reality.

There is another interpretation of *not* which is sometimes suggested. On this interpretation *Fido does not have a tail* matches some state which is mutually exclusive with Fido's having a tail - a state, e.g., Fido's having a smooth bottom, which convinces us that *Fido has a tail* cannot be true. What is matched may depend on the context of use of the negative statement. This interpretation of *not* perhaps deserves further exploration. No thorough account of this method has been given. The meta-description interpretation seems perhaps more intuitive since it seems possible to understand a sentence like *Bob isn't going to the movies* while thinking only about Bob and going to the movies and not thinking about what else if anything Bob might be doing instead. Perhaps another way to make the point is to note that Bob isn't doing most everything. *Going to the movies* is picked out only because there is reason to believe he might be doing this. It is not picked out just to refer to what he is doing in an oblique way.

2.9 Logical Deduction

Existence is a relationship between one description and the world. A second such relation is uniqueness, which says that only one individual matches the description. There are also relationships between sets of descriptions and the world. A <u>singular</u> description is one which only one object in the world can satisfy. By <u>exclusion</u> we mean that the world is such that anything that satisfies description A can't satisfy description B. By <u>exhaustion</u> we mean that the world is such that anything that satisfies description A will satisfy one or more of descriptions $B_1 ... B_n$.

The familiar deductions of the sentential calculus are seen to be based on knowledge of mutual exclusion by observing that all of the familiar logical operators can be derived by composition of mutual exclusion. Let x|y stand for *not both x and y*. ("|" is termed the Sheffer stroke). Then we can write

~x	for	x|x	negation
x∧x	for	~(x|y)	conjunction
x⊃y	for	x|~y	material conditional
x∨y	for	~x|~y	alternation
x≡y	for	(x|y)|(~x|~y)	equivalence

We now have two methods of computation. The first is based on the notion that the world will repeat itself. It might be called induction focused. If one structure partially matches another, we assume it will match further. The second method is based on exploring the consequences of our knowledge about mutual exclusion. We call it deduction focused.

2.10 The Liar's Paradox

The explanation of logical notation as meta-description is compatible with Kripke's [50] solution to the liar's paradox. The liar's paradox arises when one tries to ascertain the truth of the sentence: *This sentence is false.* One might try to eliminate this problem by saying that sentences can't refer to themselves. But this is not enough because there can be chains of reference, e.g., *The following sentence is false. The preceding sentence is true.* Kripke's solution to this paradox is sketched as follows by Susan Haack: [36]

> *Kripke's solution: groundedness*
>
> Kripke seeks to supply an explanation of the source of paradox which is more satisfactory in this respect, and then to build a formal theory on this basis. (My hunch is that this is the right way round to go about it.) His proposal depends upon the rejection of the idea—taken for granted by Tarski [101]—that the truth-predicate must be totally defined, that is to say, that every suitably well-formed sentence must be either true or false. It thus has affinities both with Bochvar's [8] proposal of a 3-valued logic, and with the no-item proposals discussed above. But Kripke stresses that his idea is *not* that paradoxical sentences have some non-classical truth-value, but they have *no* truth-value.

The key idea in the explanation of how ordinary sentences are assigned truth-values—and how extraordinary sentences fail to get a value—is the concept of *groundedness*, first introduced by Herzberger [40]. Kripke explains the idea as follows:

> Suppose one is trying to explain the word 'true' to someone who doesn't understand it. It could be introduced by means of the principle that one may assert that sentence is true just when one is entitled to assert that sentence, and one may assert that a sentence is not true just when one is entitled to deny it. Now given that the learner is entitled to assert that: 'Snow is white.' This explanation tells him that he is entitled to assert that: ' "Snow is white" is true.'

Now he can extend his use of 'true' to other sentences, e.g., as 'Snow is white' occurs in Tarski 1944, the explanation allows him to assert that: "Some sentence in 'The semantic conception of truth' is true." And he can also extend his use of 'true' to sentences which already contain 'true', e.g. to assert that: ' " 'Snow is white' is true" is true,' or: 'Some sentence in " 'The semantic conception of truth' is true" is true.' The intuitive idea of *groundedness* is that a sentence is grounded just in case it will eventually get a truth-value in this process. Not all sentences *will* get a truth-value in this way; among the 'ungrounded' sentences that won't are: "This sentence is true," and: "This sentence is false."

This idea has affinities with the notion expressed in Russell's [91] V.C.P. (Vicious Circle Principle) and by Ryle [93] and Mackie [56]—that what's wrong with paradoxical sentences is a kind of vicious self-dependence. However, ungrounded sentences are allowed to be meaningful, whereas Russell's idea is that violation of the V.C.P. results in meaningless.

Formally, this idea is represented (I simplify considerably) in a hierarchy of interpreted languages where at any level the truth-predicate is the truth-predicate for the next lowest level. At the lowest level, the predicate *'T'* is completely undefined. (This corresponds to the initial stage in the intuitive account.) At the next level, the predicate *'T'* is assigned to wffs which don't themselves contain *'T'*. It is assumed that this assignment will be in accordance with Kleene's [48] rules giving the assignment of values to compound wffs given the assignment—or lack of assignment—to their components: '-*p*' is true (false) if '*p*' is false (true), undefined if '*p*' is undefined; '*p* ∨ *q*' is true if at least one disjunct is true (whether the other is true, false, or undefined), false if both disjuncts are false, otherwise undefined; '(∃ *x*)*Fx*' is true (false) if '*Fx*' is true for some (false for every) assignment to *x*, otherwise undefined. (This corresponding to the first stage, in which the learner assigns 'true' to a sentence if he is entitled to assert the sentence.) At each level the wffs assigned '*T*' and '*F*' at a previous level retain those values, but new wffs, for which '*T*' was previously undefined, are assigned values—'*T*' gets *more defined* as the process goes on. But the process doesn't go on indefinitely with new sentences getting values at each level; eventually—at a 'fixed point'—the process stops. Now the intuitive idea of groundedness can be formally defined: a wff is grounded if it has a truth-value at the smallest fixed point, otherwise ungrounded. The smallest or 'minimal' fixed point is the first point at which the set of true (false) sentences is the same as the set of true (false) sentences at the previous level. All paradoxical sentences are ungrounded, but not all ungrounded sentences are paradoxical; a paradoxical sentence is one that cannot consistently be assigned a truth- value at *any* fixed point. This supplies some explanation of why 'This sentence is true' seems to share some of the oddity of 'This sentence is false', and yet, unlike the Liar sentence, is consistent. A truth-value *can* be given to 'this sentence is true', but only *arbitrarily*; a truth-value *cannot* consistently be given to 'this sentence is false'. The picture also allows for the 'riskiness' of truth-ascriptions: for the paradoxical character of a sentence may be either intrinsic (as it would be with 'This sentence is false') or empirical (as it would be with 'The sentence quoted on p. 147 ll. 22-3 is false').

2.11 Satisfaction of Structural Descriptions

Using the notation introduced so far, a structural description for *A lookout saw a boat* might be written:

(11) <u>lookout</u>(LOOKOUT) & <u>boat</u>(BOAT) & <u>see</u>(SEE)
LOOKOUT S SEE & BOAT S SEE

This description could be satisfied by a particular see which has a lookout and a boat in its structure. A parallel proposition in first order predicate calculus might take the form:

(12) ∃(LOOKOUT) ∃(BOAT) ∃(SEE) <u>lookout</u> (LOOKOUT) & <u>boat</u> (BOAT) & <u>see</u> (SEE) & LOOKOUT S SEE & BOAT S SEE

This would be true for an appropriate triple of individuals.

Suppose instead we are given the sentence *Three lookouts saw two boats.* What should we take to be the satisfaction conditions of this sentence? Most people would agree that this sentence is true if there were some number of seeings of boats such that the total number of lookouts which saw boats was 3 and the total number of boats seen was 2.

To express this in standard first order predicate calculus, one would typically form explicit sets of the lookouts which saw boats and the boats which were seen. The goal is to write down a statement of the form:

(13) let L be a subset of lookouts & B be a subset of boats then
 L has 3 members and B has 2
 and for all lookouts L_i of L there is a boat B_j of B such that L_i saw B_j
 and for all boats B_i of B there is a lookout L_j such that L_j saw B_i.

This might be written:

(14) $\exists L\ \exists B\ (\underline{count}(L)=3\ \&\ \underline{count}(B)=2\ \&$
 $\forall B_i\ (B_i \in B) \supset (boat(B_i)\ \&\ \exists L_j\ \exists SEE\ (\underline{lookout}(L_j)\ \&\ \underline{see}(SEE)\ \&\ L_j \in SEE\ \&\ B_i \in SEE))\ \&$
 $\forall L_i\ (L_i \in L) \supset (lookout(L_i)\ \&\ \exists B_j\ \exists SEE\ (boat(B_j)\ \&\ see(SEE)\ \&\ L_i \in SEE\ \&\ B_j \in SEE))$

The reason for this long expression is the need to reduce *3 lookouts saw 2 boats* to a logical expression on the satisfaction of predicates and structural descriptions by single individuals. One reason for doing this is that the variety of meta-description is minimized. One only needs the notions of existence (count(L)=3 is true), mutual exclusion (to define the logical connectives), and exhaustion ($\forall B_i$). This simplicity of the meta-notation is paid for in complexity of the expression.

We now suggest an alternative concept of satisfaction which produces simpler expressions but which (a) requires more meta-descriptive notions, and (b) has a fundamental tenant that structural descriptions are satisfied by "summaries" of the structure of the world. To be more explicit, let us take 2-count and 3-count to be meta-descriptive predicates and rewrite (14) as

(15a) $\underline{lookout}$(LOOKOUT)
(15b) $\underline{boat}$(BOAT)
(15c) $\underline{see}$(SEE)
(15d) $\underline{2\text{-}count}$(LOOKOUT)
(15e) $\underline{3\text{-}count}$(BOAT)
(15f) LOOKOUT S SEE
(15g) BOAT S SEE

We now consider the set of all individuals in the world which will satisfy (13). The meta-descriptive predicates 2-count and 3-count are then taken to mean that in this set there are 2 lookouts which satisfy LOOKOUT and 3 boats which satisfy BOAT. Note that on this account the relationships between individual lookouts and boats is normally not important. Only in special cases is this spelled out. Many examples will be given in Part II.

Perhaps some more examples will help the reader appreciate this approach. A sentence like *The boys hit the girls* represents a statement whose truth would be supported by hit conditions existing between individual boys and individual girls. Indeed *Have the boys ever hit the girls?* requires a reporting of even a single instance. However, since we have no information about what boy/girl pairs are involved, there seems to be no sensible way to interpret this statement in terms of conditions on individuals without making arbitrary choices. Were all the boys and girls involved? Was any girl hit by more than one boy? We don't know. So while *the boys hit the girls* implies hit relationships between individual boys and girls, there is nothing to be gained by expressing it in these terms since none of the particulars about individual hit relationships are known.

Now the important thing about *the boys hit the girls* may be the relationship between the groups, not the singling out of the individuals actually involved. Bierwisch [4] supports this notion with sentences like

(16a) The whites oppress the negroes.
(16b) The Romans destroyed Carthage.
(16c) The Chinese of the seventh century knew porcelain.

In all these sentences we have actions or attitudes by all the members of the group which may support those actually involved. Care must be taken or this group sense may be lost in a representation focused on individuals. For example, a typical method of representing *the boys hit the girls* in terms of individual <u>hittings</u> would be to form a predicate calculus expression reading:

(17) let B be a subset of the boys and G be a subset of the girls then
for all members b of B there exists a girl g in G such that b hit g
and for all members g of G there exists a boy b in B such that b hit g.

In this notation the importance of the larger sets is lost. The notion that the larger set <u>the</u> <u>boys</u> was responsible for or supportive of the action of the subset, B, is gone. This could be stated separately or recovered by inference, but we take the position that it is counter-productive to disaggregate the information in the first place. No double statement or inference should be required.

2.12 Concluding Remarks

In the introduction to this chapter it was suggested that increased use of meta-description can lead to a simpler notation. In the last section an example was given involving the relationship of a structural description to the world. In the next chapter we will see the use of meta-description to specify the relation of one description to another.

2.13 Exercises

1. Using only *ancestor* as a structural primitive it is proposed to set up a structural description of the way people are related to each other such that there will be only one way that the people in the real world can satisfy this description. Is this possible?

2. The positive integers can be given a structural definition by defining a set of structures such that each structure has a unique structurally defined successor and such that there is a structure corresponding to zero. Can you think of a way to do this? Can you think of more than one way to do it? (Do you think it makes sense to form the structural description of a number and say that all the individuals which meet that structural description are instances of that number? If not, do you think that the concept of a given number lacks an extension in the real world?)

3. Show a way of forming a structure corresponding to an ordered tuple such that there will be a one to one correspondence between ordered triples and your structures.

4. In section 2.4, it was proposed that a match should match both (a) the structures of the two individuals, and (b) the structures containing the two individuals and their component structure.

   i. How will a matching algorithm know that it can stop tracing down structure links, or is it necessary to consider all the links in matching two individuals?

   ii. Is there a way a matching algorithm might get into an infinite loop?

   iii. Describe a matching procedure.

3. Intensions

3.1 Introduction

By semantics one understands the question of whether the descriptions in a representation of knowledge are faithful to the real world. Since any system of knowledge representation usually provides a method of combining descriptions to form new ones, a potentially infinite number of descriptions can be generated from those currently existing in a network. There is the question of whether those generated descriptions will also be faithful to the world.

The most popular approach to this problem stems from Tarski [101]. His method is based on using two kinds of rules:

(18a) Rules to verify that some base descriptions correspond to the world.
(18b) Rules for verifying the composition of descriptions given that the individual descriptions can be verified.

This approach leads naturally to the idea that concepts like DOG should be verified in terms of rules involving sense data. The sense data will somehow provide the set of primitive terms used to build up the definition of concepts.

By contrast, we suggest that the focus for the representation of knowledge in computers should be on verifying that the entire semantic network represents some portion of the world, taken as as a single structure. There are several reasons for preferring this holistic approach in the context of semantic networks.

First, consider the fact that the typical computer system employing a semantic network suffers more sensory deprivation than a blind, deaf person. It sends and receives only character strings. The question of concept verification in terms of sensory primitives is not currently relevant. However, suppose we changed the name of every node and link in a very detailed semantic network to a number. We postulate that it would still be possible to determine if that network was a faithful model of a portion of the world, since only one portion of the world, if any, would have the structure being described by that particular network. We would do this by comparing the computer's model with our own. The relevant process in semantic networks is the verification of the computers model of the world by comparing its structure with the structure of our own model.

There are other factors which contribute to an interest in the holistic approach. One is a certain pessimism about ever getting a model of the world which is sufficiently accurate to permit confidence to be placed in deductions of any complexity. Certainly this can be done in specialized areas like mathematics. But in general it seems necessary to operate with very incomplete and inaccurate data. The basic thinking process would center more on induction from structural descriptions than on deduction from a set of logically consistent facts. There is a question of whether the difficulties of prediction faced, to take an extreme example, by economists, can ever be reduced to deduction from premises. Instead one would always be making holistic comparisons with the past.

A second factor is that some concepts like ELECTRON, BACHELOR, and SLAVERY seem inherently to rest on their relationship to other concepts rather than their definition in terms of sense primitives (e.g., Schank's Conceptual Dependency framework). The thesis that a primary mode of thinking is the expansion of concepts into their definitions has never made much headway. For example, Schank's [94] moves in this direction lead to a combinatorial explosion of irrelevant inferences. Authors such as Minsky [71], Bobrow and Winograd [7], and Fodor [27] have taken positions against such an expansion. We believe that is is often not the definition of a word like long that matters in dealing with a long x, but rather the implications of an x being long. For example, a long train needs several engines and must be waited for longer at crossings while a long snake is often more dangerous to man. In summary, it seems that many of our concepts are linked to others in what Quine [84] has called the web of belief.

What are the consequences of such a view of knowledge representation? The point is perhaps subtle, but the result will be an emphasis on computation in terms of intensions rather than extensions. By the extension of a structural description is meant the set of individuals which satisfy it. By the intension is meant the description itself.

In extrinsically oriented computation one can substitute one description for another if they have the same extension. For example, one could always substitute daughter for woman since every daughter is a woman and every woman is a daughter. In intensionally oriented computation such a substitution could not be freely made. The intent of the above remarks is to argue for an intensionally oriented form of computation—and knowledge representation.

As will be shown in the next two sections, English has many constructions which are naturally represented using intensions. Attempts to get away from intensions have resulted in the same kind of additional complexity that accompanied a sparse use of meta-description to represent the relation between descriptions and the world discussed in Chapter 2. *What then is the most natural form for an intensional representation?*

3.2 Opaque Predicates in English

Suppose we have a fellow with two names *Giorgione* and *Barbarelli.* We then learn the fact that

(19) Giorgione was so-called because of his size.

Now since *Giorgione* and *Barbarelli* both name the same person, we might try to substitute *Barbarelli* for *Giorgione* in (19), but this yields a falsehood. The problem is that the predicate *so-called* describes a relationship between the individual and the name. A predicate that blocks substitution in one of its arguments is sometimes said to set up an opaque context. You can't go into that argument position and make substitutions without changing the meaning of the whole expression.

Next suppose that John is a basketball player and at one of his games someone points to him and says

(20) That basketball player is short.

Now since that basket ball player is John we might try to substitute *John* for *that basketball player* in (20). This gives *John is short*, which may not be true. All we know is that he is short <u>for</u> <u>a</u> <u>basketball</u> <u>player</u>. Words like short always make a comparison to a peer group. The peer group can be explicitly mentioned with a <u>for</u> phrase e.g., John is tall for a midget. If the peer group is not mentioned, then it is deduced from the way the subject has been described. He is tall when viewed as a midget, perhaps not, when viewed as a person.

Another form of this example is seen in the following dialog:

(21) Betsy crossed the English Channel in 12 hours.
That was slow.
But she swam.
Oh, that was fast.

The same event was both a slow crossing and a fast swimming.

Another class of examples has been called hedges. When one says *Esther Williams is a regular fish*, he doesn't mean that she is a fish who is regular. *Regular*, as used here, means that she has some of the functions of its argument, presumably swimming well. Just how hedges like regular should be described is a topic for research, but it is clear that it creates a new description by using some parts of the description which is its argument. While *regular* preserves function, *fake* indicates that something is functionally deficient in an important way. A fake plant is not a plant which is fake; in some important functional sense it is not a plant at all.

As a final example, compare the intensional vs. the extensional senses of *and* in *The miner and sapper went to work*. This sentence can refer to either one person or two, depending on whether *and* is taken to conjoin *miner* and *sapper* or what they denote.

3.3 Is 'be' Opaque

Samuel Clemens wrote under the name Mark Twain. Samuel Clemens and Mark Twain are both names for the same individual. If *is* expresses the identity of denotations, then *Mark Twain is Samuel Clemens* and *Mark Twain is Mark Twain* should means the same thing. But we don't feel that they do, so this interpretation of *is* misses the way people actually use it. Another famous example is

(22a) The Morning Star is the Evening Star.
(22b) The Morning Star is the Morning Star.

3.4 The Referential/Attributive Distinction

The above examples show pretty convincingly that English has certain predicates which create opaque contexts. Looked at in programming language terms, and temporarily drawing an analog between denotation and evaluation, predicates like *so-called* need to be passed their arguments <u>unevaluated</u>. In a programming language like LISP, the programmer declares for each function whether the arguments are to be evaluated or not. If *so-called* were modeled as a LISP function, it would be declared to take an unevaluated argument.

This much seems straight-forward, but unfortunately it appears that English sentences often allow two interpretations according as to whether an argument to an arbitrary predicate is quoted and evaluated by the predicates or evaluated when it is passed in.

This was first pointed out by Donnellan [19]. Explaining his distinction, Donnellan says:

> I will call the two uses of definite descriptions I have in mind the attributive use and the referential use. A speaker who uses a definite description attributively in an assertion states something about whoever or whatever is the so-and-so. A speaker who uses a definite description referentially in an assertion, on the other hand, uses the description to enable his audience to pick out whom or what he is talking about and states something about that person or thing. In the first case the definite description might be said to occur essentially, for the speaker wishes to assert something about whatever or whoever fits that description; but in the referential use the definite description is merely one tool for doing a certain job — calling attention to a person or thing — and in general any other device for doing the same job, another description or a name, would do as well. In the attributive use, the attribute of being the so-and-so is all important, while it is not in the referential use.
>
> To illustrate this distinction, in the case of a single sentence, consider the sentence, 'Smith's murderer is insane.' Suppose first that we come upon poor Smith foully murdered. From the brutal manner of the killing and the fact that Smith was the most lovable person in the world, we might exclaim, 'Smith's murderer is insane.' I will assume, to make it a simpler case, that in a quite ordinary sense we do not know who murdered Smith (though this is not in the end essential to the case). This I shall say, is an attributive use of the definite description.
>
> The contrast with such a use of the sentence is one of those situations in which we expect and intend our audience to realize whom we have in mind when we speak of Smith's murderer and, most importantly, to know that it is this person about whom we are going to say something.
>
> For example, suppose that Jones has been charged with Smith's murder and has been placed on trial. Imagine that there is a discussion of Jones's odd behavior at his trial. We might sum up our impression of his behavior by saying, 'Smith's murderer is insane.' If someone asks to whom we are referring, by using this description, the answer here is 'Jones.' This, I shall say, is a referential use of the definite description.
>
> That these two uses of the definite description in the same sentence are really quite different can perhaps best be brought out by considering the consequences of the assumption that Smith had no murderer (for example, he in fact committed suicide). In both situations, in using the definite description 'Smith's murderer,' the speaker in some sense presupposes or implies that there is a murderer. But when we hypothesize that the presupposition or implication is false, there are different for the two uses. In both cases we have used the predicate 'is insane,' but in the first case, if there is no murderer, there is no person of whom it could be correctly said that we attributed insanity to him. Such a person could be identified (correctly) only in case someone fitted the description used. But in the second case, where the definite description is simply a means of identifying the person we want to talk about, it is quite possible for the correct identification to be made even though no one fits the description we used. We were speaking about Jones even though he is not in fact Smith's murderer and, in the circumstances imagined, it was his behavior we were commenting upon. Jones might, for example, accuse us of saying false things of him in calling him insane and it would be no defense, I should think, that our description, 'the murderer of Smith,' failed to fit him.
>
> It is, moreover, perfectly possible for our audience to know to whom we refer, in the second situation, even though they do not share our presupposition. A person hearing our comment in the context imagined might know we are talking about Jones even though he does not think Jones guilty.

To recapitulate, attributive use asserts something about whoever or whatever is the so-and-so. Referential use enables the user to pick out whom or what is being talked about. The distinction is brought out nicely by the two sentences:

(23a) The President has been married since 1945. *referential*
(23b) The President has lived in the White House since 1800. *attributive*

Sentence (23a) refers to the person who is currently President, while sentence (23b) says that since 1800 it has been true of whoever was President that he lived in the White House. Both of these sentences have both attributive and referential readings, but in each case the listener can reject one reading as counterfactual.

Donnellan limited his discussion to definite noun phrases, but this is probably an unnecessary distinction. The definite article *the* is used in references to boundable entities which the listener already has in mind. By contrast, the indefinite article *a* is used to introduce an entity to the listener. The distinction between referential and attributive use made for definite noun phrases can also be made for indefinite noun phrases. Thus in parallel with (23a-b) we have

(24a)	A member of the House has been married since 1945.	*referential*
(24b)	A white man has occupied the White House since 1800.	*attributive*

This distinction says that any predicate may be said either (a) to apply to a certain individual or (b) to apply to who-or-whatever meets a given description. What are the ramifications of this? It shows that for any predicate an argument position may either be taken as opaque or not. It calls into question the whole idea of making substitution under extensional identity the standard mode of operation.

3.5 Interpretations of 'be'

The common sense view of a proper name holds that the meaning of a proper name just is an individual. If *be* is taken to mean identity of individuals, then *Mark Twain is Samuel Clemens* is no different from *Mark Twain is Mark Twain.* Another trouble with combining this interpretation of *be* with the common sense view of proper names is that *Pegasus is non-existent* seems to refer to an individual and then deny it exists.

To avoid these difficulties, Russell [91] proposed treating *Mark Twain, Samuel Clemens,* and *Pegasus* as descriptions. That is, names are just a short hand for an identifying description of the individual named. One could clear away the confusion of English by rendering the above sentence *Mark Twain is Samuel Clemens* as

(25) $\exists x\,(\underline{\text{Mark-Twain}}(x) \,\&\, \underline{\text{Samuel-Clemens}}(x) \,\&\, \forall y\,(\underline{\text{Mark-Twain}}(y) \supset x = y))$

and the sentence *Pegasus is a winged horse* as

(26) $\exists x\,(\underline{\text{Pegasus}}(x) \,\&\, \underline{\text{non-existent}}(x) \,\&\, \forall y\,(\underline{\text{Pegasus}}(y) \supset x = y))$

These propositions are obviously true just in case there is a single object which fits both these descriptions. This solution allows statements about non-existent individuals to be meaningful, but is has the unintuitive result that all such statements are false. For example *Pegasus is a winged horse* becomes

(27) $\exists x\,(\underline{\text{Pegasus}}(x) \,\&\, \underline{\text{winged-horse}}(x) \,\&\, \forall y\,(\underline{\text{Pegasus}}(y) \supset x = y))$

which is false if Pegasus is taken to be non-existent. This comes about because while Russell uses descriptions, he does not use intensions, so the only way to show a relationship between descriptions is to show that they apply to the same individual. If there is no such individual, the solution doesn't work so well.

An alternative to Russell's approach is to question the interpretation of *be* as identity of individuals and *existent* as a predicate on individuals. Let us introduce a meta-descriptive relation, #c. We will read X#cY as Y characterizes X. This relation is defined as follows

X#cY	whatever satisfies X, satisfies Y
x#cY	the individual denoted by *x* satisfies Y
X#c*y*	only the individual denoted by *y* satisfies X
x#c*y*	the individual denoted by *x* is identical to the individual denoted by *y*.

The sentence *Mark Twain is Samuel Clemens* can now be understood as asserting either that the names are co-descriptors (e.g., MARK-TWAIN #c SAMUEL-CLEMENS) or that the entities are equivalent *Mark-Twain* #c *Samuel-Clemens.* In contrast, the sentence *Pegasus is non-existent* can be rendered: <u>not</u>(PEGASUS) The sense in which Pegasus doesn't exist will be discussed further below. One feels that existence in story and song should count as existence in some sense.

Use of #c like a promising direction, but #c will be of limited use as long as it is only a relation on constants and variables. We also need to relate structural descriptions.

3.6 Roles

In order to relate descriptions we need some way of referring to a description. In predicate calculus a description is a single <u>expression</u>, a linear string of symbols. This expression can serve as a piece of larger expression. Thus X and Y serves as an argument in X and Y $\supset$ Z. Figuratively speaking, we could pick this expression up by its top level connective, $\supset$.

Now consider the structural description

(28) *Bob S love-1*
Mary S love-1

How shall we refer to this? One possibility would be to use conjunction, *Bob S love-1* & *Mary S love-1*, just as predicate calculus would do. This is not bad, but it

(29a) introduces another connective and makes the representation bigger
(29b) is the meta-description of two structural descriptions, not a structural description
(29c) has & rather than *love-1* as its top level connective.

A second alternative is to replace the structure relation, S, with an intensional relation #r, read *role in.* Our structural description becomes

(30) *Bob* #r *love-1*
Mary #r *love-1*

The relation tuple X #r Y is defined to mean that "X" is in the structural description of "Y" and is satisfied by two individuals *x* and *y* only if *x* is in the structure of *y*. In defining roles it is convenient to disallow cycles, so that no variable is a role in itself, or a role in a role in itself, etc. The notion of a role will become more meaningful as we put it to use below.

3.7 Specialization

We now have a way of creating structural descriptions, and by using #c we can relate them in terms of the individuals which will satisfy them. The decision to relate two descriptions by #c is made on pragmatic grounds, that it will be useful to know that anything which can be described in the one way can be described in the other. This knowledge may have been arrived at analytically, or by observing the world.

It is also useful to have a purely intensional way of relating descriptions to be able to say that the first is a further specification, a specialization, of the second. In some cases, a description B may be specialized to a singular description A. In this case we can say that A is an individuation of B. We will write the specialization relation as A #sp B, read A is a specialization of B.

3.8 Functions over Concepts

To describe an entity *a* meeting the description A, we can do any or all of the following:

(31a) Declare it has an entity *b* meeting a certain description B in its structure; B #ROLE-IN A.
(31b) Declare it is in the structure of an entity *b* meeting a certain description B; A #ROLE-IN B.
(31c) Explicitly Characterize:
- i) the entity, A #c C
- ii) an entity in its structure, B #r A, B #c C
- iii) an entity whose structure the given entity is in, A #r B, B #c C

(31d) Implicitly Characterize:
- i) an entity in the given entity's structure
- ii) an entity whose structure the given entity is in.

The distinction we have in mind between explicitly characterizing an entity and implicitly characterizing on an entity is that between *is-a* and *is* predication which goes back at least to Aristotle [1]. In English we see the distinction in:

(32a) Snow <u>is</u> white. *is*
(32b) White <u>is</u> <u>a</u> color. *is-a*

Note that these two statements do not imply that snow is a color. Our explanation is that *is-a* predication (which may be written without *a* e.g., Mark Twain is Samuel Clemens) implies correspondence of structural description while *is* predication is Russell's notion of a predicate application.

One may question whether two kinds of predication are needed or whether one can be reduced to the other. In fact, we said earlier that Russell popularized the use of just *is* predication in mathematics.

We have chosen to reduce *is* predication to *is-a* predication. In order to do this and to implement implicit characterization it is useful to introduce the notion of functions. For example, we will take TALL to be a function which maps the description, A, into the description TALL-A. The individuals which satisfy TALL-A comprise those individuals, *a* which are possible referents of A and whose *height* attribute lies in the upper range of the distribution of the *height* attributes of the peer group of *a*. That is, the function TALL derives from a state, *s*, which has a role filled by the *height* attribute of the individual, *a*, described by the argument, A, to TALL.

Constants, variables, descriptions, and functions will be referred to collectively as concepts. Descriptions may be used to refer; functions do not refer but map concepts into concepts. Functions may take either descriptions or other functions as arguments. For example, we might have the function, VERY, applied to the function, TALL, yielding the function, VERY-TALL; then VERY-TALL applied in turn to the description, BOY, yielding the description, VERY-TALL-BOY.

Since functions do not refer, it doesn't make sense to relate concepts which are functions by #CHARACTERIZATION. It does make sense, however, for concepts which are functions to have roles.

Functions may be classified by the way in which the mapping between their arguments and values is defined. First, we distinguish adjuncts, conjuncts, and disjuncts.

(33a) Adjuncts add to or modify a description - e.g., TALL.
(33b) Conjuncts produce an itemization which contains their argument as one element - e.g., ONLY. For example, ONLY might take THE-DOG into the itemization {THE-DOG, NOTHING-ELSE}.
(33c) Disjuncts produce an itemization which contains their argument as one element and a description of a condition on their argument as a second element, e.g., SURPRISINGLY. SURPRISINGLY might take HE-SNORES into the itemization {HE-SNORES, THAT-HE-SNORES-IS-SURPRISING}.

Adjuncts may be further subdivided into hedges and non-hedges. A hedge builds its output description from its input description. For example, in *John is a regular fish*, the hedge, *regular*, selects some part of the description of the function of a fish, but not the description of the form of a fish. In *alleged communist*, the hedge ALLEGED applied to COMMUNIST creates ONE-SAID-TO-BE-A-COMMUNIST. That is, ALLEGED is derived from a state and it uses its argument to construct an exemplar of a description of that state. An example of a non-hedge adjunct is TALL, discussed above. Non-hedge adjuncts are always derived from states.

Note that *old friend* can mean either *a friend who is old* or *someone with whom an old friendship* exists. From this we note that when an adjunct is applied to a concept like FRIEND, it may

(34a) be derived from a state on the referent of FRIEND
(34b) be derived from a state on the concept, FRIENDSHIP, on which the concept FRIEND is based

(in a way we will describe later).

3.9 Three Ways to Treat Attributes

The introduction of functions from concepts to concepts and the #ROLE-IN and #CHARACTER-IZATION relations provides a method of dealing with the intensional constructions of English. To get a better understanding of this approach, let's compare it with two other approaches: second order predicate calculus and Montague's intensional logic [74]. The example task will be to represent the color sense of the adjective *red.* Our comparison will be limited to a look at the fundamental way of describing the world from which the three approaches start. Our approach is based on

(35a) treating concepts as individuals, and
(35b) basing representation on structural descriptions.

For example, we say that for an apple to be red means that the structural description of the apple includes an individual *color-of-apple* which may be characterized as RED. In order to take this approach, we have to assume existence of individual attributes like the color of a particular apple. Such individuals are discussed further in the chapter on ontology. Given this conception of what it means to be a red apple, we are willing to define an intensional function, red (from descriptions to descriptions), which could be described as modifying its argument structural description to yield a structural description which has a red color as a role. The exact definition of these functions is difficult, because, for example, a red man is not the same color as a red apple. This problem will be temporarily postponed.

In predicate calculus, *red* is generally modeled as a function from individuals to the two truth values, true and false - that is, as a predicate. If red is taken to be such a predicate, there is no need to postulate the existence of individuals like *color-of-apple.* To indicate that an apple is red we just form the sentence <u>red</u> (apple).

In order to increase the expressive power of predicate calculus, first order predicate calculus is sometimes extended to second order. Second order predicate calculus allows not only variables satisfied by individuals, but also predicate variables. For example, we can state that two individuals are equal if and only if they satisfy the same predicates by writing $(X)(Y)((X=Y)\equiv(P)(P(X)\equiv P(Y)))$. There has never been much agreement on whether there are individuals in the world, e.g., attributes, which satisfy predicate variables. Some say a virtue of first order predicate calculus is that it avoids reference to attributes.

Fig. 3. Illustration of Role-IN and Characterization Links
<==<fig2-1.press<

Another possibility is that predicate variables are satisfied by some sort of intension, which is a function from individuals to truth values, or perhaps by the set of all individuals for which that predicate is true. None of the interpretations has found much favor, but the third method, Montague's intensional logic, does use these ideas.

Montague introduced an elegant formal semantics. Introducing his system, he said [74]:

It has been maintained that we need not tolerate such entities as pains, events, tasks, and obligations. They are indeed not required in connection with sentences like 'Jones has a pain', 'the event of the sun's rising occurred at eight', 'Jones performed at eight the task of lifting a stone', or 'Jones has the obligation to give Smith a horse', which can be paraphrased without reference to the entities in question — for instance, in the case of the second example, as 'the sun rose at eight'.

There are, however, other sentences that most of us on occasion accept and that entail the existence of such dubious epistemological, metaphysical, and ethical entities as pains, tasks, events, and obligations. I have in mind sentences like 'Jones just had a pain similar to one he had yesterday,' 'not all psychological events have physiological correlates,' 'God cannot perform every possible task,' and 'Jones has not discharged all his obligations.' We cannot lightly dismiss sentences like these; they play a conspicuous role in philosophy, perceptual psychology, and everyday discourse. It therefore appears desirable to investigate the nature of the entities in question, construct an exact and convenient language in which to speak of them, and analyze the pertinent notion of logical consequence. The last task would seem a necessary preliminary to the rational treatment of certain philosophical paradoxes.

It seems clear Montague proposes his system as a way of talking about these entities without being committed to their existence. Montague's idea, in our terms, was to think of knowledge about a property as a procedure for generating the set of items having that property rather than as a procedure for telling if an item had the property. Just like Russell, he modeled this knowledge with functions, but Montague's functions were from possible worlds to sets of individuals rather than from individuals to truth values.

To be more specific, let I be the set of all *possible worlds*; for each member i of I, let A_i be the set of *individuals existing in the possible world i*; and let U be the set of all *possible individuals*. Then U will include the union of all sets A_i for i in I, and may indeed coincide with it, though we need not explicitly impose such a limitation. A *property of individuals* is a function having I as its domain and subsets of U as its values. (If P is such a property and i a possible world, $P(i)$ is regarded as the set of possible individuals that partake of P in i. For example, the property of being red is the function that assigns to each possible world the set of possible individuals which in that world are red.) More generally, an *n-place predicate of individuals* is a function having I as its domain and of which the values are sets of n-place sequences of members of U. If P is a predicate of individuals and i is a possible world, we regard $P(i)$ as the *extension* of P in i; the extension of an n-place predicate will always be an n-place relation (in the extensional sense).

Of course, this formulation left Montague with the need to admit the existence of the set of possible worlds and the set of possible individuals. He flinched a bit on that. But the assumption of only two sets has a mathematical neatness which promises tractability.

One advantage of this approach is that *slowly* in *Mary swims slowly* can be represented as a function from properties (e.g., *swims*) to properties (e.g., *swims slowly*). Since all functions are represented as sets, quantification over properties only affirms, by Quine's principle [84], that to be is to be denoted by a variable, the existence of certain sets.

The representation of *slowly* as a function from *swims* to *swims slowly* shows that Montague uses functions from functions to functions and so do we. But because Montague starts out with functions from possible worlds to sets of individuals, a function from functions to functions is his typical method of representing a concept. In contrast, our typical representation is a description, less often we have a function from descriptions to descriptions, and occasionally we have a function from functions to functions.

3.10 Reasoning About Knowledge and Belief

In section 3.2, a number of examples were given where substitution into opaque contexts caused problems. Another source of such examples are verbs expressing someone's knowledge of or attitude toward a fact. For example, from the statements

(36a) knows(pat, combination(safe1))
(36b) combination(safe1) = "45-25-17"
(36c) combination(safe2) = "45-25-17"

one can derive knows(pat, combination(safe2)), which may not be true, by substitution of equal expressions. The standard way of viewing this problem is to say that knows is an opaque operator which thus blocks substitution in its second argument. In an effort to reinstitute substitution, McCarthy [65] suggests treating concepts as individuals distinct from the individuals they denote. Introducing Safe1 as the concept of safe1 and Combination as the concept of combination he writes

(37) Knows(Pat, Combination(Safe1))

to assert that Pat knows the combination of safe1. The previous trouble is then avoided by taking

(38) Combination(Safe1) $\neq$ Combination(Safe2)

which McCarthy feels to be reasonable, since we do not consider the concept of the combination of safe1 to be the same as the concept of the combination of safe2, even if the combinations themselves are the same. The relation between concepts and the objects they denote is given by the denotation function (or partial denotation function)

(39) Safe1 = Den(Safe1)

The functions combination and Combination are related in a way which may be called extensional

(40) $\forall$s (Combination(Den(s)) = Den(Combination(s)).

This relation allows us to find the denotation of Combination(Safe1) in order to open the safe. But since the knows predicate lacks this extensionality in its second argument, the undesired evaluation is not possible.

McCarthy's formulation is summed up in Figure 4. His treatment rests two plausible assumptions.

(41a) It makes sense to identify some object as the denotation of a concept. That is, the thing it stands for.
(41b) Just because two concepts denote the same thing doesn't mean they are the same object because they may have different intensional meaning.

Fig. 4. McCarthy's Safe Example
<= =<fig2-2.press<
Combination(Safe1) = concept of the combination of the safe

den

45-25-17

Safe1 denotes the concept of the safe

den

safe1 denotes the safe

In McCarthy's approach *Safe 1* is the name of the concept of the combination of the safe. If a concept had more than one name, then there would still be a question of whether one name could be substituted for the other. The question of a concept having more than one name is not completely silly, because some people would assert that the identical concept can be expressed in more than one language.

McCarthy's solution also postulates that concepts such as the concept of the combination of a safe have a natural denotation. Some people object to this.

It is possible to formulate things so that the substitution of equals does not even arise in this particular *safe* problem.

Let us take, COMBINATION-SAFE1 and 45-25-17 to be descriptions of the combination of safe1. Since 45-25-17 also describes the combination of safe2, 45-25-17 is not equal to COMBINATION-SAFE1, but is a CHARACTERIZATION of it. Similarly, 45-25-17 is a CHARACTERIZATION of COMBINATION-SAFE2. Thus the question of substitution of equals does not arise. A value may be defined which maps a description D into one of its characterizations. For example,

(42) VALUE (COMBINATION-SAFE1) = 45-25-17

This second formulation is summarized by Figure 5.

Fig. 5. McCarthy's Safe Example (Revised)
<= =<fig2-3.press<

COMBINATION-SAFE1 characterization 45-25-17

describes describes

the-combination-of-the-safe

VALUE { COMBINATION-SAFE1 } = 45-25-17

COMBINATION-SAFE1 and 45-25-17 don't denote the same individual and so the question of substitution doesn't come up, but what about expressions like Samuel-Clemens and Mark-Twain which do denote the same individual. Should we be able to compute *Pat knows Samuel Clemens wrote Tom Sawyer* from *Pat knows Mark Twain wrote Tom Sawyer*?

One question we have to ask is whether in the sentence *Pat knows Samuel Clemens wrote Tom Sawyer*, *Samuel Clemens* is Pat's terminology or ours. Perhaps Pat knows him as Mark Twain, but in making this statement we know it will be read by someone who knows him as Samuel Clemens, so we use that name. The difference is more clear in "Pat said 'Samuel Clemens wrote Tom Sawyer' " where *Samuel Clemens* is definitely Pat's term vs. *Pat said that Samuel Clemens wrote Tom Sawyer*, where we can't tell what term Pat used.

The necessary mechanisms for the attribution of terms to individuals will be introduced in the next chapter. Let us consider the situation where Samuel Clemens is Pat's term. Can Mark Twain be substituted for Samuel Clemens? This wouldn't make sense if Pat doesn't know that Samuel Clemens and Mark Twain are the same individual, but suppose he does. It then becomes a question of how far we want to go in doing Pat's thinking for him. We may conclude he knows something that he doesn't because it may be deduced from the knowledge he has by a chain of substitutions which he hasn't thought to make.

Observe that Montague's formulation runs into a difficulty of this sort. Recall that Montague defines a property of individuals to be a function from possible worlds to sets of individuals. P(i) is the extension of predicate, P, in world i. Thus if two predicates, like creature-with-a-kidney and creature-with-a-heart have the same extension in our world, but not in all possible worlds, then they are not the same. But if two predicates have the same extension in all possible worlds, then they are the same. This would presumably be the case for *3* and *the real cube root of 27*. So in Montague's scheme, these English expressions would be translated into the same predicate.

Montague uses this predicate as the intension in a context of belief or knowledge. Thus he would not distinguish between

(43a) Pat knows 3 is odd.
(43b) Pat knows the real cube root of 27 is odd.

This would seem to call into question whether opaque contexts can be modeled by using intensions based on necessity in all possible worlds.

In fact, without knowing Pat there wouldn't seem to be much we can say about how well he can handle his knowledge.

3.11 Exercises

1. Use #c and the variable WINGED-HORSE to represent *Pegasus is a winged horse.*

2. Why can't 'X be the intension of X in Montague's sense?

3. Do you think there is any difference in extension between *He <u>is</u> male* and *He <u>is</u> <u>a</u> male*? Can you describe

any difference in the intensions of these two sentences?

4. Structure and Context

4.1 Introduction

This chapter begins with an examination of descriptions of structure. There are three related ways of describing structures: frames, semantic nets, and sentences in predicate calculus. The issue of whether there are significant differences between these methods is examined. Examples of different structural descriptions are given.

These structural descriptions give us a way of describing the world as it currently is, but we still need a method of describing future or past situations, and hypothetic plans of action, without confusing these descriptions with our descriptions of the actual world. It seems natural to try some sort of indexing scheme. But efforts so far in this direction have run into problems.

We explore whether a description can be indexed by being a component of another structural description. This leads us to the suggestion that every singular description of an individual be taken relative to some structural decomposition. It also raises the issue of temporal parts.

4.2 Frames

Minsky [73] sketches the notion of a frame:

> Here is the essence of the theory: When one encounters a new situation (or makes a substantial change in one's view of the present problem) one selects from memory a structure called a Frame. This is a remembered framework to be adapted to fit reality by changing details as necessary.
>
> A frame is a data-structure for representing a stereotyped situation, like being 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 and data. Each terminal can specify conditions its assignments must meet. (The assignments themselves are usually smaller "sub-frames.") Simple conditions are specified by markers that might require a terminal assignment to be a person, an object of sufficient value, or a pointer to a sub-frame of a certain type. More complex conditions can specify relations among the things assigned to several terminals.

This is a very informal definition, but so far the use of frames seems to be more a commitment to certain strategies of representation than a question of particular technical details. At this general level, frames have been extremely popular in the artificial intelligence community.

What can we say about frames? Frames are structural descriptions; they are not functions for finding or identifying individuals. Frames are abstract; abstract enough to describe many different individuals. Frames provide access to several kinds of information about individuals which would satisfy the frame. They represent aggregations of information.

Frames, semantic nets, and predicate calculus expressions form a hierarchy of complexity of notation. In predicate calculus the basic unit of description is the expression, for example, X R Y. If one has three expressions such as X R W, X R Y, and X R Z there is no way to refer to them all but to build a larger expression containing them, e.g., ((X R W and X R Y) and X R Z). In semantic nets the basic unit of description is a node pair linked by a directed arc. Binary relations can be represented, for example, by assigning nodes variable names and arcs relation names. But note that this produces an aggregation of the information that was missing from the expressions. It opens the possibility of referring to all the expressions without introducing any "ands". Finally, as Minsky said, a frame can be though of as a network of nodes and relations, a yet bigger aggregate with more conventions possible.

In contrast to frames, semantic nets and predicate calculus do not have an inherent commitment to large aggregates of information. Now the way information is aggregated can be considered a matter of efficient implementation. Information stated in a disaggregated way can be aggregated on the basis of use in order to facilitate rapid retrieval. It is natural to ask whether there is any difference in what can be said with these three types of representation or whether they are just three different ways of implementing languages with the same expressive power. The key potential for difference apparently lies in meta-description. First, the three schemes may not allow the same sorts of meta-description. For example, using frames it is typical to allow the default specification of values for slots, while in standard predicate calculus there is no way to indicate that something is true in the absence of other information. This lack has led A/I researchers to explore so-called non-monotonic logic. In standard logic the addition of new information never makes something which was already true, false. In non-monotonic logic this can happen. A second difference is that frames may lead to the meta-description of larger aggregates of information. For example, the specification of what it takes to give a criterial description of something — to distinguish it from all others — may be easier using frames.

4.3 From Predicate Calculus to Frames

The discussion will start with standard first order predicate calculus and progress toward frame systems and semantic nets. We then consider how the use of intensions to enrich the semantic network representation.

In predicate calculus, the sentence *John gave Mary the book* might be represented as give(*John, Mary, book-1*). In this representation, the giving event is modeled as as property of the tuple: *John, Mary, book-1.* This property is detected by the predicate give. In a frames systems, one would typically denote the giving event directly. The relation of *John, Mary and book-1* to this event would then be specified. Thus the following predicate calculus form would be closer to a typical frames representation.

Fig. 6. Binary Relations
<= =<fig3-1.press<

(44) ∃x(x∈*giving-events* & agent (x, *John*) & recipient(x, *John*) & object(x, *book-1*))

By naming the giving event (skolemizing the existential variable) in the above formula we get

(45) *g-1*∈*giving-events* & agent(*g-1, John*) & recipient(*g-1, Mary*) & object(*g-1, book-1*)

The predicates agent, recipient, and object are known as predicate argument relations. They replace the ordered tuple as a way of distinguishing the different roles of *John, Mary*, and *book-1* in the giving event. In Part II we will attempt to choose a set of predicate argument relations which are related in a systematic way to syntactic structures. One advantage of this form over the predicate on a tuple which we started with is that it is easy to add additional description. A disadvantage is that it uses a bigger data structure to represent the same information. We trade storage space for flexibility.

It would also be possible to express the predicate argument relations as functions over the set, *giving-events*:

(46) *g-1*∈*giving-events* &
giver(*g-1*) = *John* &
recipient(*g-1*) = *Mary* &
object(*g-1*) = *book-1*

Using functions, *someone gave Mary the book* can be represented as

(47) *g-1*∈*giving-events* &
agent(*g-1*)∈*persons* &
recipient(*g-1*) = *Mary* &
object(*g-1*) = *book-1*

Now one point of using frames is that a higher level of notation can be defined. For example, the previous description could be abbreviated as:

g-1	self:	(member-of *giving-events*)
	agent:	(member-of *persons*)
	recipient:	*Mary*
	object:	*book-1*

Here *self* may be interpreted as an identity function used to increase the uniformity of the notation. Sometimes abbreviations are used just to make notation more readable, they are expanded out in the internal representation. If this is not done, the programs using the representation must be written to understand the abbreviated form. There will be a trade off between the size of the frame notation and the size of the interpreting programs.

Suppose we want to represent the above frame as a network of nodes and relations, i.e., as a semantic network. This could be done as in Figure 7, where solid nodes represent constants, hollow nodes represent variables, and links represent two place relations. It can be seen that this notation is slightly less compact than

Fig. 7. Representing a Frame as a Semantic Network
<= =<fig3-2.press<

the frame notation, but uses fewer conventions.

The above notation is based on extensions, and so we will get into trouble when we try to represent some of the examples in the last chapter. Consider, for example, that one event can be both a fast swimming and a slow crossing. Using the notation of Figure 7, an event which was both a crossing and a swimming could be represented as Figure 8. But in this representation there is no way to add the modifiers fast and slow except to create a different type of event-set for each modifier of each event type or else to create a different type of fast and slow predicates for each type of event they might be applied to: e.g., crossing-slow (*e-1*) and

Fig. 8. A Crossing and a Swimming
<= =<fig3-3.press<

Fig. 9. A Slow Crossing and a Fast Swimming
<= =<fig3-4.press<

swimming-fast (e-1). What this misses is the generalization about the peer group made in Chapter 3.

The reader might note that a similar conflict can occur between predicate argument relations. For example, the same event can be both a buying and a selling. The same person who is the recipient of the selling is the agent of the buying. As before, this can be resolved by creating the relations, buying-agent and selling-recipient.

One way to look at this problem is that two different descriptions are being merged because they describe the same individual. The merge creates ambiguities, which can be disambiguated by specialization of the predicates and relations involved.

An alternative approach is to work with intensions and keep the alternative descriptions separate. This is illustrated in Figure 10. In this figure the predicate argument relation, agent, has been replaced with an intensional one, #agent. It indicates that the relation it names is part of the structural description of its source node.

In this way of doing things the event node e-1 only serves to relate the description bearing nodes, CROSSING-1 and SWIMMING-1. A node like e-1 has been called an anchor. We can consider getting rid of the anchor node e-1 and using the equivalent structure shown in Figure 11. Whether this is a good idea isn't known. The decision is more complex if there are more than two characterizations of the anchor and if some of these characterizations are not singular.

Having got this far, it's time to take a closer look at what should be represented by a link.

4.4 What Should a Link Represent.

Links are used to represent binary relations. It has been shown that they may be given either intensional or extensional interpretations. The question of what binary relations should be represented by links deserves some further discussion.

Fig. 10. Illustration of Intensionality
<= =<fig3-5.press<

Fig. 11. Removing Anchor Nodes
<= =<fig3-6.press<

Whenever a binary relation exists between two individuals, several representational options are open to us. For example, suppose *x* believes *y*. This can be represented as a property of the tuple *x,y* which is detected by a predicate. Alternatively, it can be represented as an individual, *believe-1*, having predicate argument relations believer and belief. Further, we could define functions believers and beliefs such that believers (*y*) is the set of everyone believing *y*, and beliefs (*x*) is the set of everything believed by x, so that x∈believers(*y*) and *y*∈beliefs(*x*).

Turning to intensions, we can also define the roles believer and belief as shown in Figure 12. In this figure, representation a) does not commit us to individual beliefs, while b) and c) do. Representation b) will use a wide variety of link types while c) will not. Further, c) allows us to construct a frame to describe something by virtue of its being a belief, and then to specialize this description for a particular individual. As we go from a) to b) the amount of space required for the representation increases, but the number of predicate or link types used decreases, so the representation becomes nearly self contained.

Instead of going from a) to b) one can introduce a new type of link and get b' as shown in Figure 13. One point of doing this is to make the node *believe-1* optional, while still providing a way to describe the relation if need be. A second advantage of this representation is rapid access to y from x.

In summary, whenever we have a relation xRy we can take R to name a link in which case R cannot be directly described, or we can take R to be a node, in which case we have subsidiary relations between x and R and R and y. As we expand, the links are named with progressively more primitive relations, but the size and detail of the representation increases. We must choose the correct trade off between flexibility and efficiency.

4.5 Inheritance and the Individuation of Roles

Using notation introduced so far the structural description of an arch can be represented as shown in Figure 14. Fahlman [24] has pointed out an interesting question which arises in the interpretation of structures like that shown in Figure (47) the structural description of an arch. If this figure is a description of the structure of the generic arch, then it should be possible to put this description into correspondence with

Fig. 12. Progressively General Representations
<= =<fig3-7.press<

Fig. 13. Descriptions of Links
<= =<fig3-8.press<

the description of an individual arch, say ARCH-1. Figure 15 shows a fragment of Figure 14 extended to include the corresponding fragments of two individual arches, ARCH-1 and ARCH-2. The dotted links from R2-1 to R2 and R2-2 to R2 represent specialization links. In this case R2-1 and R2-2 inherit all the description of R2 except the role-in link from R2 to ARCH. In each case, this role-in link is overridden by

Fig. 14. Simplified Structural Description of an Arch
<= =<fig3-9.press<

Fig. 15. An Example of Individual Nodes
<= =<fig3-10.press<

the mode's own role-in link.

Specialization links specify the inheritance of all description which is not explicitly overridden. Characterization links indicate an alternative description. One might think that the specialization links are not needed in Figure 15, that they could be replaced with characterization links. However, this leads to a problem pointed out by Fahlman.

Suppose we were to say that since R2-1 is the top of an individual arch, ARCH-1, it may be described as R2, the top of an arch. That is, when one node is described as another, its roles are described as the corresponding roles of the other. Doing this gets one into difficulty. To see this note that we may ask "which individual block in the generic arch is the top", and the answer is R2. R2 is taken as an individual node and clearly it represents a single one of the three blocks comprising an arch. But now suppose that nodes ARCH-1 and ARCH-2 describe arches in the real world. Then R2-1 and R2-2 describe the tops of these arches, and since R2-1 and R2-2 may be described as R2, R2 describes the tops of these arches. Thus, it would seem that R2 does not, in fact, describe a single individual. The reader will no doubt be able to articulate to his satisfaction what is going on in this example. We hope to capture this by expressing and invoking a rather far-reaching principle of interpretation.

Accordingly, we will stipulate that every individual node R must be a role in some other node N. In addition, every entity r is taken to be an individual with respect to the entity n, where r and n are described by R and N respectively. The only exception is that the real world is not in the structure of anything.

One effect of this stipulation is to make the role-in link an essential part of any description of an individual. Now by using inheritance links instead of characterized links from R2-1 to R2 and from R2-2 to R2, we can override the role-in link from R2 to ARCH and thus preserve R2 from describing these individuals.

The essential difference between inheritance and alternative description is that an inheriting node can override a criterial part of an inherited description, so that the inherited description no longer picks out the same individuals. By contrast, alternative description preserves the integrity of individuals. It lets one get a different description of the same individual.

4.6 Making Every Individual Node a Role

This section will touch briefly on the philosophical ramifications of making every individual node a role. In the Tarskian [101] view of English semantics correspondence between descriptions and the world is described in terms of propositions. A proposition is true if it corresponds to the world and false otherwise. Propositions are either atomic or compound — formed by conjunction, etc. The truth or falsity of atomic propositions must be given. The truth or falsity of compound propositions is then determined from the truth or falsity of their components by applying rules of composition associated with their connectives.

In the theory proposed here, discussion of correspondence between descriptions and the world is not limited to propositions. We can discuss correspondence for any individual, proposition or not. However, the key difference is that since every description of an individual must include a description of the structure containing it, it doesn't make sense to ask for the truth or falsity of atomic propositions independently of the truth or falsity of the structures containing them. The view presented here shifts away from <u>composition</u> to a system which places more weight on <u>decomposition</u>. One starts with a node representing the world and structurally decomposes it in various ways. The parts of these decompositions are then further decomposed in turn to any desired depth. This produces a structural hierarchy of descriptions. This structural hierarchy is complemented by the abstraction hierarchy of descriptions produced by the characterization links.

The importance of this approach comes out if one believes that there are, in fact, individuals which have no description which does not depend criterially on their being part of a structure. This realization has caused some philosophers to deny that such individuals are entities at all. This view is put forward by Benacerraf [2] in debating the existence of numbers:

Therefore, numbers are not objects at all, because in giving the properties (that is, necessary and sufficient) of numbers you merely characterize an *abstract structure* — and the distinction lies in the fact that the "elements" of the structure have no properties other than those relating them to other "elements" of the same structure. If we identify an abstract structure with a system of relations (in intension, of course, or else with the set of all relations in extension isomorphic to a given system of relations), we get arithmetic elaborating the properties of the "less-than" relation, or of all systems of objects (that is, *concrete* structures) exhibiting that abstract structure. That a system of objects exhibits the structure of the integers implies that the elements of that system have some properties not dependent on structure. It must be possible to individuate those objects independently of the role they play in that structure. But this is precisely what cannot be done with the numbers. To *be* the number 3 is no more or less than to be preceded by 2, 1, and possibly 0, and to be followed by 4, 5, and so forth. And to *be* the number 4 is no more and no less than to be preceded by 3, 2, 1, and possibly 0, and to be followed by .... *Any* object can *play the role of* 3; that is, any object can be the third element in some progression. What is peculiar to 3 is that it defines that role — not by being a paradigm of any object which plays it, but by representing the relation that any third member of a progression bears to the rest of the progression.

Arithmetic is therefore the science that elaborates the abstract structure that all progressions have in common merely in virtue of begin progressions. It is not a science concerned with particular objects — the numbers. The search for which independently identifiable particular objects the numbers really are (sets? Julius Caesars?) is a misguided one.

On this view many things that puzzled us in this paper seem to fall into place. Why so many interpretations of number theory are possible without any being uniquely singled out becomes obvious. There is no unique set of objects that are the numbers. Number theory is the elaboration of the properties of *all* structures of the order type of the numbers. The number words do not have single referents. Furthermore, the reason identification of numbers with objects works wholesale but fails utterly object by object is the fact that the theory is elaborating an abstract structure and not the properties of independent individuals, any one of which could be characterized without reference to its relations to the rest. Only when we are considering a particular sequence as being, not the numbers, but *of the structure of the numbers* does the question of which element is, or rather *corresponds to*, 3 begin to make any sense.

While I agree with Benacerraf's analysis of numbers, I am unwilling to therefore deny that numbers are entities. In a semantic network it is simplest to have nodes which represent discriptions of entities and one needs nodes for anything which will be described, including numbers. Rather than deny that numbers are descriptions of entities one can put all descriptions on the same footing as numbers — require the description of any individual to include the structure in which that description is used. For some sorts of descriptions, like numbers, the structure in which they are used will dominate. Other descriptions will admit of complex internal structures.

4.7 Structure and Existence.

In the above discussion structural descriptions were created using the intensional relation, #role-in. If A is a role in B, then any individual satisfying A must be contained in the structure of an individual satisfying B. We speak of A as being <u>singular relative to B</u>, if we have stipulated that for any individual *b* satisfying B, there is at most one individual in the structure of *b* which will satisfy A.

These ideas are illustrated in Figure 16. Here, we continue the practice of representing singular descriptions with solid nodes. The constant *real-world* we stipulate to denote, and thus be satisfied by, and only by, the real world. The real world is separated from the network of nodes by a wavy line to indicate that, in a sense discussed below, we take its existence on faith. A node representing an individual person in the real world is shown to be a specialization of the typical person node and a role in the *real-world*. Similarly, the individual nose in the typical person is singular relative to the typical person. A further specialization of the

Fig. 16. Structure and Existence
<= =<fig3-11.press<

individual nose of a typical person is the individual nose of an individual person in the *real-world*. This node is singular relative to that individual person in the *real-world*.

The hypothesis made above is that every singular node is in fact only singular relative to another node, with the exception of the real world. Thus, to be an individual is either (a) the real world, or (b) individuated relative to some other individual. The distinction between the real world and other things was suggested by Carnap [12]:

Are there properties, classes, numbers, propositions? In order to understand more clearly the nature of these and related problems, it is above all necessary to recognize a fundamental distinction between two kinds of questions concerning the existence or reality of entities. If someone wishes to speak in his language about a new kind of entities, he has to introduce a system to new ways of speaking, subject to new rules; we shall call this procedure the construction of a linguistic *framework* for the new entities in question. And now we must distinguish two kinds of questions of existence: first, questions of the existence of certain entities of the new kind *within the framework*, we call them *internal questions*; and second, questions concerning the existence or reality *of the system of entities as a whole*, called *external questions*. Internal questions and possible answers to them are formulated with the help of the new forms of expressions. The answers may be found either by purely logical methods or by empirical methods, depending upon whether the framework is a logical or a factual one. An external question is of a problematic character which is in need of closer examination.

Let us consider as an example the simplest kind of entities dealt with in the everyday language: the spatio-temporally ordered system of observable things and events. Once we have accepted the thing language with its framework for things, we can raise and answer internal questions, e.g., "Is there a white piece of paper on my desk?", "Did King Arthur actually live?", "Are unicorns and centaurs real or merely imaginary?", and the like. These questions are to be answered by empirical investigations. Results of observations are evaluated according to certain rules as confirming or disconfirming evidence for possible answers. (This evaluation usually carried out, of course, as a matter of habit rather than a deliberate, rational procedure. But it is possible, in a rational reconstruction, to lay down explicit rules for the evaluation. This is one of the main tasks of a pure, as distinguishing from a psychological, epistemology.) The concept of reality occurring in these internal questions is an empirical, scientific, non- metaphysical concept. To recognize something as a real thing or event means to succeed in incorporating it into the system of things at a particular space-time position so that it fits together with the other things recognized as real, according to the rules of the framework.

From these questions we must distinguish the external question of the reality of the thing world itself. In contrast to the former questions, this question is raised neither by the man in the street nor by scientists, but only by philosophers. Realists give an affirmative answer, subjective idealists a negative one, and the controversy goes on for centuries without ever being solved. And it cannot be solved because it is framed in the wrong way. To be real in the scientific sense means to be an element of the system; hence this concept cannot be meaningfully applied to the system itself. Those who raise the question of the reality of the thing world itself have perhaps in mind not a theoretical question as their formulation

seems to suggest, but rather a practical question, a matter of a practical decision concerning the structure of our language. We have to make the choice whether or not to accept and use the forms of expression in the framework in question.

We adopt Carnap's idea that existence of individuals is a matter of the internal consistency of a representation of knowledge, and go beyond this to postulate that any description of an individual must place it in the structure of another individual. This approach gives a natural way of describing the status of various non-existent entities, like unicorns. The problem of non-existent entities is discussed by Moore [75]

How can a thing 'appear' or be 'thought of' unless it is there to appear or be thought of? To say that it appears or is thought of, and yet that there is no such thing, is plainly self-contradictory. A thing cannot have a property unless it is there to have it... When I think of a unicorn, what I am thinking of is certainly not nothing; if it were nothing then, when I think of a griffin, I should be thinking of nothing and there would be no difference between thinking of a griffin and thinking of a unicorn. But there certainly is a difference; and what can the difference be except that in one case I am thinking of a unicorn, and in the other a griffin? And if a unicorn is what I am thinking of then certainly there must be a unicorn, in spite of the fact that unicorns are unreal. In other words, though in one sense of the word there certainly <u>are</u> no unicorns - the sense, namely, which to assert that there are would be equivalent to asserting that unicorns are real - yet there <u>must</u> be <u>some</u> other sense in which there are such things; since, if there were not, we could not think of them.

One view is that things like unicorns, <u>subsist</u> rather than exist. As Eaton [22, p. 152] says:

Subsistence is a kind of being to which existence may be added, but to which existence is not necessary. In the realm of subsistence, the lion and the unicorn lie down together. To think of a lion or of a unicorn is to think of entities that partake equally of this impartial being, this thinner reality, which includes the possible and the imaginary as well as the real.

Eaton goes on to suggest that <u>subsistence</u> can be interpreted to mean that we have procedures which we can and do apply to check for existence. That we can give a procedural interpretation to the notion of subsistence.

The importance of this issue in semantic networks arises from the simplicity which would derive from equating the existence of a description in a semantic network with the existence of the entity it describes. This possibility is ruled out by the need to represent hypotheticals like "a blue yo-yo" in "John told a story about a blue yo-yo". The representation of this statement will contain the description corresponding to "a blue yo-yo", but it would be wrong to conclude from this statement that a blue yo-yo exists.

On the other hand, if we require that an identifying description of any individual include a description of the structure containing the individual, then we may equate the existence of such a description with the existence of the individual in that structure. An individual is said to exist in the normal sense when it exists in the real-world. This is what the unicorn can't do. There is nothing to keep us from putting a unicorn concept in real-world, but when one applies procedures to verify the internal consistency of this they fail. The unicorn subsists in Eaton's sense, but the real world unicorn must be marked non-existent at the meta level to avoid further attempts at verification.

Re-examining "John told a story about a blue yo-yo" in this light, we would describe a blue yo-yo as being part of the structure of a story entity. Understanding the nature of the existence of the blue yo-yo then becomes a matter of understanding the nature of stories.

4.8 Describing Structures.

It may be instructive to apply some of the concepts developed thus far to the description of the structure of an arch like that shown in Figure 17. This arch is chosen because it is a familiar example in the artificial intelligence community. The Oxford English dictionary claims that senses of *arch* derive historically from *a part of a circle* and the above is perhaps not even a typical arch, let alone the general form of an arch. So we are not defining the English concept arch here, just a particular exemplar. Such an arch has more than one structural decomposition:

(48a) Structural Members: A, B, and C are blocks and D is a space.
(48b) Positional: B is the top and A and C are sides; D is the under-region of the arch.
(48c) Functional: A and C are supports and B is the supportee.

Compare the Winston arch with the McDonald's Golden Arch shown in Figure 18. The Golden Arch has only one structural member and so the sides and top of this arch correspond to parts of a structural member rather than distinct structural members. This illustrates how different structural decompositions sometimes cast the same individual in several roles, but sometimes describe different individuals.

Fig. 17. Winston Arch
<= =<fig3-12.press<

Fig. 18. McDonald's Golden Arch
<= =<fig3-13.press<

A partial description of a typical Winston Arch is shown in Figure 19. This figure will serve to illustrate several issues. The ARCH has only one TOP, and so being a TOP in the structure of an arch is an individuating description. An individual satisfying this TOP description will also satisfy BLOCK-1. We wish to say of this individual that it is lying and that it is supported by the sides. There is a decision to face in making these statements.

We must decided for each of the processes, SUPPORT and LIE, whether we should use the generic concept, e.g., SUPPORT, the concept for a generic role in ARCH, or the concept for a particular individual in ARCH. While it is useful to define generic roles, — for example, the wheel of a car — these make no statement about the existence of individuals in the structure described. There must be at least one description of each individual which represents that individual as such, either implicitly or explicitly.

An example will help explain how an individual can be implicitly represented. Note that, a person has one and only one father. Thus given a person, the father is uniquely determined. On the other hand, given a father, the son is not uniquely determined. We will say that in a father/son relation, the son implicitly individuates the father, but not the reverse.

In Figure 19, BLOCK-1 is represented explicitly as a singular description. But since we have nothing in particular to say about the LIE in which BLOCK-1 is engaged, there is no need to represent it explicitly. We just indicate that BLOCK-1 may be described as the subject of the generic LIE. However, if in fact LIE is individuated by its subject, then we have implicitly represented an individual LIE.

Fig. 19. Partial Description of Winston Arch
<==<fig3-14.press<

Consider next that each side of the arch supports the top. It won't do just to say that a side is the subject of a support, we want to say that this support is the support which supports the top. In this case we need a specialized support.

Observe next, that since BLOCK-1 and TOP-1 are mutual characterizations of each other, it is not clear which one (or both) should be characterized as the subject of LIE. It is recognized that alternative aggregations of descriptions are possible, but no one has a good idea of how this should be done.

4.9 Contexts.

Instead of appealing to structure, one can appeal to context as a way of distinguishing real individuals from hypothetical ones. The dictionary defines context as "the interrelated conditions in which something exists or occurs." This seems like a rather vague notion on which to base a representation of knowledge, so the practice in artificial intelligence has been to define a set of intentional contexts or partitions which do not necessarily have any relation to the real world. For example, Figure 20 shows how Hendrix [38] would represent the fact that legal persons can own physical objects. In this figure rectangles (including the square) represent partitions, ovals represent sets, and circles represent individuals. An individual or set is in a partition if its representing circle or oval is in the rectangle representing that partition. In this figure the individual, I, is an implication that any individual owning, X, has an individual agent, Y, which is a legal person, an individual object, Z, which is a physical object, an individual start-time, t_1, which is a time, and an individual end-time, t_2, which is also a time.

This same information is shown in Figure 21, using the notation presented here. The key differences between the representations in Figures 20 and 21 are:

a) i) Figure 20 shows the set of ownings. To indicate properties which must be true of every member of this set, an individual implication I is created. I is related to OWNINGS by the link, delineation. The antecedent of I is a partition containing an individual, X. By convention this individual, X, in the antecedent of I is the typical member of ownings. The individuals in the structural description of X are in the partition which is the consequent of I.

ii) In Figure 21, the node, owning, represents the typical member. As a generic representing the typical owning it does not need to be contextually constrained. Thus, it is not taken to be a role in any other node. Part of the complexity of Figure 20 is avoided by making the typical member a primitive notion (generic node) rather than constructing it as an individual which exists only in an abstract space (the antecedent and consequent of an implication).

b) i) In Figure 20 the individuals Y, Z, t_1, and t_2 which are in the structural description of X are given context by being placed in the partition which is the consequent of I.

ii) In Figure 21 the individuals Y, Z, t_1, and t_2 which are in the structural description of X are given context by being made roles in X itself. Thus a typical member serves as the context of its own structural description.

Fig. 20. The Delineation Theorem of Ownings
<==<fig3-15.press<

Fig. 21. The Delineation Theorem of Ownings (Revised)
<= =<fig3-16.press<

---

c) In Figure 20, the individuals X, Z, t_1 and t_2 are related to X by links labeled with "case relation names" or "slot names". Thus, as mentioned above, a distinction is made between nodes and slots. By contrast in Figure 21, agt, obj, start-time, and end-time are themselves nodes and can be described just like any other node. It is perhaps worth pointing out that the logical organization of Figure 21 does not prevent memory from being physically organized so that given <u>owning</u> and <u>agt</u> one can retrieve node Y at least as quickly as it can be done with the organization of Figure 20.

In summary, Figure 20 takes as primary the notion of <u>context</u> and implements it by placing nodes in <u>partitions</u>. Figure 21 takes as primary the notion of <u>structure</u> and implements it by making a node a <u>role</u> in the structural description of another node.

The above example shows that context and structure are in some ways competing rather than complementary notions. If a system provides both there should be a clear understanding of what the intended uses are.

A possible advantage of contexts over structure is that a context doesn't have to correspond to anything in the real world. There is more freedom in its use — sometimes contexts will correspond to real world entities and sometimes they won't. Contexts provide the maximum flexibility in creating a filing scheme for our knowledge. Unfortunately, setting up a filing scheme for knowledge that changes over time runs into such severe problems that it calls into question whether the mechanism of the filing scheme itself, for example, a context mechanism, will be of any particular help. The difficulties we refer to are known as the frame problem.

4.10 Situations and the Frame Problem.

A standard device for dealing with hypothetical or time varying knowledge is the introduction of situation variables, or possible worlds. For example, Figure 22 shows two configurations of blocks which might exist at different points in time. These are labeled as situations s_1 and s_2. The content of Figure 22 can be described by using predicates which have both blocks and situations as arguments. For example,

(49a) $\underline{\text{on}}\,(a, \textit{table}, s_1)$
(49b) $\underline{\text{on}}\,(b, a, s_1)$
(49c) $\underline{\text{on}}\,(c, \textit{table}, s_1)$
(49d) $\underline{\text{on}}\,(c, b, s_2)$
(49e) $\underline{\text{on}}\,(b, a, s_2)$
(49f) $\underline{\text{on}}\,(a, \textit{table}, s_2)$

Given situation s_1, the event of moving *c* on to *b* will produce situation s_2. This can be expressed using a function <u>result</u>, result(move(c, b), s_1)$=s_2$. The function result takes two arguments, a description of an event and a situation. Its value is the resulting situation.

Using this notation, a plan can be expressed as a series of situations resulting from previous ones. For example, starting with all the blocks on the table a plan to reach situation s_2 would be

(50a) $\underline{\text{on}}\,(a, \textit{table}, s_0)$
(50b) $\underline{\text{on}}\,(b, \textit{table}, s_0)$
(50c) $\underline{\text{on}}\,(c, \textit{table}, s_0)$
(50d) s_2 = result(move(c,b), result(move(b, a), s_0))

Fig. 22. Two Situations in the Blocks World
<==<fig3-17.press<

In order to prove that the results of the two moves will be s_2, the machine needs the general axiom:

(51) on(x, y, result(move(x, y), s)

This axiom says that if x is moved onto y in any situation, x is on y in the resulting situation. But this one axiom is not enough! The machine also needs to know that the on relations of blocks not moved remain intact. It needs an axiom like:

(52) $w \neq x$ & on(w, z, s) $\supset$ on(w, z, result(move(x, y), s))

Here we have the essence of the frame problem. It's not enough to specify what changes as the result of an event; we must also specify what remains the same. There is no a priori justification for inferring any properties of result (e, s) from those of s. If it were usually the case that events made widespread and drastic alterations to the world (explosions, the Second Coming, etc.), then we could hardly expect anything better than the use of axioms to describe in detail, for each event, exactly what changes it does or does not bring about. The hope of a more general solution is based on the fact that the world is, fortunately, fairly stable. Most events make only small local changes in the world and are not expected to touch off long chains of cause and effect.

Hayes [37] describes five general schemes for implementing solutions to the frame problem:

(53a) Frames: McCarthy and Hayes [66] suggested partitioning the facts in a situation into mutually exclusive categories, so that each event will only change the facts in one or a few categories. For example, moving a block doesn't change the color of any block.
(53b) Causal Connection Proved Not to Exist: Hayes [37] suggests that all facts be moved forward for which it can be proved that the given event doesn't change them.
(53c) Causal Connection Not Proved to Exist: MICRO-PLANNER [100] moves forward all facts which it can not prove have been changed.
(53d) STRIPS [26]: With each event we list the facts which are not to be moved forward.
(53e) Move Unless It Makes the Resulting Situation Inconsistent

Rescher [85, 86] suggests moving facts until the resulting situation becomes inconsistent. The big problem with this is that consistency is very expensive to maintain and check. A second problem is that there are many ways of getting an inconsistent set, so one needs some way to choose the order in which facts will be moved.

Although these proposals have been known for a decade, no simple way of implementing them for specific problems has ever been found. Importantly, the requirement to understand causality and logical implication in order to know what facts can be carried forward has led to the realization that a representation of knowledge should perhaps be organized around these relationships rather than around situations.

4.11 Context Hierarchies.

In an effort to facter situations into changing and constant parts, CONNIVER introduced context hierarchies. The idea is illustrated in Figure 23. In context s_0, all three blocks are on the table. Context s_2 is represented in terms of additions and deletions to s_0. Similarly s_2 is represented by additions and deletions to s_1. In CONNIVER, a new context was generally formed when a subroutine was called. Thus, the resulting hierarchy of contexts mirrored the hierarchy of subroutine calls. The original idea was that when a subroutine returned, the system would also return to the higher context, but of course this meant that information stored in the lower context was no longer available even though it might still be true. Once a decision was made to avoid this by placing information sometimes in a higher, sometimes in a lower context, the "automatic indexing" aspect of the context mechanism began to be vitiated. It was also observed that making the hierarchy of contexts mirror the hierarchy of subroutine calls was not always advantageous, as the order in which facts were first discovered was not always the best ordering for later retrieval. The suggestion was made to generalize the context hierarchy from a tree to a lattice, and to form this structure independently of the order of subroutine calls. Thus, given facts 1-10 which follow from facts A, D, C, and P as shown in Figure 24a, the contexts in which various facts were true could be represented as in Figure 24b. But at this point one can raise the question of whether 24a or 24b is the most useful representation.

Fig. 23. Context Hierarchies
<==<fig3-18.press<

Fig. 24. Context Graphs
<==<fig3-19.press<

<==<fig3-20.press<

The use of contexts becomes even more difficult when information needs to be organized simultaneously along more than one dimension. For example, in medical diagnosis some things are true about the patient at one time, but not at another. Therefore, one needs a series of temporal contexts. At the same time, one needs contexts corresponding to various hypotheses about what is wrong with the patient. Unfortunately, some of the temporal findings are hypothesis dependent and some are not. In this problem, a series of explicit relationships has been preferred over contexts by most researchers.

The possibility of creating a useful context hierarchy clearly depends on the nature of the problem being solved. Dynamic scoping of variables in programming languages is a special case of context hierarchy. It forms a tree-like data structure which has been useful in solving some well structured problems.

4.12 Temporal Structure.

If situations or contexts are not to be used to differentiate facts which are true at one time from those which are true at another, how is this to be done? One approach which has been known for a long time, but never explored in any depth, is to treat time as analogous to space. However space is represented, time is represented analogously. The key to this analogy is the acceptance of temporal parts.

Every entity which is a physical body occupies some region of space. Every part S, of the space occupied by a physical body determines a part, S', of the physical body. A physical body can be decomposed into spatial parts. Similarly, any entity occupies (persists through) a certain period of time. The time occupied by an entity has parts. Every part T of the time occupied by an entity determines a temporal part, T', of the entity: T' is as much of the object as occupies T (in a temporal sense). One of the dimensions of the structure of an object is the temporal dimension, the structure of an object can be decomposed along this dimension.

Presenting this view of temporal structure Karp [44] says:

> It may be objected against this view that if a person is seen on Tuesday, and then seen again on Wednesday, then it is the very same thing which is seen on both occasions. This is held to be incompatible with a person having a temporal part occupying Tuesday and a distinct temporal part occupying Wednesday.
>
> But I may see John when John is facing me or when his back is towards me. Similarly, I may see a building by looking at any side of it. If I see John when he is facing me, John's front visual surface is presented in the perception; if I see John when he is facing away, John's back visual surface is presented — different perceptions of John, different spatial parts of John are presented. Correspondingly, if I see John at one time, and then at another time, different temporal parts of John are presented in the perceptions. This is not in any way incompatible with each perception being a perception of Johnv. A perception in which any of John's temporal parts may be presented may be a perception of John.

In referring to these temporal parts we may say, "John, the boy," "the ten year old John," or "the young John." One sometimes speaks of different temporal manifestations of the same individual.

A discussion of the following scenario will show the usefulness of temporal parts. Suppose that a bear-figurine made of soft clay is reshaped into a tiger-figurine. In the course of the change, the bear-figurine goes out of existence and the tiger-figurine comes into existence. But the bear-figurine's going out of existence is not an annihilation and the tiger-figurine's coming into existence is not an emergence from nothing. They are made of the same clay which persists from one to the other. In fact, after the bear-figurine is destroyed and before the tiger-figurine is formed, we have only clay.

It was stated above that we may ask of any two descriptions whether or not they can apply to the same individual. For example, "the red block on the table today" and "the red block in the box yesterday" can represent the same individual because we believe that a block's identity as an individual is not changed by a change in location. By contrast, if I make a bear figurine into a tiger figurine, it appears that one individual has changed into another. While the distinction is clear at the extremes, it is not possible to specify what accumulation of change should constitute change to a separate individual. For example, a horse may have his parts replaced systematically with those of a cow without one being able to say precisely when he becomes a cow. Most descriptions of individuals do have elements which cannot be changed. For example, one cannot change the number of sides in the description of a square and have it remain a description of the same individual. An event of Bob sneezing cannot be described as someone else sneezing. It is essential that Bob do it for it to remain the same event.

In the above scenario it would be impossible to maintain that the descriptions "bear figurine" and "tiger figurine" both apply to the same individual, the *clay*. But this is not the intuition people have. They don't believe that a *bear-figurine* is the same individual as a *tiger-figurine* in the same way that they believe that Bob curled up in a ball is the same individual as Bob standing up. Further, this approach leads to the conclusion that two objects are the same individual whenever they are composed of the same matter. Such a conclusion not only goes against people's intuition but makes the notion of an individual too weak to be a practically useful assumption. The usefulness of this notion must rest on the persistence of the bulk of the individuals properties through time.

We are lead, then, to the conclusion that there are at least three individuals in the above scenario; the clay, the bear figurine and the tiger figurine. Since the bear figurine and the clay occupy the same space it is tempting to say that they are the same individual. But an individual either exists or it doesn't, and when the bear figurine ceases to exist the clay goes on so they can't be the same. We are lead to follow Karp [44] and assume that the individual which is a particular bear figurine is a particular temporal part of the individual which is the clay. Note that this implies that two individuals, an individual and one of its temporal parts, can exist in the same place at the same time.

When a green leaf turns to a red leaf this change of color is not a sufficient alteration to require the postulation of two separate individual leaves. On the other hand, the *color-of-the-leaf* is described first as green and then as red, and red and green do describe different individuals. Therefore, this change in leaf color requires the postulation of temporal parts of the *color-of-the-leaf*, but not temporal parts of the *leaf*.

Quine [83] gives an interesting example based on Heracleitus' problem: "You cannot bathe in the same river twice, for new waters are ever flowing upon you." Quine says:

> Let me speak of any multiplicity of water molecules as a <u>water</u>. Now a river is at the same time a water stage, but two stages of the same river are not in general stages of the same water. River stages are water stages but rivers are not waters. You may bathe in the same river twice without bathing in the same water twice, and you may, in these days of fast transportation, bathe in the same water twice while bathing in two different rivers."

It is often claimed that no object can be in two places at once, though it can occupy two or more times at one place; this could be taken as a basic difference between time and space. But someone who is standing with one foot in the doorway and the other outside is occupying two places at once. Of course it is tempting here to object that only a part of him is in either place; he is not both entirely inside and entirely outside. But when this has been said, it must be remembered that it is a different temporal part of an object which, at a given place, occupies two or more times.

Over an interval of time, something can move back and forth in space. Can anything move back and forth in time over an interval of space? Consider an earthquake which occurs at $time_1$, in two nearby towns, and that it occurs everywhere between these two towns, but at one of these intermediate places at a time other than $time_1$. The earthquake is thus something which, over an interval of space, moves back and forth in time.

There is one point on which the analogy seems less intuitive. Consider the situation where an object is annihilated and then recreated later at the same place. It is in the same place at two times, but not all intervening times. The analogy would be two identical billiard balls, which we treat as one ball occupying two places at the same time, but not all intervening places. It is grantedly counter-intuitive to treat this as one ball, but by the symmetry of the situation we might then also treat the recreated object as a different individual.

5. Ontology

5.1 Introduction

The notion of <u>structure</u> used in the previous chapter is a broad interpretation of the usual sense of the word. We pursue a line of thought which may be found in Eaton [22, p. 127].

> If analysis is breaking up an object into its parts, is not the discovery that an object is an instance of a universal, or that a certain description fits it, something other than analysis?
>
> I can analyze an object into parts, a, b, and c, which are related in a certain way. I can discover that my ink-well is composed of a small glass jar with a glass cover, and this seems to be genuine analysis. But if I find that the ink-well is black and that it is on the table, I am discovering relations and quantities which attach to it, and these do not seem to be parts in the same sense. And if I now describe it as 'a black object on the table,' I do not appear to be analyzing the ink-well.
>
> This is a superficial distinction. Whether its blackness and its position on the table are 'parts' of the ink-well depends on the point of view. These may be, on one definition, external to the object and, on the another definition, internal to it. If by 'the ink-well' I intend the bare x. the individual term—what Locke would have called its 'substance'—then the fact that it has a glass container and a small glass cover is as external to it as the fact that it is black and on the table. But I can mean by 'the ink-well' either the individual, the *x* alone, to which certain predicates attach, or I can mean the individual taken with more or fewer of its predicates and relations. And this is possible because the individual, though it may not be completely determined by its predicates and relations, is continuous with them. They enter into the individual. Any predicates and relations of an object are 'parts' of the object in a logical sense, for they <u>participate in</u> the object; and when the object is described, it is analyzed as a complex whole in which these quantities and relations are elements. The bare individual, the *x*, of which the predicates and relations hold, becomes one element along with the others.

Let us re-interpret what Eaton seems to be driving at in terms of the ideas being developed here. We consider the world to be one big structure. People find it useful to identify various substructures of this structure as individuals. An individual may be analyzed by describing it. For example, an ink-well may be described as being black and being on a table. This will be done by taking the *color-of-the-ink-well* and the *location-of-the-ink-well* to be individuals in the structure of the *ink-well* and describing these entities in turn. The *color-of-the-ink-well* is an individual which admits of the description, BLACK.

The *location-of-the-ink-well* and the *table* are related by an individual we might call *on*. Two ways of conceptualizing the structural relationship between the *location-of-the-ink-well*, the *table*, and *on* will be presented. One possibility is that *on* is described as a state with the *location-of-the-ink-well* and the *table* in its structure. The other possibility is that *on* is described as an attribute in the structure of the *location-of-the-ink-well* with only the *table* in the structure of *on*. Both of these descriptions of *on* can be maintained at the same time! What is in the structure of an entity is in part a question of how one wishes to conceptualize it. It is useful to maintain alternative conceptualizations of the same entity. The world admits of alternative structural decompositions—but not arbitrary structural decompositions. Any decomposition must maintain a correspondence between the structure of the description and the structure of the world.

Individuals like the *color-of-the-ink-well* and the *location-of-the-ink-well* are sometimes called property particulars, because color and location are often thought of as properties of objects. It is hard to make conclusive arguments for the existence of property particulars. We introduce them in order to maintain the structural correspondence between descriptions and the entities they describe and because it leads to a straightforward interpretation of expressions such as

(54a) The weight of the stone depressed the scale.
(54b) What is the state of Bob's health.
(54c) Bob's exercising caused his physical fitness.

If the weight of the stone depresses a scale, it is a cause. It is convenient to attribute a cause to some entity. Similarly, it is convenient to say that entities have states and are caused.

The study of the kinds of things which exist is known as ontology. In this chapter we explore the traditional sorts of things one might find in an ontology as well as some less traditional ones motivated by English usage. We also develop the proposal, mentioned above, that whether something is, for example, an attribute or a state may depend on how we choose to look at it, not just on the nature of the world.

5.2 Natural Kinds

The advantage of forming abstract descriptions is prediction. The same or similar entities can be recognized in the future and their behavior forcast. This was a theme of Chapter 2. In that chapter the dilemma of the captain of the Ship of Theseus was used to illustrate how the existence of individuals is an assumption people make, and one that works well unless their criteria for re-recognition of an individual is thwarted by a philosopher's example.

In the computer processing of natural language we are not so much interested in whether individuals exist. What is important to us is that people talk and think of them as existing. The same can be said for natural kinds.

We state that kinds exist in the world because it is our intuition that English speakers make this assumption. For example, we claim that in

(55) The elephant plays an economic role in some countries.

the speaker is referring to elephant kind, and he is doing so as if elephant kind exists as an entity in the world. The speaker believes in individual elephants, sets of elephants, and elephant kind. The existence of kinds like elephant kind is not universally accepted. For example, Lyons [55] develops an ontology which admits the existence of the controversial entities, propositions, but not the existence of kinds.

At least when dealing with countable entities, it is important to distinguish two alternative conceptions of universals—the conception of universals as typical individuals (or exemplars) and the conception of universals as kinds [44]. The notion that universals are typical individuals might be stated as

(56) The Exemplar Principle: All and only those things are true of an abstract substance which are true of its typical examples.

So stated the principle suffers from oversimplification. For example, a typical example of an abstract substance is an individual, while an abstract substance itself is not an individual. Further, as Wittgenstein [111] points out, typical examples frequently have almost nothing common to all of them, e.g., some cups have handles, some don't, some are glass, some are porceline. Nevertheless, we may leave these difficulties aside for the moment and consider the simplified Exemplar Principle in distinguishing between typical individuals and kinds.

The indefinite article *a* may be used in referring to typical individuals— *a lion has four legs.* Typical individuals may also be described using the definite article *the* and the bare plural, e.g., *lions.*

(57a) A lion has four legs.
(57b) The lion has four legs.
(57c) Lions have four legs.

There are other statements which can be made with *the* and the bare plural which cannot be made with *a.*

(58a) The lion is almost extinct.
(58b) Lions are almost extinct.
(58c) *A lion is almost extinct.

(59a) The lion is widespread in Africa.
(59b) Lions are widespread in Africa.
(59c) *A lion is widespread in Africa.

(60a) The lion is a species of mammal.
(60b) Lions are a species of mammal.
(60c) *A lion is a species of mammal.

These later statements are widely recognized as referring to universals, but they ascribe to these universals properties which cannot be held by any individual lion and thus not by the typical individual. No individual lion is almost extinct, or widespread, or a species. These latter statements refer to lion kind, that is, to lions as a species.

5.3 Core Facts and Concept Formation

Having acknowledged the existence of kinds and individuals, we would like to go on to make some general claims about the sorts of individuals one might find in the world. For example, above we postulated the existence of property particulars, like individual heights. But can we make any claims about what it means to be a property particular as opposed to some other sort of individual—claims that have operational implications for knowledge representation. Such claims could be either about individuals or about the sorts of descriptions which they admit.

One individual may satisfy many different descriptions, he can be a dog, a barker, an animal. At the same time, one description can often be satisfied by many different individuals. This happens because any structural description of a typical individual, any concept, represents a generalization of experience along certain dimensions and not along others. If someone is short, I may try to predict his behavior from what I know about short people. If the same person is a doctor, I have another method of forecasting. One abstracts across past individuals and then fits the appropriate abstractions together to predict the behavior of a new individual.

The procedure has an analogy in production forcasting of new cars. Cars have so many optional features that almost every one is unique. A forecast of all the cars with a particular configuration would be highly eratic and not very useful. What is actually done is to forecast by color for the paint department, by body style for the assembly line, and by options for the suppliers. Different abstractions are useful for different purposes.

This many to many relationship between individuals and their descriptions means that general claims could be focussed either on individuals or their descriptions. The distinctions to be made in this chapter are distinctions regarding descriptions. They are distinctions between sorts of concepts. They relate to the world only indirectly in that the world is said to support the formation of concepts of these sorts.

The general difficulty in "defining" the meaning of a concept has been pointed out by Winograd [109]:

The word *bachelor* has been used in many discussions of semantics, since (save for obscure meanings involving aquatic mammals and medieval chivalry) it seems to have a formally tractable meaning which can be paraphrased *an adult human male who has never been married.* Traditional theories of semantics deal with tasks such as determining whether the sentence *my bachelor uncle is unmarried* is analytic. In the realistic use of the word there are many problems which are not as simply stated and formalized. Consider the following exchange:

Host: I'm having a big party next weekend. Do you know any nice bachelors I could invite?
Friend: Yes, I know this fellow X...

The problem is to decide, given the facts below, for which values of X the response would be a reasonable answer in light of the normal meaning of the word *bachelor.* A simple test is to ask for which ones the host might fairly complain *You lied. You said X was a bachelor.*

A: Arthur has been living happily with Alice for the last five years. They have a two year old daughter, and have never officially married.

B: Bruce was going to be drafted, so he arranged with his friend Barbara to have a justice of the peace marry them so he would be exempt. They have never lived together. He dates a number of women, and plans to have the marriage annulled as soon as he finds someone he wants to marry.

C: Charlie is 17 years old. He lives at home with his parents and is in high school.

D: David is 17 years old. He left home at 13, started a small business, and is now a successful young entrepreneur leading a playboy's life style in his penthouse apartment.

E: Eli and Edgar are homosexual lovers who have been living together for many years.

F: Faisal is allowed by the law of his native Abu Dhabi to have three wives. He currently has two and is interested in meeting another potential fiancee.

G: Father Gregory is the bishop of the Catholic cathedral at Groton upon Thames.

Words as Symbols for Abstract Exemplars

The cast of characters above could be extended indefinitely, and in each case there are problems in deciding whether the word *bachelor* could appropriately be applied. In normal use, a word does not convey a clear definable combination of primitive propositions, but evokes an *exemplar* which posesses a number of properties. This exemplar is not a specific individual in the experience of the language user, but is more abstract, representing a conflation of typical properties. A prototypical bachelor can be described as:

1. a person
2. a male
3. an adult
4. not currently officially married
5. not in a marriage-like living situation
6. potentially marriageable
7. leading a bachelor-like living situation
8. not having been married previously
9. having an intention, at least temporarily, not to marry
10. ...

Each of the men described above fits some but not all of these characterizations. Except for narrow legalistic contexts, there is no significant sense in which a subset of the characteristics can be singled out as the "central meaning" of the word. In fact, among native English speakers there is little agreement about whether someone who has been previously married can properly be called a bachelor and fairly good agreement that it should not apply someone who is not potentially marriageable (e.g. has taken a vow of celibacy).

Not only is this list open-ended, but the individual terms are themselves not definable in terms of primitive notions. In reducing the meaning of *bachelor* to a formula involving *adult* or *potentially marriageable*, one is led into describing these in terms of exemplars as well. *Adult* cannot be defined in terms of years of age for any but technical legal purposes and in fact even in this restricted sense, it is defined differently for different aspects of the law. Phrases such as *marriage-like living situation* and *bachelor-like life style* reflect directly in their syntactic form the intention to convey stereotyped exemplars rather than formal definitions. There have been attempts to use quantificational and statistical methods to provide a more precise formalization of "fuzzy" concepts. Labov [Labov74], for example has attempted has attempted to map out the characteristics of a population of speakers in distinguishing between exemplars such as *cup* and *glass* as the characteristics of the object being described are varied along a number of dimensions. However, these still provide an overly uniform mathematical structure on the different characteristics of the exemplar.

An exemplar brings together different kinds of characteristics which have quite distinct properties in selecting lexical items. There is an intuitive impression, for example, that the meaning of *bachelor* includes some absolutely necessary conditions, and is being metaphorically stretched when applied to individuals violating them, (as in talking about *my bachelor aunt* or responding to the question. *Has your dog ever sired puppies?* with *No, he's a bachelor*). But even for seemingly straightforward predicates such as *human* and *male*, it is easy to generate examples where notions of primitive predication are inadequate (for example in situations involving eunuchs and near-human monsters).

There is also a tendacy (as G. Lakoff [53] has pointed out) to use *hedges* when applying a term to cases where characteristics which are non-essential but typical are being connoted (*Anthony's wife is away on business trips so often that he's a regular bachelor* or *That little episode with Sarah confirmed my belief that Carl is a true bachelor*). The ways in which different hedges emphasize different characteristics is highly complex, and it is impossible to draw a sharp line between "essential" and "secondary" or between "defining" and "characteristic" properties.

The complexity of all these problems should have an impact on the nature of semantic theory. A theory which ignores them cannot claim to be a full explication of the meaning of words. This case is not all pathological: in fact, *bachelor* has been used as an example because its definition seems so precise compared to words like *friend* and *game*.

Winograd's points are well taken, but as Putnam [81] says of the followers of the philosopher Wittgenstein:

in the process of 'debunking' this fact—the fact that something as simple as a lexical definition can convey the use of a word—they forget to be impressed by it. To be sure there is a great deal of stage setting, but it is rarely stage-setting specifically designed to enable one to learn the use of this word. The fact that we can acquire the use of an indefinite number of new words, on the basis of simple 'statements of what they mean' is an amazing fact: it is the fact, I repeat, on which semantic theory rests.

Putnam argues that to learn a new concept, one need only be supplied with certain core facts which, the world being as it is, will allow him to correctly build up greater and greater knowledge of the concept and to correctly integrate it into his general knowledge. Among these core facts is some way of picking out one or more individuals which may be described by the concept and some indication of the direction of the abstraction. For example, if one is shown a tiger at the zoo and told that tigers are a species of animals he can learn more by observing this individual. One believes implicitly in the notion of species, that there may be deep lying explanations of *tigerness* (e.g., genetic), that other tigers exist, that odd ball tigers exist of which this might be one, etc.

Sometimes children using this procedure go wrong in amusing ways. A three year old boy assumed that tires which weren't flat were *spare*, because his father had replaced a flat tire with the spare one, and this was the key difference between them. He made a wrong assumption about the direction of abstraction. On being taken to the *doctor* another boy exclaimed, *but mamma, he's a man*. He had assumed that *doctor* named a species, not a professional role.

There is no firm theory of what kinds of notions one would expect to find in core facts. One type of core fact, seen in the dictionary, is a trick for helping the learner pick out individuals for study. Also of interest would be a catagorization of basic types of abstractions. The following sections discuss some ideas which arise from a consideration of English usage.

5.4 Classification of Descriptions

If one chooses to admit the existence of particular events or attributes, this doesn't mean he feels they are in every way like stones or tigers. In this section such individuals and their descriptions will be classified in a way which points out some possible differences. Descriptions will be classified into five types:

(61a) Attributes
(61b) States/Processes
(61c) Relational Characterizations
(61d) Attributive Characterizations
(61e) Other kinds

This classification will be based on two criteria: (a) ontological priority, and (b) requirements for the identification of individuals of a kind.

The notion of one kind being ontologically prior to another may be illustrated with the kinds *physical body* and *weight*. Since one can conceive of physical bodies without weight, but not the reverse, physical bodies are said to be ontologically prior to weight.

The requirements for identification came up in regard to the Ship of Theseus. They will be reviewed here in terms of another example.

```
0    X
+    X
```

Looking at the example above, one may say that the figure above the + is the same as the figure to the left of the X. Using another sense of same, one may say that the figure in the upper ~~left~~ right corner is the same as the figure in the lower right corner. In the first case we have numerical identity, in the second case we have only structural identity. In LISP terms, we could say that in the first case the figures are EQ, while in the second they are EQUAL. To identify an individual we must establish numerical, or EQ identity.

Suppose that a person is blindfolded and moved about so that he loses track of his whereabouts. When the blindfold is removed, he discovers that he is in his own bedroom. He believes the room is numerically the same. He could be in a carbon copy of that room, but he firmly believes that he is not. He believes in the existence of individual lamps, chairs, etc. and the continuity of their existence in space and time. Seeing the familiar configuration of these familiar individuals he firmly believes that he has uniquely established his whereabouts.

Now if the same person were to come upon a piece of his furniture in an unfamiliar location, he would probably be reluctant to identify it as numerically the same. His memory of the description of the piece is not enough, he needs the bedroom context to be sure. Unblindfolded, he must verify the numerical identity of the room before he can numerically identify its constituents. On the other hand, since the room is composed of its constituents, one might be le~~a~~d to assume that our subject could not identify the room without first so identifying its constituents. Neither the room nor the constituents come first. After all, he might be in a carbon copy. We sense, however, that this is not right. One never feels oneself to be in such a dilemma.

The obvious answer is that to verify the room is numerically the same people, in fact, only require that pieces of the same structural description are at their expected positions in the room and that there is no strong negative evidence. They consider the description of their room sufficiently unique to establish its identity. They are not concerned with whether the pieces of furniture in the room are numerically the same as their own. Once the room is verified to be numerically the same, the pieces can be also so identified if that is desired.

Underlying the previous discussion is the assumption that corresponding to the structure and context of an entity we may have structural descriptions of the entity and descriptions of the context of the entity. Further, the structural description of an entity, such as a lamp, may be used either: (1) as a role in the structural description of another entity, or (2) to establish the numberical identity of an individual. We may find that an individual fills a role without establishing the numerical identity of the role filling individual. Using these distinctions, kinds may be classified into the five categories given above. We now discuss each category in further detail.

5.4.1 Attributes

Examples are attributes such as *health*, *color*, and *length*. Note that one can refer to *health* as a kind, as in *health is important to happiness.* He can also identify a particular health, but only by referring it to the entity for which it is a part of the structural description, as in *What is the state of Bob's health.* He can't say *some health* or *a health*. One can describe the color of a particular chair as red, but the description *red* does not numerically identify that individual the way describing it as the color of a particular chair does.

Based on these considerations, attributes will be defined as descriptions whose corresponding particulars are individuals which can only be numerically identified through their "dependence" on one or more other numerically identified entities. For attributes, dependence means that they fill a role of the <u>sponsoring</u> particulars' structural description. Attributes such as *height*, *weight*, and *health* have corresponding particulars which fill roles in directly identifiable entities. Physical disturbances like *weather*, *flashes*, and *bangs* can be described as attributes of portions of space which are themselves attributes of directly identifiable entities. Some descriptions in this category may have either relation to their sponsoring entity. For example, we may say either

(62a) the {smell, temperature} <u>in</u> a room
(62b) the {smell, termperature} <u>of</u> a room

As a subclass of attributes we may distinguish scales, such as height, weight, and size. A scale may be modeled by a ray of a line running from zero to infinity. Scales support the notion of magnitude comparison. The method of making the comparison will vary with the scale. In the case of a complex attribute like *size*, for example, the method will depend on the object having that attribute. A second class of attributes are opposites like happy and unhappy. These may be modeled by the entire real line. Another important subclass of attributes may be termed intervals. Examples are degree of fullness and extent to which one is grown up. Intervals may be modeled by a line segment running from zero to one.

Attributes may also be subdivided according to their temporal behavior into catagories corresponding to states and processes.

5.4.2 States/Processes

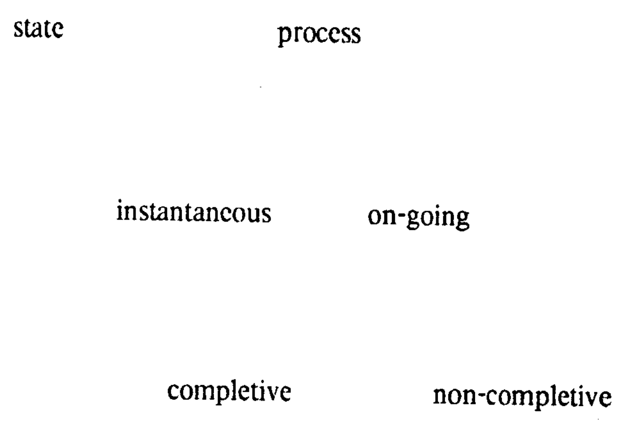
Examples are states and processes involving physical bodies such as hit and hold. States and processes will be defined as descriptions whose corresponding particulars are individuals which, as with attributes, can only be numerically identified through their dependence on one or more numerically identified sponsoring entities. However, in this case dependence means that the sponsoring entity either fills a role in the sponsored particular's structural description. For example, interpreting *Bob kissed Mary by the barn yesterday*, to specify a particular process has *Bob* specifying the role of subject, and *Mary* specifying the role of object, *by the barn* specifying the location, and *yesterday* the time. It would be difficult to argue that no state or process can be identified by its structural description alone. Our intuition is, however, that English speakers don't do this. (Even if a speaker says *remember the time that a guy kissed a girl on the elbow*, he is implying the hearer knows about the incident and is thus referring it to the hearer.)

For any two individuals *a* and *b*, if *b* can be described as an attribute of *a*, then *a* can be described as filling a role in state or process *b*. In *Tom sneezed*, for example, sneezed could be considered either a particular attribute in the structur~~r~~al description of Tom or a particular state with Tom as an individualizing particular in its structural description. It is important to remember that the distinction between attributes and states/processes is a distinction between structural descriptions, not a distinction between individuals. The same individual can often be described either as a particular attribute or a particular state depending on how we wish to conceptualize it. States and processes are participated in or done by the entities sponsoring them, while attributes are in the structure of their sponsoring entities.

The *rising of the sun* is normally taken to be a process identified by having the *sun*, a certain *place*, and *moment of time* in its structure. Montague [74] treats this example by making *the rising of the sun* an attribute of the *moment in time*. That is, he takes the option of making it an attribute. While this is not really foreign to our treatment, we must consider it as insufficient. The moment in time is not itself enough to identify an event.

States and processes may be subdivided as shown below. States and processes are described using verbs, nouns, adjectives, adverbs and prepositions.

Fig. 25. States and Processes
<==<fig4-2.press<



States may be described by what Joos [43] calls <u>status</u> <u>verbs</u>. He claims there are two main groups:

(63a) psychic state, including the specific perceptions (*see, hear,* etc.) and the intellectual and emotional attitudes (*believe, understand, hate, like, regard,* etc.)
(63b) relation, such as the relations of representing, depending, excluding, and so on.[8]

It is frequently observed that status verbs repell the progressive, e.g., *Bob is resembling his father more each day.* Another property of status verbs is that they cannot have future reference without a specific time shifter such as *will.* With a process verb we can say *Don't worry: he leaves next week*; but we can't say, *Don't worry: the baby resembles his father next year.*

Joos points out that most status verbs can be used both as status verbs and as process verbs, but they have only one sense as a status verb:

> Take "hear" as a typical verb-base used both ways. As a process verb it has a number of fairly distinct meanings, as in "The Judge is hearing a case just now; you'll have to wait;" "I hope to be hearing from him soon", and as many others as we care to distinguish. But as a status verb it seems to have exactly one meaning.

What is the nature of a state? Joos says:

> For relation verbs it is obvious, and for psychic-state verbs it is now not too hard to see, that the (single!) meaning of each status verb is such as to reject the time limited validity of temporary aspect. 'That makes no difference' is not a process, not an event that essentially preceeds but is now frozen for our inspection: it is instead a relation between 'that' (whatever it is) and the whole world we live in: it doesn't happen, but simply <u>is so</u>. Equally truly, 'I drop the tablet into this warm water, and you <u>see</u> it dissolves quite nicely' is not a process of seeing; it doesn't mean seeing is proceeding within time so that it could be made temporary by using the temporary aspect. Instead, this perception, a kind of psychic state if you like, is a sort of <u>relation</u> between the beholder and the dissolving tablet.

8. It seems interesting that *have* and the copula *be* are status verbs of this sort, so that possession in English is a status: *He has a farm* and *He is a farmer* are usually synonymous.

While status verbs have a sense which always describes states and not processes, other verb senses can be used to describe either states or processes. For example, *The pig ate a sandwich* describes a process, while *pigs eat corn* describes a state. *Dinosaurs ate kelp* can describe either a state or a process.

A process can be either <u>instantaneous</u>, e.g., *hit*, or <u>on-going</u>, e.g., *crying* or *building a house*. *Building a house* is <u>completive</u>, but *crying* isn't. One can say: *How long did it take you to build the house?*, but not *How long did it take you to cry?* (disregarding the interpretation *to start to cry*). Also, if one has been crying then he has cried, but if one has been building a house, he has not built it. On these distinctions see, for example, [23].

Note that while *hit a tree* describes an instantaneous process, *hitting trees* can describe an on-going process which consists of repeated hits. Note also that while *walking* is a non-completive process, *walking to the store* is a completive process.

These examples provide further evidence that the distinction between states and processes is not a distinction between verbs, but between senses of phrases. Adjectives can be sorted into those which normally describe states and those which normally describe non-completive processes. The former include *long* and *male*, and resist the form:

(64) I found them _____.

which the latter (e.g., *sick*, *true*) accept. Prepositional phrases (e.g., *in the woods*) normally describe non-completive processes.

5.4.3 Relational Characterizations

Examples are father, employee and neighbor. These descriptions have corresponding particulars which are individuals which can be numerically identified without any sponsoring entities. However, such a description must have an ontologically prior description and particulars of the relational characterization always fill a role in the description of the ontologically prior description. For example, to be a natural, legal or ersatz father requires in each case that a certain state or process exist. A husband may be identified by saying who he is the husband of, or what marriage he is involved in. I may identify someone as a husband without identifying his wife. But since he may have been married more than once, this is not numerical identification of this particular. To get this I must specify the wife. In general, the numerical identification of a particular in this category requires at least the identification of a sponsoring entity which is either an entity of its ontological prior description or an entity filling a role in the structural description of this ontologically prior description.

One must make a rather fine distinction at the boundary between relational characterizations and attributes. An example of the distinction we have in mind was noticed by Fillmore [25]. He shows that *hit* takes a location and *break* an object.

(27) I broke the top of the table.
(28) I hit the top of the table.

In (27) the noun <u>top</u> must be referring to the top of a table as a more or less distinct object, while in (28), it can refer either to that or a portion of the surface area of the table.

The sense of *top* in (27) is a relational characterization. In (28) we have either this sense or the sense of *top* as an attribute of the table.

5.4.4 Attributive Characterizations

Examples of these are *bachelor* and *teenager.* These kinds have corresponding particulars which are individuals which can be identified without any sponsoring entities. However, such a description must have an ontologically prior attribute. For example, there can be no notion of *teenager* without a notion of *age.* Particulars of this sort must have a certain characterization of this ontologically prior attribute.

5.4.5 Others

The remaining structural descriptions are either adequate to identify a particular, or they are generally inadequate to identify a particular and have no special relation to sponsoring particulars. For example, one identifies his own room from its structure, but encounters severe problems in keeping tabs on a particular electron.

5.5 Word Senses and the Classification of Descriptions

In making the above classification of descriptions it was implied that such a classification would be useful in describing the "core facts" associated with a concept. We can shed some light on this by attempting to distinguish senses of words using the above classification. Very frequently, a single word will have senses in several of the categories—the categories are one way of distinguishing senses.

5.6 Attributes

Expressions referring to attributes are varied in form. For example:

(65a) What is the length of that snake.
(65b) It reached a length of 15 ft.
(65c) He dived to an unsafe depth.
(65d) What is the distance from Boston to New York.
(65e) The temperature within 500 ft of the center is over 500^{o}.
(65f) The temperature of the oven is higher than the temperature in this kitchen but not much.
(65g) He came out from under the house.
(65h) Red is my favorite color.
(65i) Do you like the weather in England.
(65j) He was at the top of the class.
(65k) At 1000 yards from the center no damage was done.

What might be called "attribute words" also have senses which are not attributes:

(66a) The strong man walked up to the weight.
(66b) Each person was issued a length of rope.
(66c) You have put on some weight.
(66d) I broke the top of the table.
(66e) He floated over the tree tops.

In these latter examples, *the weight* is a physical body named by its most salient attribute, weight. *A length of rope* is like *a piece of rope.* In the phrase *put on some weight,* the noun *weight* refers to the mass entity associated with the attribute, weight, in the same way that oak wood is associated with oak tree. This relationship is sometimes called substance and form. The table top example was given earlier to illustrate the subtle difference between an attribute and a relational characterization. Finally note that *tree tops* doesn't have a singular. This idiomatic expression refers to a generic place. The expressions which do describe attributes may be classified into generics and individuals.

(67a) Generic: a length of 25 ft, an unsafe depth
(67b) Individual: the length of that snake, the distance from Boston to New York, the temperature of the oven, the temperature in this kitchen, the weather in England, the top of the class, 1000 yards from the center, under the house

Recall that an attribute is individuated in part by being in the structure of another entity—it is "an attribute of" that entity. The expressions describing attributes can be classified based on whether the attribute and owner are explicitly mentioned.

(68a) Explicit Attribute and Owner: the length of that snake, the temperature of the oven, the top of the class
(68b) Explicit Attribute and Implicit Owner: the distance from Boston to New York, the temperature in this kitchen, the weather in England, 1000 yards from the center, the temperature within 500 ft of the center

The expressions with implicit owners all have owners which are enclosures, surfaces, locations, or directed paths. Temperature and weather can be attributes of any of these, while distance (or yards) can only be an attribute of a path.

The expression *the distance from Boston to New York* associates the descriptions of a source, *from Boston,* and a destination, *to New York,* with a distance. The question arises whether one should take these to be the source and destination of a distance or the source and destination of a directed path which has the distance as an attribute. We have already suggested that the distance is an attribute of a directed path. What do the source and destination descriptions apply to? Our answer will be an equivocation, but one which exhibits a powerful and generally operative mechanism.

Observe that if A is part of B and B is part of C, we may or may not speak of A as part of C. For example, the first joint is a part of the index finger, but not a part of the body. The fingernail is a part of the finger and is considered a part of the body. From this we conclude that a part of a part may optionally be considered a part of the larger structure. More generally, a role of a role may optionally be taken to be a role of the larger structure.

Note that if a source and destination are sufficient to individuate a directed path, then giving the source and destination of a distance gives the source and destination of its directed path and individuates this. Since the directed path is the owner of the distance, giving the source and destination of the distance determines the individual of which the directed path is an attribute.

5.7 Relational Characterizations

Expressions referring to relational characterizations are:

(69a) the message from Bill to Tom about the meeting
(69b) the author of the book
(69c) the general secretary of the party
(69d) my candidate for a trip to the moon
(69e) prolegomena to any future metaphysics
(69f) the prospects for peace
(69g) the reason for his refusal
(69h) his advantage over his rivals
(69i) his mercy towards the victims
(69j) Bob's father
(69k) the three friends
(69l) the top of the table

Many of these examples were mentioned by Chomsky [16]. He points out that deriving *author* transformationally from *auth* seems less attractive when one considers that the same approach should then probably be used for phrases like *general secretary*. The topic here is not parts of speech and syntactic situations, but Chomsky's concern translates into the fact that relational characterizations can have ontologically prior descriptions and processes. Determination of the descriptions ontologically prior to the senses of these particular expressions takes us into the realm of speculation. Nevertheless, it is probably more informative to discuss these difficult cases. The descriptions prior to these expressions might be classified as follows:

(70a) <u>Have an Ontologically Prior Process</u>: the message from Bill to Tom about the meeting, my candidate for a trip to the moon, his mercy toward the victims
(70b) <u>Have an Ontologically Prior State</u>: the author of the book, the prospects for peace, the reason for his refusal, his advantage over his rivals, Bob's father, the three friends
(70c) <u>Have Some Other Description Ontologically Prior</u>: the general secretary of the party, prolegomena to any future metaphysics, the top of the table

Suppose Bill takes out a slip of paper and writes on it, *the meeting tomorrow is cancelled.* He sends the slip to Tom. This slip of paper becomes a "message" by virtue of its being used by Bill to communicate information to Tom, by virtue of the role it plays. The message inherits roles from its ontologically prior process.

5.8 Classification of Descriptions Based on Form

In English, a sense of a noun may be mass (e.g., *advice, water, furniture*), count (e.g., *book*), or unlimited in space and time (e.g., *unlimited*). Some nouns are ambiguous in this respect. For example, the noun *egg* can be count or mass:

(71a) I cracked an egg. *count*
(71b) He had egg ~~in~~ on his face. *mass*

There are a number of linguistic tests for classifying nouns. For instance, mass nouns take the quantifiers *much* and *a little* whereas count nouns take the quantifiers *many* and *few*. Unlimited nouns refuse all of these quantifiers.

Mass and count senses are related in the following ways:

(72a) Substance: For any count concept e.g., an *egg*, we may form the concept of the substance it is composed of, e.g., *egg*. Other mass examples, *they took up a short length of street, we came across a wide expanse of cookie.*

(72b) Individuators: For any mass concept, e.g., *toast*, we may form concepts for individuating it, e.g., *piece of toast.* Steps a) and b) are combined in *piece of cookie.* Individuating concepts are usually somewhat idiosyncratic to the mass concept being individuated, e.g., *stroke of luck, speck of dust.*

(72c) Collectives: Some count concepts, e.g., *family, flock, committee*, are composed of other count concepts.

(72d) Non-empty Containers: When two conjoined verbs share the same object noun phrase, one expects that they will both take it in the same sense. This accounts for the oddity of a sentence like: ??*She baked a cake for and called her son John.* So when a sentence like: *I opened and ate a can of beans* seems perfectly normal, one assumes that both these verbs take the same sense of *a can of beans* in spite of the fact that a person opens the can but eats the beans. A *can of beans* is an entity like *a coke*, in *I opened and drank a coke.* The can is a container with something in it, which has properties different than those of an empty container or those of a quantity of the stuff contained. For example, compare:

(73a) I broke a glass.
(73b) ?I broke a glass of wine.
(73c) I broke a glass containing wine.

5.9 Summary

If one allows individuals other than physical bodies it becomes important to have some systematic understanding of these individuals and how they differ. Here the decision was made not to discuss these individuals, but rather, the descriptions they satisfy. Winograd expresses well the difficulty of making a definition of such a description, but Putnam suggests that there are certain core facts which go surprisingly far toward helping a learner form a new concept. We suggest that one key element of these key facts is the type of abstractions. With this in mind some very general distinctions were made on the basis of structure and form. In the next chapter, more, somewhat less general distinctions will be made.

6. Abstraction and Analogy

6.1 Introduction

Abstraction and analogy are two methods of understanding something new in terms of previous experience. Figure 26 shows a basic relationship between abstraction and analogy. Whenever A is analogous to B, one can form a description C which characterizes them both. For example, the oscillation of a weight connected to a spring is analogous to the oscillation of current in an electrical circuit composed of a capacitor, inductor, and resistor in series. Both can be described by the same first order differential equation.

This relationship between analogy and abstraction makes it more difficult to know how to deal with many expressions. For example, should we say that a *dog house* is analogous to a *house* (for people) or should we say there is one abstract sense of house which covers both of these. It is possible that people have all the information: concepts of house, dog house, an abstract notion of house, and an awareness of and ability to play on the analogy between a house and a dog house. (I remember when I first took my son to the zoo after he started to talk. We were watching the camels when the food truck arrived. I said *Look, the camels are going into their house for supper.* His response, *Do they eat supper in the kitchen or the dining room?*)

In linguistics it is sometimes tacitly assumed that is people have a general rule, they will not also have all the specific cases. But consider, for example, *crackage.* People know what this means, and they also know that it is not a word. They know it fits with *breakage* and *damage*, but that these are used and *crackage* isn't. Thus, at least in some cases people know both the general rule and all of its normally occurring cases.

Why have both the rule and the examples? The rule facilitates understanding new cases. Consider *transmote*, most people don't feel they know what this means until they learn that President Johnson wanted to get rid of some government officials which he couldn't promote or demote, so he decided to transmote them. People have the examples promote and demote needed to understand transmote, but they don't already have a rule which these and transmote will fit.

Fig. 26. Abstraction and Analogy
<==<fig5-1.press<

It could be that individual cases are remembered for efficiency reasons, and it could be because we usually know some idiosyncratic information about each word sense. At any rate, it would be false to argue that abstractions don't exists because the instances of them are idiosyncratic in some way.

In summary, it seems likely that a computer system should have rote memory, analogy, and abstraction.

6.2 Three Criteria for Filling a Slot

An important use of abstraction is in determining what can fill a given role in a description (slot in a frame). It is possible to distinguish three levels of acceptability.

(74a) Conceivable e.g., *poison is edible.*
(74b) True of the world e.g., *grass is edible.*
(74c) Typical of the world e.g., *lettuce is edible.*

We see the progression in

(75a) John ate the movie. *inconceivable*
(75b) John ate the rubber ball.
(75c) John ate the leaves.
(75d) John ate the peas. *typical*

The notion of conceivability has been developed by Keil [45], who in turn drew his inspiration from papers by Sommers [98]. Keil suggests that if something is inconceivable, then its negation is also inconceivable, while if something is merely false its negation is perfectly acceptable. For example, *that bachelor is married* is conceivable but false. Instead, *that bachelor is not married* sounds fine. By contrast, *the idea is green* is inconceivable, and *the idea is not green* is no better, just as inconceivable.

The split between conceivability, truth, and typicality could be modeled in a computer program. For each frame slot, we would have a structural description of what could fill that slot. Whenever the slot was filled, the filler would have to match that description. For example, the sentences

(76a) I sliced the ice cream.
(76b) I mopped up the ice cream.

show how since sliced takes a solid and mopped up a liquid, one must pick the appropriate phase of ice cream to fill the slot. As a second example, note that *on* takes a surface and *in* an enclosure, so that

(77a) I looked out the window and saw it on the grass. 2-D

evokes a two dimensional grass image while

(77b) He got down on his knees and hunted for it in the grass. 3-D evokes a three dimensional grass image. The structural description of what can fill a slot would presumably be quite abstract, allowing many things to be viewed in such a way that they could fill the slot. If there was no way of viewing a particular thing, *x*, as fitting the slot's description, then *x* filling the slot would be inconceivable.

Whether something is true in the world could be tested by comparison with known examples. Knowledge of how the world is would be much more articulated, much less abstract, than knowledge about what can fill a slot.

Finally, the question of being typical of the world would be addressed as suggested by Winograd [109], by prototypical descriptions of what can fill a slot.

In summary, knowledge of what can fill a slot may involve abstractions both on what is conceivable, and on what is typical. It may also involve a wide knowledge of what actually does fill slots.

6.3 Predicability

Working in the tradition of logic, Keil [45] saw the question of what could fill a slot as the question of what descriptions would accept a given predicate. He worked mainly with one place predicates. Figure 27 shows what happened when a group of college students were given a list of objects and a list of predicates and where asked to indicate which predicates could be applied to which objects. Keil found that some predicates, like IS INTERESTING, could be applied to all of the objects. Other predicates like IS HONEST, were said to apply only to a few objects, here men and girl. The surprising result Keil reports is that if a predicate P_1 applies to an object applied to by a second predicate P_2, then P_1 applies to every object applied to by P_2. This makes it possible to organize the predicates into a tree as shown in Figure 27.

Each node in the tree in Figure 27 defines an abstract class of objects which are accepted by all the predicates at or above that node. In Figure 5.3, Keil has attempted to give names to these classes.

That these classes would form a tree is an unexpected result and certainly deserves closer examination. First note, that if we were describing truth instead of predicability, there would be no chance of getting a tree. For example, Figure 29 shows that taxonomies organized by truth-values are not Hierarchical. For example, while both EXPENSIVE and EDIBLE are true of caviar, neither dominates the other as each is true of something the other isn't. Thus, it is only because we have relaxed the test from truth to predicability that the possibility of a tree exists.

Some apparent counterexamples to a tree can be explained as a confusion of word senses. For example, Figure 30 shows that IS RATIONAL applies to both man and 6, but IS HONEST applies to man and not 6 while IS DIVISIBLE BY 5 applies to 6 and not man. This configuration defeats formation of a tree, but it is eliminated by realizing that two different senses of IS RATIONAL are in question.

Not only is it necessary to recognize different senses of a predicate in order to get a tree, but it is also necessary to assume that some predicates apply only to certain descriptions of an entity. Figure 31 illustrates this. Flag poles and people, but not companies, can be tall, while people and companies, but not flag poles can owe money. A solution here is to distinguish between a person's physical description, which can be tall

Fig. 27. Keil's Taxonomy
<==<fig5-2.press<

Fig. 28. Leif's Taxonomy with Names
<= =<fig5-3.press<

Fig. 29. Truth-Value Taxonomies are Non-Hierarchical
<==<fig5-4.press<

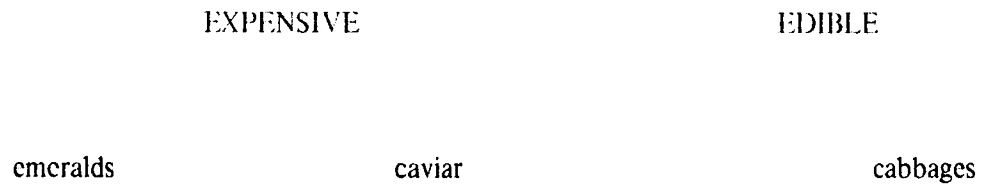


Fig. 30. A Confusion of Word Senses
<==<fig5-5.press<

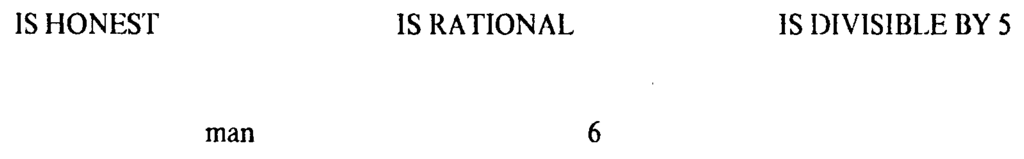
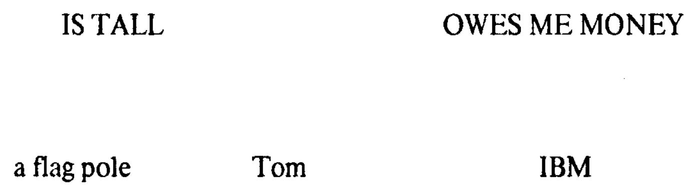


Fig. 31. A Confusion Between Descriptions
<==<fig5-6.press<



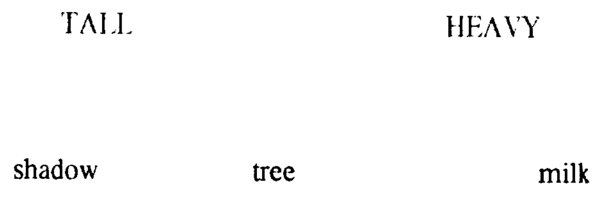
and his description as a social entity which can owe money. Support for this approach will come from the discussion near the end of this chapter. Other examples requiring this trick are

(78a) ?Italy is sunny and democratic.
(78b) ?The fire was red and an hour long.
(78c) ?The song was about birds and an hour long.
(78d) ?The book was about birds and weighed two pounds.

which sound odd because they conjoint two incompatible descriptions. Keil suggests that when one is dealing with multiple descriptions of the same entity it is usually possible to find paraphrases which make these descriptions explicit. For example, the climate in Italy is sunny and its society is democratic.

Unfortunately, not all problems in the construction of a tree are naturally resolved by an appeal to multiple predicate senses or multiple descriptions. Figure 32 shows two such examples. It seems less reasonable to split the description of a tree up into a part accepting dimension and a part accepting weight than to split a person into social and physical characterizations. However, this example suggests a way to save the tree scheme. Predicates like tall and heavy apply to anything which has a single attribute, here height, or weight. It may be possible to either place a predicate in the tree or have it test for one or more particular attributes. This should be explored.

Fig. 32. Incomparable Attributes
<= =<fig5-7.press<



The importance of Keil's observations to computational linguistics is that it opens up the possibility of making a test for what can fill a slot based either on a) being a member of an abstract class, b) possessing a particular feature. Both of these tests can be efficiently implemented as long as the number of classes and features is not too large. One can see from the abstract nature of Keil's classes that there will not be a very large number of classes. There may be more features, and some sort of inheritance might be needed to efficiently attribute all the needed features to each description.

6.4 Prototypes

Predicability constraints can serve as an initial filter on what can fill a slot. At the opposite extreme comes detailed knowledge of the world. A proposed slot filler can be checked with actual experience.

Another use of past experience is in filling slots with default values. It has been suggested that this can be facilitated by the construction of prototypical individuals. For example, imagine a ball, when asked to do this most people will imagine one of their own personal set of prototypical balls. Sometimes its an actual ball they once had and sometimes it isn't. Usually, though, it is representative of balls, it isn't made of ice cream, isn't the world's largest ball etc. Rosch [87] has done experiments to determine what features a prototype of a category will have. The results are as one would expect. A prototype

(79a) Should have features in common with many members of the category,
(79b) should not have features which other members of the category don't have,
(79c) Shouldn't be a good prototype of some other category,
(79d) Should be commonly occurring.

The use of prototypes is not very well understood. It isn't known how abstract they should be, for example. Suppose someone hears the sentence

(80) Harry walked over to the bench and picked up one of the new tools.

How much detail should be filled in on that tool. At its most abstract, the physical description of a prototypical tool might consist only of a handle and a business-end. If more detail is needed, a switch is made to some prototypical type of tool which fits the situation, say a screwdriver. This has a handle and a shaft, but perhaps we don't know if it is a Phillips head screwdriver or not. It seems possible that at each level of abstraction, e.g. tool, we have a list of typical handles, typical blades, etc. and for each of these we also have classes of less abstract tools, e.g. screwdriver which would have a handle of that sort. This allows us to move down in abstraction to whatever level is necessary to get understanding, hypothesizing part from whole and whole from part.

6.5 Is There a Basic Level of Abstraction

Objects in very abstract categories, e.g. animal, have few properties in common. Consequently, to say that something is an animal is not to say very much about it. At the other extreme, objects in very particular categories, e.g. orange tabby Persian, have much in common with other categories. That level of differentiation is rarely called for. Roesch [88] argues that there must be some intermediate level, she calls it the base level, which affords the proper degree of discrimination.

Objects in very abstract categories differ so much there isn't much one can learn which applies to them all. Consequently, one can't tie much of his knowledge to a level like animal. At the same time it is redundant to store all of one's knowledge at a level like orange tabby Persian because most of this could be moved up to Persian, even to cat.

In considering this suggestion the reader might find it helpful to look at Figure 33 and mark what he would consider to be the basic categories. As yet, no one has made effective use of basic categories in computational linguistics. It has been argued that they could be used for matching by setting a commonical level at which matching could be done.

6.6 Types of Abstractions

One frequently possessed both structural and functional knowledge about a concept. For example both structural and functional knowledge are used in accessing category membership. Subjects assessing whether each of the objects in Figure 34 is a cup or a bowl are influenced in their judgment by whether it contains coffee or mashed potatoes.

Even spatial concepts may have a functional interpretation. To explore this, consider the assessment of when a situation may be described as *x is near y*. Three criteria are proposed:

- i) x and y are separated by less than 1/8 (say) the maximum distance evoked by the context, e.g. in a 16 x 16 room a desk 2 ft. from the window is near the window.
- ii) The distance between them is small compared with the size of the objects — measured in any way you feel appropriate.
- iii) The situation is recognized as fitting a frame in which there is a threshold distance required for some process — near is under this threshold.

The reader might want to test these in the following sentences.

Fig. 33. Taxonomy of Basic Categories

- PHYSICAL-OBJECT
  - TOOL
    - HAMMER
      - BALL-PEN-HAMMER
      - CLAW-HAMMER
    - SAW
      - HACK-SAW
      - CROSS-CUT-SAW
  - TREE
    - MAPLE
      - SILVER-MAPLE
      - SUGAR-MAPLE
    - OAK
      - WHITE-OAK
      - RED-OAK
  - CLOTHING
    - PANTS
      - LEVIS
      - DOUBLE-KNIT-PANTS
    - SHIRT
      - DRESS-SHIRT
      - T-SHIRT
  - FISH
    - BASS
      - SEA-BASS
      - STRIPED-BASS
    - TROUT
      - RAINBOW-TROUT
      - STEELHEAD-TROUT

Fig. 34. The Cup/Bowl Continuum
<= =<fig5-9.press<

(81a) His office is on the 8th floor, near Minsky's.
(81b) Don't stand near the tracks.
(81c) The dog was near the telephone pole.
(81d) The bug was near the telephone pole.
(81e) The dog was near the fire hydrant.
(81f) The two pages were near each other in that 400 page book.
(81g) The mark was near the end of the yardstick.
(81h) Apartment for rent, near MBTA.

Some concepts seem to be abstractions primarily along structural lines, e.g., square, some along functional lines, e.g. container, and some along both, e.g., mammal.

6.7 Analogy and Transfer frames

Winston [110] suggests that understanding an expression like *Robbie is like a fox* involves a transfer of structure from fox to Robbie as illustrated in Figure 35. Close relatives of the source and destination frames are used to help guess the intended analogy. Not only are properties transferred, but a transfer frame is constructed. This can be used for similar analogies in the future. Winston's algorithm is shown in Figure 36.

The transfer frame is a key idea, because it allows us to record, as data, how one frame is to be related to another.

6.8 Word Sense Extension

When something new comes along which doesn't match any of our concepts too well, we may extend or specialize the sense of a given concept. Suppose that the prototypical HOUSE is one for humans. It is misleading to refer to a house for dogs as a HOUSE, because it differs from the HOUSE prototypes in too many ways. What English lets us do is to indicate by DOG HOUSE that the HOUSE concept is to be modified in some way so as to make it appropriate to DOG. As Marchand [59] points out, this process of word formation is not limited to nouns, e.g., *color blind*, and *hog tie*. We postulate that when someone hears a term like *dog house*, he can use some mix of the following three strategies in understanding it.

Fig. 35. Winston's Transfer Frames
<== <fig5-10.press<

The basic idea behind the theory of learning presented in this paper. The teacher specifies a source and a destination and possibly the slots that are relevant. The student analyzes the source, the destination, and other aspects of the situation to discover and use a transfer frame.

Fig. 36. Winston's Algorithm
<= =<fig5-11.press<

Overall organization of the hypothesizing and filtering methods. Hypothesizing methods are tried until one produces one or more slots that are not filled in the destination. After grouping into transfer frames, all filtering methods are used in an effort to reduce the number of surviving transfer frames. Filters have an effect only if they recommend dropping some, but not all of the transfer frames that they see.

a) He may have a concept for which he knows this to be an unanalyzed name. e.g.. *bull dog* or *skid row*. How many people know that a bull dog is a dog for fighting bulls or that skid row was a row of shanties along a log skid in early day Seattle, Washington. We know what horse shoes and alligator shoes are.

b) He may have a concept for which this would be an appropriate name. For example, although people don't have a name for the little plastic cylinder on the end of a shoestring [116] they know of it and they can understand the sentence *my shoestring end came off*. This presumably because END could be extended in the context of SHOESTRING to mean this cylinder, and this can come off.

c) He may create a concept by principled analogy. For example, by analogy with *dog house* and *bird house* one could understand *grasshopper house*. We contend that this process of analogy is always at work. We understand *crackage* by analogy with *breakage, pilferage, damage*. We understand *inchage* by analogy with *mileage* and *yardage*.

6.9 Naming Concepts

It is interesting to ask whether this process of word formation might be formalized as a way of getting a systematic method of naming concepts.

McDermott [67] has suggested that concepts should be given names in some invented language, lest we confuse our understanding of the name with an understanding of the concept. It seems unrealistic, however, to ask someone to learn several thousand names in order to use or evaluate a system. McDermott's suggestion seems practically limited to small or one person systems. For others, it will be necessary to name concepts using terms the user is familiar with, either directly or by supplying a translation. Since a natural language, such as English, is the only widely known form of expression for non-mathematical concepts, it is the only real choice for concept names. Most all semantic networks do, in fact, use English in creating their node and link names. The issue is to what extent this can be done in a principled way.

In a semantic network concepts can be named and they can also be individuated (or defined) by their relationships to other concepts in the network. When a person is studying one concept in the network it is extremely helpful if he can get at least a general idea of what the concepts are that it is related to in the network just from their names. Otherwise mapping the semantic networks onto his knowledge of the world becomes a combinatorial puzzle. Fortunately, this can be made possible by capitalizing on a natural skill of every English speaker, the ability to extend a sense of a word for use in a particular context.

The problem with using expressions like *dog house* as the names of concepts is that they can be ambiguous. For example, *snake poison* can mean either poison from snakes or poison for snakes. A *steel drill* can be either made from steel or for drilling steel. A *river bank* could be a financial institution.

The ambiguity is bad for two reasons. First the user does not know which concept is meant. While this is bad, the user can always study the semantic network in order to see how the concept is used and thus what is meant. A worse problem is that the naming scheme does not generate enough distinct names; it proposes the same name for two different concepts. We wish a scheme which will allow us to generate a distinct name for each concept.

The ambiguity in *river bank* arises from the ambiguity in *bank*. It seems reasonable to resolve this by saying that names will be constructed from senses of words instead of words. Thus, RIVER BANK and RIVER FINANCIAL-BANK. The ambiguity in *snake poison*, however, does not seem to arise from any ambiguity in *snake* or *poison*. Instead it arises from two different relationships between these terms. We have already suggested in Section 6.8 that a person uses his knowledge of the world in order to determine what relationship is meant. It has long been recognized that there are certain regularities to this process. For example, corresponding to any substance such as steel, a *steel x* can mean an x composed of steel.

Rhyne [90] discusses attempts by various authors to capture these regularities in the case of compound nouns. These may be divided into attempts to classify the semantic relationship between the two constituents of the compound and methods for deriving the compound from a relative clause or other expression.

The difficulty with expressing the relation between the elements of a compound by a linguistic expression may be seen by considering *widget store*. Everyone knows what stores are for and so one meaning of *widget store* is clear. But how should this be expressed by a relative clause. Is it a store where widgets are sold, a store which carries widgets, a place to go to buy widgets, etc.

We postulate that somewhere in the semantic network there is a concept which corresponds to the store's wares. A *widget store* is a store where this wares concept may be characterized as widgets. There may be many different relative clauses which can express the relationship between the store concept and its wares concept, since in general they will be linked by a network of different relationships.

Loosely speaking, our solution to this problem will be to take names as triples. The second element of a triple will indicate how the first and third are related. In an example like *widget store* the second element will indicate how WIDGET and STORE are related by providing access to the store's WARES concept mentioned above.

6.10 A Model of the World

Whorf [103] has written a number of essays supporting the view that the language of a culture determines (or demonstrates) their model of the world. He believes that different languages show that quite a wide variety of models are in fact workable for common sense reasoning. Describing the Hopi language, he says:

> After long and careful study and analysis, the Hopi language is seen to contain no words, grammatical forms, constructions or expressions that refer directly to what we call 'time,' or to past, present, or future, or to enduring or lasting, or to motion as kinematic rather than dynamic (i.e. as a continuous translation in space and time rather than as an exhibition of dynamic effort in a certain process), or that even refer to space in such a way as to exclude that element of extension or existence that we call 'time' and so by implication leave a residue that could be referred to as 'time.' Hence, the Hopi's language contains no reference to 'time' either explicit or implicit.

The metaphysics underlying our own language, thinking, and modern culture (I speak not of the recent and quite different relativity metaphysics of modern science) imposes upon the universe two grand COSMIC FORMS, space and time: static three-dimensional infinite space, and kinetic one- dimensional uniformly and perpetually flowing time — two utterly separate and unconnected aspects of reality (according to this familiar way of thinking). The flowing realm of time is, in turn, the subject of a threefold division: past, present, and future.

The Hopi metaphysics also has its cosmic forms comparable to these in scale and scope. What are they? It imposes upon the universe two grand cosmic forms, which as a first approximation in terminology we may call MANIFESTED and MANIFESTING (or UNMANIFEST) or, comprises all that is or again OBJECTIVE and SUBJECTIVE. The objective or manifested comprises all that is or has been accessible to the senses, the historical physical universe, in fact, with no attempt to distinguish between present and past, but excluding all that we call future. The subjective or manifesting comprises all that we call future, BUT NOT MERELY THIS; it includes equally and indistinguishably all that we call mental — everything that appears or exists in the mind, or, as the Hopi would prefer to say, in the HEART, not only the heart of man, but the heart of animals, plants, and things, and behind and within all the forms and appearances of nature in the heart of nature, and by implication and extension which has been felt by more than one anthropologist, yet would hardly ever be spoken of by a Hopi himself, so charged is the idea with religious and magical awesomeness, in the very heart of the Cosmos, itself. The subjective realm (subjective from our viewpoint, but intensely real and quivering with life, power, and potency to the Hopi) embraces not only our FUTURE, much of which the Hopi regards as more or less predestined in essence if not in exact form, but also all mentality, intellection, and emotion, the essence and typical form of which is the striving of purposeful desire, intelligent in character, toward manifestation — a manifestation which is much resisted and delayed, but in some form or other is inevitable. It is the realm of expectancy, of desire and purpose, of vitalizing life, of efficient causes, of thought thinking itself out from an inner realm (the Hopian HEART) into manifestation. It is in a dynamic state, yet not a state of motion — it is not advancing toward us out of a future, but ALREADY WITH US in vital and mental forms, and its dynamism is at work in the field of eventuating or manifesting, i.e. evolving without motion from the subjective by degrees to a result which is the objective. In translating into English, the Hopi will say that these entities in process of causation 'will come' or that they — the Hopi — 'will come to' them, but, in their own language, there are no words corresponding to our 'come' and 'go' that mean simple abstract motion, our purely kinematic concept. The words in this case translated 'come' refer to the process of eventuating without calling it motion — they are 'eventuates to here' (pew'i) or 'eventuates from it' (angqo) or 'arrived' (pitu, pl. oki) which refers only to the terminal manifestation, the actual arrival at a given point, not to any motion preceding it.

If Whorf's analysis is correct, it shows that there is more than one workable world model. Hence, we may regard the world model that has evolved for use in English as somewhat arbitrary. Suppose we construct a formal interpretation of the world model of English speakers. This will be somewhat arbitrary, though we might hope that that arbitrariness is comparable to the arbitrariness of interpretation of a typical competent and intelligent English speaker. Such a world model may, nevertheless, be quite useful, particularly in a heuristic rather than a scientific sense. As with Hopi, we may find it alien to our thinking rather than ineffective for the intended use. This is because we are not used to a computer using animistic and metaphoric rather than scientific concepts. On the other hand we will argue that the speaker of English lives in a metaphorically rich environment. It is only because he is trained to regard his models as "idioms" or non-scientific ways of thinking that the Hopi model seems so different.

We feel that English resembles Hopi in that it too uses a <u>change</u> <u>model</u> as one of its most basic notions. The form of the change model is very simple and has been noticed by Schank [95], among others. Something goes from an <u>initial</u> <u>state</u> along a <u>trajectory</u> to a <u>final</u> <u>state</u>. In its most general form, this *going* is not in physical space, but in an abstract state space, that is, the object going along the trajectory is undergoing a change of state, represented by a change in one or more of its properties. Only if its LOCATION is changing is an object *going* in physical space.

Looking in the dictionary at the definition of go we find such sentences as

(82a) The car is going to my house.
(82b) The car is going to pieces.
(82c) The car is going to run.
(82d) The firm is going to the dogs.

(82e) The motor is going.
(82f) Tom and Sue are going steady.

The preposition to typically signals the destination. In the first four sentences the destination shows us what type of state change is involved. As shown the change could be a location, a fragmentation, a change in activity, or in a more abstract state. The notion *go* is that of a predictable directed change. A motor that is *going* is not just moving, but is moving in a systematic predictable way. If a couple is going steady, they are acting in a systematic predictable way with respect to one another or combining to form a functional unit.

A hat and dress can *go together.* This meaning of go, which takes a plural object, seems to mean combine to form a functional unit. Here, the final state appears to be the combination of the two objects. This meaning is, however, derivative of the one above. In fact, *go* is not suitable as the word for our notion of change. For example, it contains a default notion of *away* (as opposed to *come*). We will introduce the concept TRANS of which GO is a specialization.

An object can be accompanied on the trajectory. This can be indicated by with. Also, there can be an AGENT which is responsible for the change.

Although one can think of the change model as underlying every change described in English, the typical sentence describes only certain aspects of the change. Since the OBJECT, SOURCE, TRAJECTORY, and DESTINATION are slots in a TRANS frame, one may optionally include all of them in a sentence using go:

(83) John went (from the house) (through the gate) (to the store).

Here, the SOURCE, TRAJECTORY, and DESTINATION are used to form the initial state, location of John at house, trajectory, and final state, location of John at store, of TRANS. Almost all verbs, however, require the user to be less general and complete. They

a) Focus on only part of the change.

b) Specialize the change so that it can apply to only particular kinds of states, e.g., LOCATION.

For example, the verb *put* concentrates on the final state. If I say that I put the pencil in the desk then I mean I caused the final state. Note that *put* doesn't even allow us to mention the SOURCE or TRAJECTORY. One cannot use *put* without implying that the final state has been reached. On the other hand, one can *send* an arrow into the air with no implication of a destination. Send means *cause to go, without going with.*

When someone dies (goes to his reward) then he has experienced a particular change of state, and he has completed it. (The destination state is known as death. We are *close to death.*) We don't have to say *He finished dying.* This notion of completion of state change is an important one. When one begins eating dinner, how much does he have to eat before he has eaten dinner? From the point of view of the eater, we ask if he is finished eating. From the point of view of the dinner, we ask if he ate the dinner up. Notice the difference between:

(84a) He ran to the tree.
(84b) He ran for the tree.

This use of for marks a destination of the state change, as does to; but, unlike to, for does not imply the reaching of that destination.

At the most primitive (or highest) level of abstraction, let us postulate two models: the state change model and the life history of a thing. An object can come to exist, but if it is not also available, and operational it is of reduced importance. There is a notion of being in the collective center of things. Since conservation of matter holds, objects are not caused to exist out of nothing. Rather, they are either a) brought into the center of things, b) noticed for the first time even though they have already been there, or c) formed from something else. For example an object can appear and then disappear. It can be produced (L. producere: pro- forward + ducere, to lead, draw) or disbanded. It may be born into the world or carried out. Something goes from raw material to a finished product, or something goes out of one form into another.

(85a) I made a dress from cloth.
(85b) I made a dress out of cloth.
(85c) I made the cloth into a dress.

Once something exists, it is generally not "un-made". Rather, the end of its useful life is remembered by the state change which moved it out of the center of things or transformed it so it is no longer suitable for its old role. If John eats a piece of cake and someone asks *where is that piece of cake?* the answer is usually *John ate it,* not *inside John.* The cake still exists in a sense, but it is *gone.* A typical answer to *Where is that jar?* might be *I threw it out.* Out where? Out of the center of things. Another answer might be *I made a bird house out of it.*

6.11 Micro-Worlds

In a way, life is quite simple. Agents use information to form plans and carry them out. To describe this, English relies on analogies to the physical world! It is as if we only have one set of conceptual machinery, this being suitable for describing configurations of objects and their motion in a three dimensional setting. The more abstract concepts must then be fit to the same machinery. Our representation of them winds up as a vast complex of analogies to the physical world.

We define the physical world to be made of PHYSICAL-OBJECT's and their physical attributes. Basically, a physical object can be characterized as:

(86a) something made of matter, or
(86b) a "picture producer" (Schank's terminology), or
(86c) something made of the "four elements": earth, air, fire, and water, or
(86d) something "concrete" as opposed to "abstract".

By physical aspects we mean those which are directly observable, like size and color. Time advances in the physical world, and the activities which are aspects of physical objects take place. At any point in time, the physical world is completely described by listing all of the objects in it with their physical aspects.

As a thinking being, it is the goal of an English speaker to predict what will happen next in the physical world. This may be an explicit prediction, or it may be only implicit, in the sense of appropriate reaction to previous events. It is fair to judge a world model, be it in a computer or an English speaker, primarily in terms of its predictive capabilities. For this reason, the notion of AGENT of a change is of utmost usefulness. Even if we do not model the motivations of the AGENT, observed sequences of acts by the same AGENT give us predictive power. In the sentence, *The dog hurt the man,* we have an abstraction of the dog as an agent of hurt, but what are we to do with *The government hurt the man*? In the physical world there is no *government.* However, the set of physical events that *hurt* the man become more understandable if the abstract concept of *government* is specified as agent. Formalization of this leads us to the notion of the system world, a world of entities which serve as agents. Some entities in this world and their aspects are shown in Figure 37.

There is a hierarchy of agents which may be discovered using the pattern "x was given y". For example, *the dog was given the bone* and *the bone was given the dog* both mean the same thing because dogs come above bones in the anthropomorphic hierarchy. An entity in the system world can be a system abstraction of an entity in the physical world, or it can be unique to the system world. For example, in the sentences

(87a) The man carried the item.
(87b) The store carried the item.

we see that in the first sentence carry is ambiguous between the physical meaning of carry and the system world meaning of carry appearing in the second sentence. Store selects a meaning of carry in the system world, and man tends to select the meaning in the physical world, though the system abstraction of man allows the system world interpretation. Note that there are also physical world stores (a building set up in a certain way) but they cannot carry anything. Note that there is a certain analogy between the two meanings of carry. In fact, there is quite a strong analogy between the physical and system worlds. The most richly described system in the system world is the social system and its various subsystems. "Locations" in the social system are:

(88a) on the social scene
(88b) in an organization
(88c) in a social class
(88d) at the top of an organization
(88e) under a bad boss

The physical location prepositions are still used, but their social abstractions have less meaning. One can use these locations in analogies with the physical world like

Fig. 37. Entities and Properties in the System World
<==<fig5-12.press<

(89a) He came from the lower class.
(89b) he dropped out of school.
(89c) He got out of trouble.
(89d) He followed a career path in the organization.

Some aspects in the systems world are FAMILY-RELATIVE, CUSTOMER, BUSINESS, OCCUPATION, RESPONSIBILITY, and AUTHORITY. Corresponding to PART in the physical world, one can be a PART of an organization. A kind of a PART is a MEMBER. An organization can have a top, a bottom, arms, and a head. In addition, the systems world is the locus of "possession". In the physical world we observe the following relations between "have" and "location".

(90a) There is a cookie in the box.
(90b) The box has cookie in it.
(90c) The box contains a cookie.

(91a) There is a cookie on the plate.
(91b) The plate as a cookie on it.

(92a) There is a tree on the mountain.
(92b) The mountain has a tree.

(93a) *There is a cookie at the ice cream.
(93b) There is a cookie with the ice cream.

(94a) There is a dog at the house.
(94b) *There is a dog at Jim.
(94c) There is a dog with Jim.
(94d) Jim has a dog with him.

<u>Action</u>	<u>Result</u>
I take the dog to the tree.	He is at the tree.
I take the dog to Jim.	Jim has him.
I take the dog.	I have him.

From this we argue that to "have" an item is the physical world means for it to be located there. If the location is *in*, then the <u>have</u> can be further specified to <u>contain</u>.

By analogy, *to have an item* is specialized in the systems world to *to <u>possess</u> an item*. An item is located at a person or organization in the sense of possession. <u>Possess</u> can be further specialized to <u>own</u> or <u>control</u>. If someone controls you, you are <u>under</u> him. These specializations of have were pointed out by Charniak [13]. If my father gives me the car for the evening and I say, *I have the car*, I do not mean that it is physically *at me*, nor do I mean I possess it. I merely control it temporarily.

In this light we can associate give, get, and take with changes of having, recognizing the relationship between have and location and the fact that getting something to own can involve buying it as in

(95) I got this car for a song.

Agents take actions to get credit for them. If one does an action for somebody else, then the credit for that action goes to him. An agent hopes for success, not failure, in an organization.

In the system world, we describe all possible agents and the organizational constraints placed upon them. In order to better predict how these agents will act, we need the notion that some of these agents, namely animals and organizations, carry out actions in accordance with plans (possibly instinctive). Again, this is handled by analogy to the physical world. Thus we have the world of plans and actions. In addition to the plans and actions abstraction of agents, this world contains the entities: plan, intention, alternative, goal, resource, action and mistake.

Activities like maintain, insure, limit, and trick are abstract activities used only in the description of plans. Agents decide what alternative plan to follow. Wise agents make good decisions. Using resources, agents take actions in accordance with plans which they have. They have reasons for the decisions and actions. They carry the plan through to completion, they reach their goal, or they make a mistake. An action may be easy or difficult. An agent may encounter a difficulty. In this world an agent is located with respect to actions. He is *in the middle of dinner, in surgery, on trial,* or *under investigation.* A resource is *in use.* This analogy is much less complete than the systems world.

The mental world is even less complete. The mental world is the source of plans and reasons. It contains situations, problems, concepts, names, facts, ideas, views, and opinions. The mind is added as a location for these. The only agents in this world are the mind and the attention. Something can be located

(96a) in mind
(96b) on one's mind
(96c) in the back (or front) of my mind
(96d) on my tongue
(96e) on the tip of my tongue
(96f) in my head
(96g) on a solid surface (He put the idea on paper).

The attention can be drawn to a problem or the problem can be brought to the attention. One thinks over a situation in order to discover a problem and solve it.

It is not clear how to classify the perception and communication of information, although it probably arises in the social world. This is done by the agents of the social world, but to understand their use of it we must move to the mental world. How should we analyze *She solved the problem with a suggestion from Bob*?

7. Procedures

7.1 Procedural Representation of Knowledge

It is useful to distinguish between procedural representation of knowledge and representation of knowledge about procedures. By procedural representation of knowledge we will mean casting that knowledge in the form of knowledge about the results of a procedure. For example, the knowledge that every boy wants a lion could be cast in the form: "if you check every boy and count those who want lions, then the count of those who want lions will be equal to the count of the boys checked."

Woods [115] introduced a FOR iteration construct for representing knowledge of quantified propositions procedurally. Examples of the use of this construct are:

(97a) (FOR EVERY X / CLASS : (P X) ; (Q X))
Every X in CLASS that satisfies P also satisfies Q.
(97b) (FOR SOME X / CLASS : (P X) ; (Q X))
Some X in CLASS that satisfies P also satisfies Q.
(97c) (FOR GEN X / CLASS : (P X) ; (Q X))
A generic X in CLASS that satisfies P will also satisfy Q.
(97d) (FOR THE X / CLASS : (P X) ; (Q X))
The single X in CLASS that satisfies P also satisfies Q.

Woods assumes that for each CLASS there exists an enumeration function which will produce all the members of the CLASS. The FOR statement specifies that the predicate P(x) be applied in turn to each element adduced by the enumeration function. The predicate Q(x) is then applied to those for which P(x) is true.

The first argument of the FOR divides the statements into those (EVERY, THE EQUAL GEN) which exhaust the enumeration function and those (SOME, ORDINAL, GREATER) which enumerate either until exhaustion or until a termination criterion is satisfied. The description of the termination conditions and the resulting states assumes that, in general, three counts are maintained.

(98a) a count C of the elements enumerated so far.
(98b) a count CP of the elements passing P(x) so far.
(98c) a count CPQ of the elements passing P(x) and Q(x) so far.

For example, the (FOR EVERY ...) statement says that in the situation resulting from its execution CP = CPQ.

The FOR iteration construct allows knowledge about sets to be built up from knowledge about individuals contained in the sets. As was mentioned above, such a composition operation should not be used to the exclusion of decomposition. A decompositional approach allows predicates on sets of individuals rather than just predicates on individuals as is done here; predicates on individuals are inferred from predicates on the sets they are in. Both composition and decomposition are needed.

The sentence *every boy wants a lion* has two referential readings. On the first, they all want the same specific lion. On the second, each wants his own, possibly different, specific lion.

By converting Wood's FOR construct to our notation, the difference between the two referential readings of *every boy wants a lion* can be spelled out in more detail as shown in Figures 38 and 39. Figure 38 shows the situation where each boy wants a different lion. This reading can be represented by a structural description in which lions are, in effect, individuated by the boys who want them. In doing this, there are two ways that *every boy wants a lion* could be understood.

(99a) If you pick any boy, he wants a lion. This is the GEN option of the FOR iteration.
(99b) If you iterate through the boys and check each, each will want a lion. This is the EVERY option of the FOR iteration.

In this example, either option could be picked. The use of the quantifier *every* (as contrasted with, for example, *any boy wants a lion*) suggests, but does not force the EVERY option. In Figure 38, a FOR iteration with the EVERY option has been chosen. This iteration is represented by the node FOR-EVERY-BOY/BOYS:T;WANT-BOY-LION-1. This node has a generic role BOY which always describes the boy for the current iteration, just as *the President* describes whoever is currently President. Given BOY there is always an individual wanting of a lion, WANT-BOY-LION-1, unique to BOY, and given this wanting, there is a unique lion, LION-1. We can show the relationship between the sentence and the iteration by making LION-1 a co-characterization of the sentence structure node a-lion-1.

FOR-EVERY-BOY/BOYS:T;WANT-BOY-LION-1 can be matched to the world if WANT-BOY-LION-1 can be matched to the world uniquely for every possible match of BOY to the world.

The reader might note here how we have utilized the notion of a description only being singular with respect to some structure. LION-1 is unique only within BOY. As BOY describes different boys, the properties of LION-1 are inherited to different lions. Note also that this formulation allows the same lion to be wanted by more than one boy, the standard understanding of such a quantified expression.

Fig. 38. Each boy wants a different lion
<==<fig6-1.press<

Fig. 39. Each boy wants the same specific lion
<= =<fig6-2.press<

In this example, WANT-BOY-LION-1 is made a role in BOY. This is the first time we have seen a node representing a process made a role in the structural description of a physical object. In all the previous examples, any physical object node was made a role in a state or process node. The following question is raised:

(100) If someone wants something, is that wanting an attribute of the wanter, or does the wanter fill a role in the wanting process?

The same state of affairs can be described in either way. Since the lion is to be individuated by the boy, we here choose to view the wanting of the lion as an attribute of the boy.

In contrast to Figure 38, Figure 39 contains no iteration at all. Figure 39 represents the case where all the boys want the same lion. Since in this case nothing is individuated to individual boys, there is nothing to be gained by expanding the sentence structure into an iteration in discourse structure.

7.2 Declarative vs. Procedural Representation

One of the central issues in the representation of knowledge is the question of declarative vs. procedural representations. Some aspects of this issue have been discussed by Winograd [109]. The nature of the declarative/procedural question can be illustrated by a simple example. Suppose we have the fact that all Chicago lawyers are clever. Then for all X, (Chicagoan (X) and Lawyer (X)) implies Clever (X). This fact can be used in several different ways. If strictly procedural representations are used, a different representation would be needed for each use. For example, "to find out if someone is clever check to see if he is a lawyer, then check whether he is from Chicago," or "to find out if someone is a lawyer, check whether he is stupid, then check whether he is from Chicago, if so he is not." If a strictly non-procedural representation is used, only one representation need occur.

The nature of current computers is such that any non-primitive entity will affect computation only through the execution of processes that consist ultimately of simple machine instructions. There are basically two ways to get from entities to the appropriate corresponding series of machine instructions automatically: interpretation and compilation. Everyone is familiar with these notions, but we have to consider them in this broader context.

It may be objected that both interpreters and compilers add procedural detail to procedural statements. This is currently true, but it is not an essential part of the notion. It is true that some facts are extremely difficult to represent in a totally non-procedural way. However, a single very abstract plan may provide the necessary procedural structure and yet suffice for all uses.

It is clear that a higher level plan offers the possibility of more flexibility, since it can be fleshed out in many different ways. It also makes explanation easier, since the main steps are brought out and not hidden in detail. By reducing the number of representations of a fact, it also simplifies the problem of updating and changing incorrect facts. Since such a plan is clearly desirable, we must ask if the difficulties of writing interpreters and compilers for plans can be overcome.

One could say that no matter how good it is, an interpreter or compiler for plans is bound to be very slow. The answer to this problem is to keep the same knowledge around at more than one level of abstraction. Knowledge used frequently in a particular form will be compiled. Compilations of a plan can be kept as specializations of the plan. The compilations need not be done automatically; they can be done manually or semi-automatically.

A second potential problem is that very abstract plans do not specify exactly what will happen. The interpreter or compiler fills in so much on its own that the exact results are hard to predict. Undoubtedly this will happen, and thus it is essential that the system be able to explain what it has done and respond to user suggestions. The use of suggestions is much easier in a well-structured problem area because there it is much easier to formulate them in a way that makes clear where they apply. It may be that plans will not work in poorly formulated areas.

We also have to ask the straightforward question of whether the interpreter or compiler will be able to do the job. We are asking the system to do the problem solving. Most general problem solving systems die in combinatorial explosion. The answer is that the interpreter or compiler will not be up to the task unless it uses the kind and amount of knowledge that is currently put into hand-coded programs. There are two ways of giving an interpreter such knowledge: building it directly into the interpreter or providing it as a set of facts about a situation. Which way is best depends on the knowledge involved. For example, in CONNIVER, Sussman and McDermott [68] replace the pure back-up mechanism of MICRO-PLANNER with a programmer-specified procedure, incorporating the programmer's search ideas into the search pattern and the editing of the "possibilities list". It appears to us, however, that it would actually be preferable to decouple the programmer's suggestions from the basic search pattern. CONNIVER implicitly asserts that the primary method of reacting to the failure of a possibility is a more intelligent selection from the remaining possibilities on the list. In fact, a more global reaction is often required. For example, as Sussman says, if a robot has selected the next block to add to a construction and finds the block hot, the response may be to find a procedure to cope with hot blocks, rather than to select an alternative block. Similarly, our solution to

Sussman's problem of conflicting goals (discussed in more detail later) is to reorder actions at a higher goal level, based on a global understanding of the difficulty. What we want to do is to "backoff" and decide what goal or goals best formulate a problem space for dealing with the current difficulty, not to always focus on the goal immediately superior to where the difficulty occurred. For this reason it appears better to separate the advice from the original search strategy. The advice is organized around the trouble encountered, rather than the original plan. It is not, however, necessary to build such advice into the interpreter.

There are, however, certain very basic notions that must be built into the interpreter in order that it not be unbearably inefficient. Much of this knowledge is what Winograd calls second order knowledge, for example, "The relation NEAR is transitive as long as you don't try to use it too many times in the same deduction." Another example would be an interpreter's knowledge of how to evaluate or carry out an AND. AND does not necessarily imply any special order of evaluation as it does in LISP. The interpreter must nonetheless choose an order, based, in part, on user suggestions as will be demonstrated later. In general, the interpreter must have the basics of a particular world model built into it, including notions such as time and space and how to weigh evidence. This model serves as a framework for specific user suggestions in these areas.

7.3 Factorization of Knowledge and the Flow of Control

In the first section of this chapter the notion of procedural representation of knowledge was introduced. As an illustration, a procedural representation of a reading of a sentence containing the quantifier "every" was given. The second section then claimed that a procedural representation of knowledge often contains more detail, and less flexibility of interpretation, than a more abstract declarative representation of the same information. It went on to suggest the construction of an interpreter which worked on abstract plans from which much of the procedural detail had been removed. Writing programs for such an interpreter would be different than using a conventional programming language.

It is actually possible to identify a number of different proposals for how procedural knowledge should be represented and interpreted. These proposals differ not just in how knowledge is factored or abstracted, but also in how it is determined when a procedure should be executed — that is, in the flow of control. Before examining some of these proposals in more detail, it may be helpful to sketch the major dimensions on which they differ.

Figure 40a shows a conventional CPU accessing a passive memory. This model describes both the execution of a conventional programming language and also the operation of the proposed abstract plan interpreter. It has the disadvantage that only serial searches of memory are possible. To permit faster searching of memory, Fahlman [24] has proposed a machine represented schematically in Figure 40b. Memory consists of a large number of agents capable of propagating signals to other agents over a fixed set of wires, not connecting every agent to every other agent. A central CPU sends the same control signal to every agent, so that all agents perform the same operations in lock step. In this design the CPU still plans the search strategy as in (a), but execution is speeded by parallel operations.

Fig. 40. Four Architectures
<==<fig6-3.press

A yet more decentralized approach is being investigated by Minsky [72]. His ideas are not yet well worked out, but there are several principles of organization he intends to use. There is no central CPU. Agents communicate by placing arguments on fixed, shared bus wires. Interconnections are reduced by using a hierarchical organization.

Yet another proposal attempts to combine some of the features of (a) and (c). As illustrated in Figure 41d this proposal uses a CPU and memory, but some of the CPU's requests to memory activate independent agents which are able to make specific types of calculations. A key decision in this design is whether to let agents activate other agents by writing back in memory. If this is done, and the CPU removed, we have the CMU "blackboard" paradigm.

Approaches involving multiple problem solving agents are attractive because of the additional speed gained. However the problem of co-ordinating multiple agents has not yet been solved except for simple cases. Indeed, the problem has not even been adequately posed. A second challenge for the multiple agent systems is finding a practical way to idiosyncratically connect large numbers of agents.

It should be clear that using multiple agents requires factorization of the problem solving procedures while using a single CPU does not. Factorization nevertheless may be desirable with a single CPU for reasons of flexibility.

7.4 Evaluation

The most commonly used language in artificial intelligence research is LISP. LISP has stood the test of time. In attempting to invent a new language, one approach is to look for ways of generalizing or extending LISP, the tried and true. The hope would be to retain what is good about LISP, while seeking to do more. The possibility of success stems in part from the vastly cheaper hardware expected in the future.

There is already a family of LISP-like languages, e.g. MICROPLANNER, CONNIVER, QA4, PLASMA, and SCHEME. But the question of LISP extension is far from exhausted. Here are some additional ideas.

In LISP a procedure is written as an expression. Rules are given for the evaluation of expressions by the LISP interpreter. Simplifying somewhat, expressions may be defined recursively as follows:

(101a) An atom is an expression.
(101b) A list of expressions is an expression.

Evaluation of expressions may thus be defined by giving rules for

(102a) evaluation of atoms,
(102b) evaluation of lists.

In LISP, atoms are treated as variables whose values are maintained in an environment. There is an option of creating the environment either by lexical or dynamic scoping, or of declaring variables to be either lexical or dynamic and maintaining two environments.

Lists are treated as a function to be applied to arguments. The first element of the list either names or evaluates to a function. The remaining elements of the list are the arguments. There is an option of whether the arguments are or are not evaluated before they are passed to the function.

If procedures are to be represented in a semantic network in a notation analogous to LISP, then nodes of the semantic network must be designated as either a function or variable node. When the interpreter reaches a node, it evaluates it either as a function or a variable. One could also explore the possibility of a class of nodes on which the interpreter attempts variable evaluation and if this fails, goes on to try function evaluation.

The obvious way to pass arguments with a function node is as predicate argument relations. For example, let hit-1 be a function node, in Figure 41 hit-1 is shown as having an object role (a predicate argument) which is characterized as ball-1. The definition of the function hit-1 could be inherited from the generic hit node. This node is shown as having an individual object role, so we know that any instantiation of hit can have an individual object role. Notation for the definition of a function will be discussed in the next section.

To evaluate a variable node, the interpreter also needs an environment node. There are several ways that the value of a variable could be located in the environment.

(103a) Shallow binding: The value of a variable is linked directly to the variable. When shallow binding is used in LISP, the current value of the variable is linked to the atom representing the variable. When recursion causes the variable to be rebound, the old value is saved on a push down stack and then restored to the atom when the recursion is complete. It would also be possible to create a new node, representing a new instance of the variable whenever there is recursion. Each new value is stored with a new instance of the variable. If this is done, we have the problem of finding the current instance when a variable is to be evaluated. The situation is analogous to deep binding in LISP, which is discussed next.

Fig. 41. A Hitting and a Ball
<==<fig6-4.press<

(103b) Deep binding: The value of the variable is found by searching the environment in a prescribed way, using the variable as a key. In LISP, deep binding has been implemented with an association list. This is a list of lists of a variable and its value. When a variable is rebound a new list of it and its value is added to the association list. The association list is searched in a last-in first-out order. The addition of a new list of a variable and its value to an association list is similar to the creation of a new instance of the variable. Analogous to maintaining and searching back up an association list in LISP, we could maintain an explicit representation of the tree of events created when particular executions of procedures call subprocedures. Each of these events would have its corresponding instances of the variables linked to it and variable evaluation would be done by searching back up the event tree.

An alternative approach would be to store on each variable all of the values which it has ever had, and then somehow describe for each of these values the conditions under which it is the value of the variable. In data base terms, an inversion of the first method would be an example of this second method.

(103c) Pattern matching: We could treat a variable as a node whose roles form a structural description which must be matched by its value. The value would have to be an individual in the environment. This method is often restricted to variables which specify the set of all items of such and such a form.

(103d) Path addressing: Variable A is taken to specify a variable B and a path. The value of variable B is found and then the specified path is followed from the value of variable B in order to find the value of variable A.

(103e) Attributive values: If one is told to build a house, then the call to the build function has an object which is the concept of the house we are building. A node for this is just created, and then as the house is financed, sold, or constructed, more description can be added to this node.

(103f) Alternative notions of value: For example, we might have actual and default values. If no actual value could be found, a default value would be used.

7.5 Pattern Directed Invocation

In the previous section it was suggested that a function definition could be found by inheritance, that is, by shallow binding. It is also possible to find a function definition by any of the other schemes for evaluating variables. When a function definition is found by pattern matching it is traditionally called pattern directed invocation.

Suppose we have procedures at different levels of generality, e.g. building a house, building a wooden house, building an A-frame house. A function call to build a particular type of house could then be matched to the most specific procedure which is appropriate. All that is required is to specify for each procedure a description against which each call for a function evaluation is to be matched.

When procedures are called by matching, the addition of new procedures can obviously change the flow of control. The calling procedure doesn't know precisely what procedure will be called.

In another form of pattern directed invocation, the calling procedure presents a result it wishes to be achieved. Procedures to be invoked have patterns which represent the result which they can produce if called. After matching, a procedure which produces (or may produce) the result wanted is called.

When pattern directed invocation is used, one generally finds that more than one procedure matches the call. There then comes the question of picking which one or ones to execute. Above it was suggested that the most specific be picked, but sometimes it is difficult to say which is most specific and in general there has been difficulty in finding general criteria for choice. This dilemma has led to the idea of choosing procedures. These procedures choose or arbitrate between competing alternatives, deciding either what procedures can execute or whose result will be accepted. On this view, procedures for deciding what to try or what to accept are just as important as procedures for doing it.

7.6 Factoring Out Procedural Details

Definitions of procedures can be represented in semantic nets by direct analogy with definitions in LISP. Series of steps can be linked by an THEN relation. Constructs like if-then-else can be represented by nodes with three roles, a test and two alternative steps.

In order to make procedures more abstract, and thus more general, there are several ways that procedural detail can be removed. One idea is to explicitly list prerequisite conditions which must be true in order for a procedure to be executed. These prerequisites are then assumed true during the execution of the procedure. This allows one to remove tests for these prerequisites that might be distributed through the procedure.

If a procedure is selected for execution, the first step is to check if its prerequisites are true. If not, the interpreter may try to find procedures which will make them true. If procedures are selected by matching, there is then a distinction between conditions which preclude the match and thus block consideration of a procedure at all and prerequisites, which are conditions which must be true, but that one is willing to make true in order to execute the procedure. The programmer has his choice as to what should be made conditions for match and what should be made prerequisites.

The use of prerequisites is perhaps more important when procedures are selected by match, since pattern directed invocation is often used to decouple the calling and called procedure. They are perhaps written by different people and then the needs of one match the abilities of the other. In such a situation it is less likely that the calling procedure will have made true all the conditions needed by the called procedure. Prerequisites are a method of insuring this.

When such abstract procedures are "compiled," one step would be to replace call by match with a direct call to what will match and then arrange that the calling procedure makes the prerequisites true. The check of prerequisites can then be removed from the called procedure.

The use of prerequisites also aids in the factoring of procedures in the sense that methods to make a prerequisite true might be distributed through the main procedure. Using prerequisites these methods can be written as separate procedures to be accessed when needed to make a prerequisite true or for whatever purpose.

In addition to using prerequisites, we can factor procedures by doing a case analysis on their inputs. In particular, we can distinguish

(104a) the normal input
(104b) inputs for which special case processing is required
(104c) inputs which, although they fall into the class of things handled by the procedure, require some processing which the procedure can't do. We might call these exceptions.
(104d) erroneous inputs.

Having made this classification, the definition of a procedure can be divided into

(105a) specification of the normal method
(105b) checks for special cases, exceptions, and errors.
(105c) procedures for what to do when these occur.

There is a question of how to store and access procedures for recovery from exceptions and errors. Suppose I am executing a "bake cake" procedure and as I pour the batter into the pan it overflows. An error check (I need to know about this possible error in pouring) reveals the overflow. What do I do now? Or suppose I am removing a birthday cake from the cake pan and it breaks in half. What do I do?

When something goes wrong in a problem solving program the traditional response is to back up, to go back to a higher level goal and seek an alternative way of solving the problem. Thus, for example, when the birthday cake breaks, I can back up until I find the goal of obtaining a cake for a birthday. An alternative to baking a cake is buying one.

As an alternative to backing up, a program could back off. That is, treat its entire problem solving strategy up to this point as data. The broken cake creates a problem to be solved in the context of the steps taken so far. The program could look for methods of repairing a cake (patch it with frosting) and then contrast the use of a patched cake with choice of an alternative like buying a cake. When something goes wrong, it is often possible to make a fix (patch it with frosting) but there is a question of whether this would be acceptable to a calling procedure. It is unrealistic to ask a calling procedure to pass down constraints for every possible eventuality. This is why it is important to take a more global view when responding to a local problem.

When a program backs off, it may recognize a problem as a very general error of procedure. For example, when a lecturer is finished with a talk using slides or an overhead projector, he wants to turn off the projector and turn on the room lights. If he turns off the projector it may be too dark to find his way to the room lights switch. This happens to many lecturers because they are often standing near the projector so they choose to do the turning off first. Backing off, one can analyze this as a situation where two goals are to be achieved, but where a prerequisite for achieving one goal is removed by achieving the other goal. It is an open question, however, as to what clues might be used for suspecting a particular kind of planning error.

7.7 Searching by Parallel Marker [49]

Consider the network in figure 42. Suppose we want to find out if there is any animal which is also a means of transport. One way to go about this would be to use a recursive algorithm to search for a connection from MEANS-OF-TRANSPORT and each of its descendants to ANIMAL. The search is complicated since the connection to ANIMAL may be indirect, through any of ANIMAL's descendants. Here we would find that HORSE is connected to ANIMAL via the node DOMESTIC. This method works slowly since it must individually examine each descendant of the MEANS-OF-TRANSPORT node, as well as many of ANIMAL's descendants.

Scott Fahlman argued in his Ph.D. thesis [24] that this problem was essentially one of intersecting the set consisting of the node ANIMAL and its descendants, with the set of MEANS-OF-TRANSPORT and its descendants, and that it could better be solved using a parallel approach. He envisioned his network (as yet unbuilt) as consisting of a large number of small parallel processing elements representing nodes and links. Node elements would be able to store several distinct marker bits, which would be propagated in parallel by the link units from node to node throughout the network.

In order to find the intersection of the two sets, a marker (call it M1) would be attached to the MEANS-OF-TRANSPORT node, then passed down and attached to each of its descendants in the network. Since link units propagate all markers down one level in the hierarchy in a single parallel step, the marking of MEANS-OF-TRANSPORT and its descendants would take time proportional to the longest path from MEANS-OF-TRANSPORT to the leaves of the tree. Next, a second marker, M2, marks the ANIMAL node and is propagated down to mark its descendants (again taking time proportional to the longest path). Finally, the system checks to see if there are any nodes marked by both M1 and M2; these nodes represent the intersection of the two sets. Here, HORSE would be the only node so marked.

Fig. 42. A Typical Semantic Network
<==<fig6-5.press<

NETL is the language Fahlman developed to represent knowledge on the parallel network machine. Fahlman defines several node and link types in NETL, which are represented in graphical form. There are two basic node types.

INDIVIDUAL nodes represent specific individual entities. For example, we could create a description for an individual, Clyde, who exists in the real world (also represented by an INDIVIDUAL node).

A TYPE node serves as a description whose structure and properties can be copied by INDIVIDUAL nodes. A TYPE node x can be seen as representing the "typical x". Fahlman associates each TYPE node with an INDIVIDUAL node which he claims represents the set of all things of that type; the TYPE node represents the typical member of that set. We could create a TYPE node, ELEPHANT, and an associated set node, ELEPHANT-SET (see Figure 43).

There are nine link types defined in NETL. The representation of all relationships between nodes is done using these primitives.

The most important link that Fahlman describes is the *VIRTUAL-COPY (*VC) link. When a *VC link is created, the system behaves exactly as though a portion of the network had been copied, without actually creating a physical copy. For example, if we create a *VC link between CLYDE and ELEPHANT, we cause CLYDE to inherit all the properties, memberships, and restrictions attached to the ELEPHANT node. It is as if we had actually copied the entire elephant description, replacing the ELEPHANT node with CLYDE, except that we have only created a single link. We say that the *VC link points upward from CLYDE to ELEPHANT. *VC links are transitive, so that if we create a *VC link between ELEPHANT and MAMMAL, CLYDE will inherit the properties attached to the MAMMAL node as well (see Figure 44).

In the usual tree structure, a node may have many descendants but only one immediate superior, with a single strand of ancestors. In NETL, a node can have any number of *VC links entering and leaving it. Because of this, it can have many immediate superiors in the hierarchy as well as many descendants. In addition to being a ELEPHANT, CLYDE might also be a CIRCUS-STAR and a REPUBLICAN; he can inherit from all superior descriptions (see Figure 45).

However, the parallel network requires no more time to traverse the hierarchy of ancestors of a node than it would to traverse a single chain of the same length (this is assuming that there are no directed cycles of *VC links which would cause the markers to loop infinitely). A node must have at least one *VC link tying it to some more "general node"; if we know nothing else about it, it can be attached to the most general node, THING.

Fig. 43. Node types
<==<fig6-6.press<

Fig. 44. CLYDE, an ELEPHANT, a subtype of MAMMAL
<= =<fig6-7.press<

Fig. 45. A node may have many immediate superiors
<= =<fig6-8.press<

*VC links can be crossed by markers in either direction (although inheritance is only established in one direction—from above to below, that is, the more specific description inherits from the more general ones.) Sending markers up *VC links from a node marks all descriptions of which the node is supposed to be a virtual copy, either directly or by inheritance. Marking upward from ELEPHANT would find the descriptions of PEANUT-EATER, MAMMAL, ANIMAL and other nodes above it in the hierarchy. Marking downward finds the types and subtypes of a node. Sending markers downward from ELEPHANT would mark the descriptions of CLYDE, AFRICAN-ELEPHANT, INDIAN-ELEPHANT, and their inferiors (see Figure 46).

Fig. 46. Illustration of Marker Propagation
<= =<fig6-9.press<

---

*EXIN ("exists in") links connect an individual to the area (time, location, subject-area) in which it is considered to exist. An area is also an individual. We could connect CLYDE to the node representing REAL-WORLD with an *EXIN link (see Figure 47).

Fig. 47. CLYDE exists in the real world
<= =<fig6-10.press<

A role is defined to be an INDIVIDUAL node which is connected by an *EXIN link to a TYPE node rather than an INDIVIDUAL node representing some area. The TYPE node is called the *owner* of the role. The INDIVIDUAL node NOSE is a role in the MAMMAL description. The node WEIGHT is a role in the PHYSOB ("physical object") description (see Figure 48).

*MAP links are used to create an individual version of some role within a virtual copy, in order to say something which does not apply to the original. A MAP node is used to represent the new version. A copy of the NOSE node could be created for the ELEPHANT node. The MAP node, which gets the name TRUNK, is connected to the NOSE node with a *MAP link to indicate the role of which it is a copy, and it is connected to the ELEPHANT node with an *EXIN link to indicate its owner (see Figure 49).

Fig. 48. Role nodes: the NOSE of a MAMMAL & the WEIGHT of a PHYSOB
<= =<fig6-11.press<

Fig. 49. The NOSE role mapped from MAMMAL to ELEPHANT
<= =<fig6-12.press<

The mapped description can be modified or added to without altering the role node being mapped. We would use a MAP node in representing the fact that Clyde's trunk was unusually short, linking the CLYDE'S-TRUNK MAP node to the TRUNK node with a *MAP link, and to the CLYDE node with an *EXIN link, and then attach the additional information to the CLYDE'S TRUNK node. When we look for information about Clyde's trunk, the original TRUNK description as well as CLYDE's copy are used, but the individual copy has precedence in case of a contradiction (see Figure 50).

*EQ (equal) links are used to equate two nodes. During marker propagation, any two nodes connected by an *EQ link behave as though they were a single node: anything known about one applies to the other as well. If Clyde is the son of Jumbo, then an equal link can be created linking JUMBO with the node representing the FATHER of CLYDE (see Figure 51). The link is also used to assign a value to a property.

Fig. 50. 'CLYDE'S TRUNK IS SHORT'
<= =<fig6-13.press<

Fig. 51. JUMBO is the FATHER of CLYDE
<= =<fig6-14.press<

We could create an *EQ link between the TRUNK-LENGTH node and the node representing 1.3 meters (see Figure 52).

In order to examine a mapped role within a description, the description must first be activated with a marker, and then the role can be examined using a different marker. Suppose we want to find the properties associated with Clyde's trunk. We first activate the CLYDE description by placing a marker M1 on the CLYDE node, and propagating it upward in the hierarchy. The marker is propagated up all *VC links and across all *EQ links in either direction. This marks all CLYDE's parents in the network (see Figure 53).

We now want to mark the role TRUNK. A second marker, M2, is placed on the TRUNK node, and propagated up all *VC links and across all *EQ links in either direction. It is also propagated up or down any *MAP link pointing to a MAP node whose owner node was activated by M1. If placed on a MAP node, it will be propagated upward to the node above, if placed on a role node, it will be propagated down the *MAP wires of all the nodes beneath it. This marking causes all the M2-marked nodes to function jointly as the

Fig. 52. A TRUNK is a CYLINDER WITH LENGTH 1.3 METERS
<==<fig6-15.press<

Fig. 53. MARKING TO FIND PROPERTIES ASSOCIATED WITH CLYDE'S TRUNK
<==<fig6-16.press<

description of Clyde's trunk (see Figure 53).

A *CANCEL link is used to cancel some role that would otherwise be inherited. If all elephants have hair, but Clyde is bald, we would represent this by extending a cancel link from CLYDE to HAIR (see Figure 54).

During marker propagation, marker-bits are used in pairs: the first one to activate a copy of a description, then the other to indicate cancellations within that copy. The cancellation marker is placed on any node pointed to by a *CANCEL link whose tail is a node activated by the first marker. The presence of a cancellation marker on a node inhibits the further passage of markers through the node. Any MAP nodes for which it is the owner will not be activated. The cancellation marker is propagated to any other node tied by a *EXIN link to the canceled node. Nodes marked with the cancellation marker are ignored when the system looks for nodes of its type (see Figure 55).

Fig. 54. ELEPHANTS HAVE HAIR, BUT CLYDE'S DOESN'T
<= =<fig6-17.press<

Fig. 55. Cancellation Markers
<= =<fig6-18.press<

In order to cancel some *identity* (indicated by a *VC or *EQ link) that would otherwise be inherited, a *CANVC link must be used. Since we first activate a description and then send out cancellation markers, canceling a *VC or *EQ link with a *CANCEL link would cause us to lose track of the activation markers once they had passed the canceled nodes, since any subsequent markers looking for the activation markers would be stopped at the canceled node. The *CANVC link causes a cancellation marker to be placed on an identity node *during* an activation scan. The corresponding activation marker will not mark or pass through any node so marked. A race occurs between the activation and cancellation markers on the way to the node to be canceled, but the cancellation markers always win, since they have a direct route to the node to be canceled, while the activation markers must always pass through at least one other node. Consider the network fragment in Figure 56).

The presence of the *CANVC link from CEPHALOPOD to SHELL-BEARER inhibits the passage of markers to the role node, SHELL, when the system searches for a CEPHALOPOD's SHELL. A race is created between the marker seeking to activate the SHELL-BEARER description and the marker trying to cancel it. The cancellation marker wins since it doesn't have to pass through any intermediate nodes on its way to SHELL-BEARER. When the activation marker arrives at the SHELL-BEARER node, it does not attach itself or pass through to continue marking that path. The reassertion of the *VC link from NAUTILUS to SHELL-BEARER allows the markers to reach that node when searching for a NAUTILUS's SHELL. Here, the activation marker wins since it has a direct route to SHELL-BEARER, while the cancellation marker must pass through CEPHALOPOD. When the cancellation marker arrives at SHELL-BEARER, it does not cancel the activation marker already there; it only prevents new activation markers from entering.

7.8 Selectional Restrictions

Fig. 56. *CANVC LINK IS USED TO CANCEL AN IDENTITY
<= =<fig6-19.press<

Selectional restrictions have been used to semantically constrain, for example, the subject of senses of *bark* so that *The block barked* is semantically unacceptable, while

(106a) The dog barked.
(106b) Fido barked.
(106c) The animal barked.
(106d) The father barked.
(106e) The seal barked.

all are acceptable, and *The Sargent barked an order* must have a sense of *bark* different from the first. The question is how these descriptions could be used in determining the acceptability of various candidate subjects of ANIMAL-BARK.

Almost anything can be the subject of a verb in some context, e.g.,

(107a) A funny thing happened at work this morning. A fellow came into my office carrying a dog and a shoebox. The shoebox barked. I wonder if the dog was a ventriloquist.

So the question we ask here is not whether a certain candidate concept could be used to fill a role, but only whether the candidate is at odds with what we know about the traditional fillers of that role.

The usual practice in existing computer programs is to rule candidates in through class inclusion, e.g., to ANIMAL-BARK you must be a dog or a seal. Ruling in by class inclusion is perhaps used in existing programs because they are typically restricted to small well defined domains where concepts are not used in very many ways and it is known exactly how the system will use any concept.

One can also take the view that knowledge of the general properties of a concept is built up through generalization from known exemplars. For example, a system is told that dogs and seals bark and then generalizes this to the general category of seals, dogs, wolves, coyotes, and foxes. Such a system would first attempt to rule in by comparison with a known exemplar. Failing this, it could attempt to rule out using characterizations and functions formed by generalization.

In any case, ruling in by comparison with exemplars seems an important step. Let us consider how this could be done. It is sufficient to show that there could be some individual which is characterized by both the candidate and the exemplar. For example, *the father barked* is acceptable because there can be an individual who is both a *father* and a *dog*. Further, it seems sufficient to show that either the exemplar or candidate is a specialization of the other. For example, even though a *fake gun* is not a *gun*, there seems little to be gained by disallowing *he shot the fake gun* because you can only shoot guns.

Suppose for efficiency reasons we restrict our attempts to establish one of these two sufficient conditions to running up the specialization tree. We run up from both the candidate and the exemplar and we note all concepts at which a path from the candidate intersects a path from the exemplar. Since all paths meet at the top of the tree, only three situations can obtain.

(108a) The intersection is the exemplar, e.g., *Fido barked* intersects at DOG. Accept.
(108b) The intersection is the candidate, e.g., *The animal barked* intersects at ANIMAL. Accept.
(108c) The intersection is at some other concept, e.g., *The block barked* intersects at, say, PHYSICAL-OBJECT. *The father barked* intersects at, say ANIMAL. In this case we accept unless the two paths come in through mutually exclusive SPECIES, e.g., cat and dog, or unless the two paths pass through mutually exclusive FUNCTION's, e.g., *hen crowed* would be ruled out because crowed requires a rooster which is male and a hen is female.

If <u>any</u> intersection is rejected then the candidate is rejected, since this means there is a way of showing entities which can be described by the candidate and exemplar to be mutually exclusive.

In the psychological literature one finds the quick comparison of a candidate to a prototype being done by comparison of sets of features of each. Tversky [104] suggests that the first step is to determine the salient dimensions of the comparison. For example, Cuba is similar both to Russia and to Puerto Rico but Russia is not similar to Puerto Rico because the dimensions of their comparison to Cuba are different. One abstracts the appropriate features from each concept. The candidate and exemplar are then compared by the formula

$$(109) \quad W = A\sum a_i + B\sum b_j + C\sum c_k$$

where a_i is the saliency of a feature had only by the candidate, b_j is the saliency of a feature had both by the candidate and the exemplar and c_k is the saliency of a feature had only by the exemplar. Saliency is some function of factors such as the strength of the feature in a concept, its uniqueness to the concept, its well formedness, and its familiarity. Such an approach could be implemented in the framework we provide, but it would appear to require too much computation for current hardware.

Comparison to prototype could be useful in forming new characterizations of concepts, but it alone does not solve the problem of selectional restrictions. For example, what features of *father* and *goat* would be needed so that *the father barked* is more acceptable than *the goat barked.* Goats are very similar to dogs but we know they do not bark. Fathers are a diverse set, not having much to do with dogs in particular but we know a dog can be a father. It appears best to resort to comparison by match of features or structure only when deduction based on explicit knowledge is not possible.

8. Interactions Between Grammar and Parsing

8.1 Typical Modularzation of Natural Language Systems

A complete utterance can be a sentence, e.g., *how are you,* a simple exclamation, e.g., *hello,* or a phrase uttered in response, e.g., *in the morning* uttered in response to *when are you going?* Constructing the response to a complete utterance may be viewed as a one, two, or three stage process as shown in Figure 57.

An example of a one stage processor was the Weizenbaum's ELIZA/DOCTOR system [107]. This program matched the antecedents of production rules to the input. When a match was found, the consequent of the matching rule would be executed. This would generate an output and sometimes it would also record a fact in memory which could be used later when no rule matched an input. This system had a very weak sense of connected discourse and no one has claimed that it exhibited "understanding" of the input.

Systems which, in some sense, understand the input have been constructed using a two stage process [96]. In this paradigm we let memory contain (a) rules for constructing well formed semantic structures, and (b) a structure which has been constructed on the basis of all the input so far. We will call the latter the discourse structure.

Understanding the input utterance amounts to extending the discourse structure in accordance with the structure formation rules and on the basis of examining the input. Such a system is much better at understanding connected discourse than the (a) form of system. There is a tendency to drive the understanding component of this type of system off of the discourse structure rather than the input. One can question whether it takes advantage of the structural properties of the input as well as it could. Advocates of this approach generally don't believe that the structure of the input is of much importance compared with the discourse structure.

Reason to question this approach comes from successful two stage systems of form (c), e.g., Woods [117], Hendrix [38]. These systems have only very weak coupling between inputs. They handle each complete utterance as a separate problem. Relying strongly on the structure of the input and semantic construction rules, they construct a semantic structure which is largely independent of the semantic structures of previous inputs. The response is then generated from this semantic structure.

Forms (b) and (c) can be combined as shown in (d). Here a three stage process is used. The input is converted into a semantic structure and then the semantic structure serves as input to a module which extends the discourse structure.

In order to evaluate these alternatives, we turn to a detailed discussion of the construction of semantic structures. Efficient and powerful programs are generally created by exploiting the natural constraints of the problem. The question we face is whether there are important generalizations based on the structure of the input and whether these are best exploited by using a separate level (semantic structure) of representation.

Fig. 57. Four Architectures
<==<fig8-1.press<

8.2 Grammatical Relations

Semantic structures can be built (and usually are) by the combination of smaller structures into larger ones according to a set of rules of combination. In Part 1, it was suggested that people organize data by instantiating pre-existing frames. If we take our semantic structures to be frames, then building semantic structures amounts to the combination of frames. How can frames be combined?

A principal method of combination is for one frame to fill a slot of another. For example, given the sentence *Sincerity frightens John*, a system could locate the frames SINCERITY, FRIGHTEN, and JOHN. The frame FRIGHTEN would have slots (or roles) for the thing doing the frightening and the thing frightened:

```
(110) [FRIGHTEN
         [SUBJECT: #PREDICATE-TYPE THING]
         [OBJECT:  #PREDICATE-TYPE ANIMAL]
```

Here FRIGHTEN is specified as having a SUBJECT slot which must be filled with something predicable as a THING and an OBJECT slot which must be filled with something predicable as an ANIMAL.

To construct a frame corresponding to *sincerity frightens John*, we create a copy of the FRIGHTEN frame and fill its slots with the JOHN and SINCERITY frames. We can tell which of these fills the OBJECT slot, because only JOHN is an ANIMAL. Therefore, we can take SINCERITY as the SUBJECT and JOHN as the OBJECT.

```
(111) [FRIGHTEN-1
         [SUBJECT: #EQUAL FRIGHTEN]
         [OBJECT:  #EQUAL JOHN]
```

Obviously, there are many sentences, e.g., *The ball hit the boy* and *The boy hit the ball* where predicability will not be an adequate method of resolving which frames fill which slots. However, we notice that in all of these sentences, the grammatical subject comes before the verb and the grammatical object comes after the verb. We claim that it is possible to define a set of grammatical relations, here SUBJECT and OBJECT, such that, for any sentence, we can specify where in the sentence the phrases, if any, specifying each grammatical relation occur. For example, in *Did the boy hit the ball?*, the phrase, *the boy*, which describes the SUBJECT, comes after the auxiliary verb, *hit*.

A fairly comprehensive specification of grammatical relations is quite complex and will be delayed to a later chapter. Here we only wish to use the idea of grammatical relations in discussing the relation between grammar and parsing.

8.3 Syntactic and Lexical Transformations

The two sentences

(112a) Sincerity frightens John. *active*
(112b) *Sincerity is frightened by John. *passive*

both have *sincerity* before the verb and *John* after the verb, but while in (112a) *sincerity* describes the SUBJECT, in (112b) it describes the OBJECT. Sentence (112b) is much closer in meaning to

(113) John frightens sincerity.

than to (112a). This chapter will attempt to refine two statements made in the last chapter: (a) the SUBJECT slot occurs before the verb, and (b) the SUBJECT slot of *frighten* is the frightener. There are (at least) two methods of doing this, which we will refer to as the <u>syntactic</u> and <u>lexical</u> approaches.[9]

The syntactic approach claims that (112b) is derived from (113) by a transformation of syntactic structures. The sentence (113), for example, is held to have an implicit structure, called its syntactic structure, of the general form:

(114) $[_S [_{NP}$ John$] [_{VP}$ frightens $[_{NP}$ sincerity$]]]$ *syntactic structure*

This transformation has the form of a production rule[10] whose left side matches the structure in (114). The right side of this transformation constructs a structure of the form:

(115) $[_S [_{NP}$ John$] [_{VP}$ is frightened$] [_{PP}$ by John$]]$

by picking up constituents of the structure in (114) and inserting them into the pattern:

(116) $[_S [_{NP}$ X$] [_{VP}$ is Y-ed$] [_{PP}$ by Z$]]$

The exact form of the structures and transformations has been debated and refined for the past twenty years. Grammatical Relations (e.g., subject, object) can be assigned to (112b) by running the transformation in reverse to get (113) and then using the rules that the SUBJECT comes before the verb and the OBJECT comes after the verb.[11]

9. The debate between these two approaches first became important at MIT when Noam Chomsky wrote "Remarks" [16] in 1970. Since that time, Chomsky has generally been associated with the syntactic camp. The prime proponent of the lexical approach (at least at MIT) has been Joan Bresnan (since 1977 [10]). Chomsky and Bresnan discuss this *John frightens sincerity* example in [] and [10, p. 9], respectively.

10. A *production rule* is a rule of the form: if <pattern> then <action>.

11. It would be more accurate to say that the subject of a sentence is defined to be the noun phrase immediately dominated by a sentence node because there are languages (e.g., Turkish) where the subject can follow the object. Similarly, it may be more accurate to define the object in terms of domination relationships rather than linear precedents.

The lexical approach explains the same linguistic phenomena by operating on the representation of the lexical entries. For example, the lexical approach would create a passive entry in the lexicon for *be frighten*, which would be very similar to the active entry for *frighten*, except for the mapping relationships between constituent structure and semantics. The active entry would map the syntactic subject (e.g., the first noun phrase) into the semantic subject (e.g., the agent) whereas the passive entry would map the syntactic subject into the semantic object (e.g., the recipient).

Many different types of semantic structures have been proposed. Here we use a form of frames compatible with the discussion in Part 1. Let us see how this works for sentence (112b). First, the structure in (115) is created. Next, frames corresponding to the words are located. The word *frightened* is decomposed, as shown, into the verb, *frighten*, and the suffix *-ed*. In finding frames *frighten* is mapped into FRIGHTEN, as before. The suffix *-ed* is mapped into the frame, SECOND-PARTICIPLE. This frame has the form

```
(117) [SECOND-PARTICIPLE
        [SUBJECT: #EQUAL (OBJECT *ROLE SUBJECT-COMPLEMENT:)]
        [SUBJECT-COMPLEMENT: #PREDICATE-TYPE PROCESS]]
```

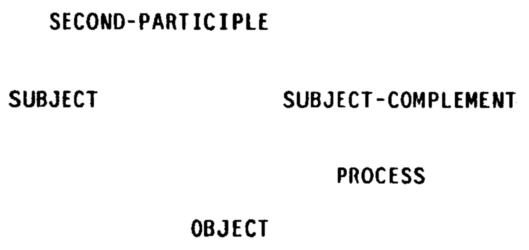
This says that the subject complement of a SECOND-PARTICIPLE must be a PROCESS and the SUBJECT of a SECOND-PARTICIPLE must be equal to the OBJECT of the SUBJECT-COMPLEMENT. The structure created is illustrated in Figure 58. What happens is that the filler of a slot of a slot filler of SECOND-PARTICIPLE is raised to also fill another slot of SECOND-PARTICIPLE.

Copying SECOND-PARTICIPLE and filling its SUBJECT-COMPLEMENT slot with a copy of FRIGHTEN, we obtain

```
(118) [SECOND-PARTICIPLE-1
        [SUBJECT: #EQUAL (OBJECT *ROLE FRIGHTEN-1)]
        [SUBJECT-COMPLEMENT: #EQUAL
          [FRIGHTEN-1 [SUBJECT: #PREDICATE-TYPE THING]
                      [OBJECT:  #PREDICATE-TYPE ANIMAL]]]]
```

As was explained in Part I, we don't actually have to copy the slots of FRIGHTEN-1 from FRIGHTEN until we fill them if we agree that FRIGHTEN-1 inherits from FRIGHTEN. Next, *is* is used to find the frame for *is* (not *is-a*) predication BE, which is defined as

Fig. 58. A Representation of Passive Sentences



```
(119)  [BE [SUBJECT: #EQUAL (SUBJECT *ROLE SUBJECT-COMPLEMENT)]
             [SUBJECT-COMPLEMENT: #PREDICATE-TYPE COMPLEMENT]]
```

BE is a frame which equates the SUBJECT of the frame in its SUBJECT-COMPLEMENT slot with its own SUBJECT.

Copying BE and filling the copy's SUBJECT-COMPLEMENT slot with second-PARTICIPLE-2, we obtain

```
(120)  [BE-1 [SUBJECT: #EQUAL (SUBJECT #ROLE SECOND-PARTICIPLE-1)]
             [SUBJECT-COMPLEMENT: #EQUAL
               [SECOND-PARTICIPLE-1
                 [SUBJECT: #EQUAL (OBJECT *ROLE FRIGHTEN-1)]
                 [SUBJECT-COMPLEMENT: #EQUAL
                   [FRIGHTEN-1 [SUBJECT: #PREDICATE-TYPE THING]
                               [OBJECT: #PREDICATE-TYPE ANIMAL]]]]]
```

This frame BE-1, is the frame corresponding to our verb, *be frightened.* It finds its SUBJECT to the left of the verb, just as *frightens* in (112a) does. In constructing a frame for *be frightened* by raising slots, we have avoided the need to inverse transform (112b) to (112a) in order to use one set of rules for locating the phrases which describe specific grammatical relations in the input. There we have the fundamental difference between the lexical and transformational approaches. The two approaches will lead to different grammars, and different parsing techniques.

8.4 The Autonomy of Syntax

It is sometimes argued that it is a mistake to construct separate syntactic and semantic structures. Indeed, this is a non-trivial argument to resolve. It requires assessment not only of subtle phenomena, but also of the whole systems which can be set up each way. The arguments in favor of separate systems are generally considered stronger. There are three types of arguments made.

First, it is often possible to define things so that there are several semantic structures corresponding to the same syntactic structure, depending on what sense of a word is chosen. For example, the sentences

(121a) Mary made John a cake.	*direct object*
(121b) Mary made John a poor man.	*object control complement*
(121c) Mary made John a good wife.	*subject control complement*

can all be said to have the same syntactic structure:

(122a) $[_{S}\,[_{NP1}\text{ Mary}]\,[_{VP}\text{ made }[_{NP2}\text{ John}]\,[_{NP3}\ldots]]]$

but they have different semantic structures; NP3 is a direct object in (119), an object controlled complement in (120) and a subject controlled complement in (121). When there are several semantic structures corresponding to one syntactic structure, it is efficient to search first for the syntactic structure and then further refine that to the proper semantic structure, rather than immediately having to search for one of a much larger number of semantic structures.

The second type of argument can be illustrated by the following sentences:

(123a) Some <u>which</u> <u>I</u> <u>saw</u> were barking. *syntactically and semantically parallel*
(123b) The dogs <u>which</u> <u>I</u> <u>saw</u> were barking.

(124a) Some <u>of</u> <u>the</u> <u>dogs</u> were barking. *semantically distinct*
(124b) *The tails <u>of</u> <u>the</u> <u>dogs</u> were barking.

All four of these sentences are syntactically parallel, but only (123a-b) are semantically parallel. When the full relative clause *which I saw* in (123) is replaced by the prepositional phrase *of the dogs* in (124), the parallelism breaks down. The *of* in (124a) is used to make a <u>partitive</u> construction so that *some of the dogs* has the semantic properties of dogs. In contrast, the *of* in (124b) is used as a <u>restrictive</u> modifier; *the tails of the dogs* has the semantic properties of tails (and can't bark). If we suppress this difference between (123) and (124) into the semantic structures, then all four sentences have very similar syntactic structures and can be searched for in a uniform way. Otherwise, (123) becomes an exception during the first stage of searching.

In both of these arguments we see the desire to reduce the number of distinct structures which we might initially impose on an input sentence. This is because the more structures you have, the more difficult (either in memory space, search time, or complexity of algorithms) it is to find the appropriate one, unless the appropriate structure is indicated by some local property of the input. Unfortunately, in English, this is usually not the case.

8.5 Grammars

Both the lexical and the transformational approach start out by finding a syntactic structure. A <u>grammar</u> gives us a method of specifying what syntactic structures may be constructed for a given language. The procedure in modern structural linguistics (which we follow here) is to give a formal definition of language and grammar. The resulting formal system is not claimed to capture every subtlety of language, but it will allow the construction of an engineering application in a principled way. The formal definitions used are usually sufficiently general to allow the expression of any insights which a system builder may have.

A formal language we define to be a (possibly infinite) subset of the finite (but arbitrarily long) strings which can be formed from a given finite set of symbols, which following common practice will be called terminal symbols. A generative grammar for a formal language is a set of rules which will generate all the strings in the language and no strings which are not in the language. (Given a generative grammar we have, in principle, a way of determining if a given string is in the language. We generate all the strings and see if it is included.)

A context free grammar consists of a set of terminal symbols, a set of non-terminal symbols, a set of production rules taking non-terminal symbols into strings of terminal and non-terminal symbols, and one non-terminal symbol distinguished as the starting symbol. A complete context free grammar for a noun phrase is shown in Figure 59.

Using this grammar, the noun phrase *a big dog house* can be derived as

(125) $[_{NP}\,[_{DET}\ \text{a}]$
$\quad [_{NP\text{-}ADJ}\,[_{ADJ}\ \text{big}]$
$\qquad [_{NP\text{-}ADJ}\,[_{NP\text{-}N}\,[_{N}\ \text{dog}]$
$\qquad\qquad [_{NP\text{-}N}\,[_{N}\ \text{house}]]]]]]$

The string *a big dog house* is said to be <u>unambiguous</u> with respect to the grammar in Figure 59 because there is only one way to derive it using that grammar. If, recognizing that *brown* can be either an adjective (e.g., *the* $\textit{brown}_A$ *book*) or a noun (e.g., $\textit{brown}_N$ *is my favorite color*), we added the production, N → *brown*, to our grammar, then *the big brown house* would have two derivations, and thus would be ambiguous.

Whenever a context free production rule has the same non-terminal on both the left and right sides, it is said to be recursive. Recursive rules arise fairly often in a context free grammar of English because there are quite a few constructions, such as modification by an adjective, which can be repeated an unlimited number of times. The only way to model repetition in a context free grammar is by recursion. It is customary to distinguish three types of recursive rules:

(126a) A → aAb	*center embedding*
(126b) A → aA	*right recursive*
(126c) A → Ab	*left recursive*

Fig. 59. A Context-Free Grammar

<u>Productions</u>	<u>Terminal Symbols</u>
NP → NP-ADJ	the a
NP → DET NP-ADJ	big brown hat
NP-N → N NP-N	dog house stand
NP-N → N	
NP-ADJ → ADJ NP-ADJ	<u>Non-Terminal Symbols</u>
NP-ADJ → NP-N	NP-ADJ NP-N
DET → the	DET ADJ N
DET → a	
ADJ → big	<u>Distinguished Non-Terminal</u>
ADJ → hot	NP
ADJ → brown	
N → dog	
N → house	
N → stand	

It turns out that the center embedding case is more difficult to deal with.[12] If it is applied n times, we get n a's followed at some later point by n b's. To parse such a string, it is necessary to save up (or at least count) the a's and match them off against the b's. No such matching is required in the left and right recursive rules. Our noun phrase grammar fortunately has only right recursive rules.

There is a disadvantage to the expression of repetition with recursive context free rules. They define a more complex syntactic structure than we need, and thus go against the arguments presented for the autonomy of syntax. For example, *a big dog house* would be adequately represented by the syntactic structure:

(127) $[_{NP}\, [_{DET}$ a$]\, [_{ADJ}$ big$]\, [_{N}$ dog$]\, [_{N}$ house$]]$

In fact, our grammar would assign *a hot dog stand* the syntactic structure:

(128) $[_{NP}\, [_{DET}$ a$]$
$[_{NP\text{-}ADJ}\, [_{ADJ}$ hot$]$
$[_{NP\text{-}ADJ}\, [_{NP\text{-}N}\, [_{N}$ dog$]$
$[_{NP\text{-}N}\, [_{N}$ stand$]]]]]]$

which suggests that what we are dealing with is a hot dog-stand, not a hot-dog stand. More production rules could be added to the grammar in order to generate a syntactic structure which first combines *hot* with *dog* and then the result with *stand*, but this would make *a hot dog stand* syntactically ambiguous. Such an ambiguity is better put off to semantics. This is what the neutral form (127) does for us.

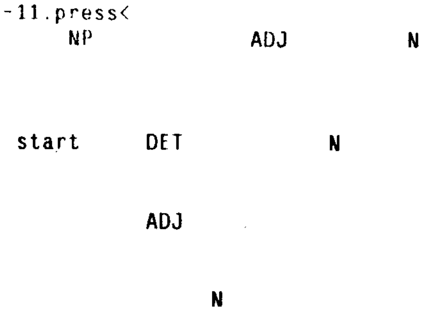
We can generate syntactic structures like (127) if we replace our context free grammar with a grammar whose productions have right sides which are <u>regular expressions</u>. Regular expressions are formed using the operations concatenation, either/or (|) and zero or more repetitions (*). Using these operations the productions for the noun phrase can be rewritten as:

(129a) NP → DET ADJ* N N* | ADJ* N N*
(129b) DET → the | a
(129c) ADJ → big | brown | hot
(129d) N → dog | house | stand

Since a regular expression can also be expressed as a state graph for a finite state machine, (129a) could alternatively be represented as:

12. Formally, left and right recursive grammars can be reformulated as finite state (regular) grammars, so that they can be processed with only finite memory (i.e., a finite state machine). In contrast, center embedding grammars cannot be reformulated; center embedding grammars cannot be processed with a finite state machine. That is, center embedding grammars are strictly context-free; center embedding grammars require the full power of a push down automata (PDA) with an unlimited amount of memory for the stack [15, 17, 54].

(130) <==<fig8-11.press<

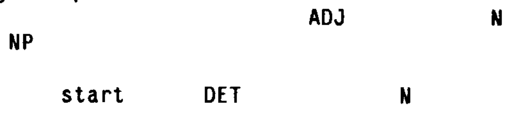


where a node with a circle in it represents an accepting state.

As will be discussed in the next chapter, the best known methods for parsing any context free grammar take time proportional to the cube (a very complex method is slightly below this) of the length of the string. By contrast, a string can be accepted by a finite state machine in time proportional to the length. By eliminating undesirable recursion we also increase our chances of getting an efficient parser. It makes more explicit where power is really needed.

There is a notational simplification which can be made to the finite state machine above which unfortunately does not lead to any increase in parsing speed. By defining a "free ride" arc, indicated by having no symbol on it, (130) can be rewritten as:

(131) <==<fig8-12.press<



Conventions are that a free ride arc can be followed without taking up a constituent from the input.

8.6 Wait and See

The non-terminal symbols in a grammar are sometimes said to define syntactic categories. Words or phrases which can be generated from a non-terminal symbol are said to be members of that syntactic category. The syntactic categories of words are often called parts of speech. Frequently, a word is a member of more than one syntactic category, e.g., cut can be either a noun or a verb. Sometimes a sentence may have one or more syntactic structures for one choice of syntactic category for a given word and none for the other possible syntactic categories for that word.

For example, *schedule* can be either a noun or a verb, but in (132a) it can be only a noun. In (132b) only the syntactic structure corresponding to the use of schedule as a verb has a semantic interpretation.

(132a) I lost my schedule$_N$ in the lecture. *noun*
(132b) I plan to schedule$_V$ the lecture you want. *verb*

The wait and see[13] principle says that one should always defer decisions whenever the expected cost of a task, given that the decision is made later is less than the expected cost of the task, given that the decision is made immediately. Thus, it would be desirable to defer the decision on the syntactic category of a word until enough was known to rule out categories for which no syntactic (or a syntactic but no semantic) structure was possible, unless deferring the decision used more computational resources than were saved. In (132a) we know that *schedule* must be a noun since a verb can't be preceded by *my*. This can be checked very cheaply. In (132b) we can determine that *schedule* is a verb only by considering the semantic structures. This can be seen by comparing:

(133a) I plan to schedule the lecture you want.
PRO V INF-to V DET N PRO V
(133b) I sent to committee the proposal you made.
PRO V P N DET N PRO V

These two sentences have the same sequence of parts of speech except that the sequence "INF-to V" in (133a) is replaced in (133b) by "P N". Since the INF-to part of speech of *to* is used if and only if it precedes a verb, there is no way to syntactically rule out the choice of *schedule* as a noun in (133a) without ruling out the syntactic structure of (133b). Thus, in the case of (133a) there is no possibility of waiting for syntactic information which will determine the part of speech of *schedule*. Whether it is cost effective to use semantic information in such a case will be discussed later.

This case can be contrasted with:

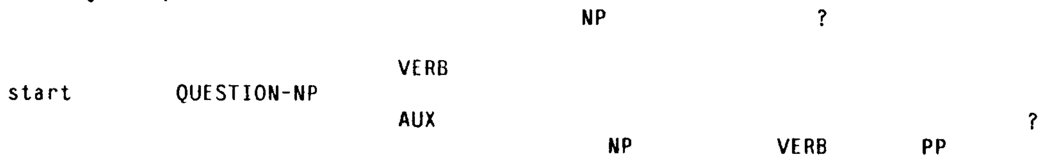
(134a) What is an AI?
NP1 V NP2
(134b) What is an AI doing in here?
NP1 AUX NP2 V PP
(134c) An AI is doing what in here?
NP2 AUX V NP1 PP

In (132), *what* is the subject of *is* and *an ai* is the subject complement of *is*. By contrast, in (134), *what* is the object of *doing* and *an ai* is the subject of *doing*. Sentence (133) is similar in meaning to (134), from which it is often transformationally derived by moving the *what* to the left end and then interchanging the subject and the auxiliary verb, *is*. (This transformation is known as wh-movement).

On the standard analysis, *is* has a different part of speech in (132) and (133). Suppose the right side of the production rule for a sentence is represented by the simplified state transition network in Figure 60. Suppose also that a parser which traverses a sentence left to right is given a sentence which starts out *what is*. After it has seen just this much of the sentence, for all it knows, the sentence could be either (132) or (133). Yet, in order to traverse the state transition network in Figure 60, the parser has to decide whether the part of speech of *is* is V or AUX.

13. "Wait and see" is known by many other names: e.g., delayed binding, lazy binding, principle of least commitment.

Fig. 60. Transition Network for "Is" Examples
<==<fig8-13.press<



One choice will be right and one will be wrong. As opposed to (133a), this decision can be resolved syntactically. A verb comes after the next NP in the AUX case and not in the VERB case. Since an NP is expected after the *is* in either case, there is no real need to make the choice of the part of speech of *is* until we reach the place in the sentence (after the NP, where we do or don't reach a verb) where we have the information to do it. Perhaps the state transition network in Figure 61 would have been a better design than that in Figure 60.

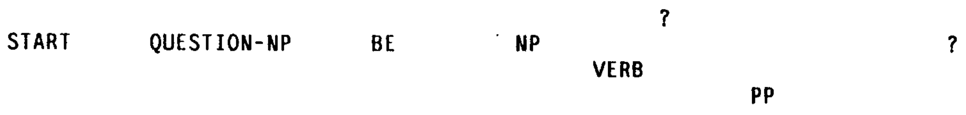
Unfortunately, this design has a disadvantage not present in Figure 61. Both of the syntactic sequences

(135a) WH-NP V NP ?
(135b) WH-NP AUX NP V PP ?

are instantiated by sentences having other verbs and other auxiliaries besides *be*. Therefore, the network in Figure 61 must be used in addition to, not in place of, the network in Figure 60.

The problem sketched here with *be* is a general one. One instance of a general pattern is hard to distinguish from some instance of a second general pattern, while in other cases the two patterns are more easily distinguished. Therefore, the wait and see strategy requires us to create a special case for the hard to distinguish instances. The application of wait and see leads to an expansion in the number of states in the network to accommodate special cases. If the network is created by hand, this extra complexity makes it more difficult to construct a relatively complete grammar. Fortunately, it should be possible to automatically construct such an expanded network, given a description of the general rules and their problem instances.

Fig. 61. Revised Transition Network
<==<fig8-14.press<



8.7 Features

· It is possible to associate with each phrase, or partially constructed phrase, a set of features. A feature is a variable having a small set of possible values. Binary features have two possible values and are particularly useful for digital computer implementations. The following discussion shows how the need for features arises.

The state transition network in Figure 59 will accept all of the following strings as noun phrases.

(136a) These dogs
(136b) *These dog
(136c) *This dogs
(136d) This dog

There is a rule of English which says that if the determiner and the verb are both marked as singular or plural, then they must agree. This has not been captured by (131), which allows (136b-c) even though they don't agree.

Agreement of determiner and noun could be enforced by expanding (131) as shown in Figure 62. It can be seen that the effect of remembering the number of the determiner in the state is basically to triple the number of states, corresponding to the three possibilities to be remembered.

Fig. 62. Agreement
<= =<fig8-15.press<

The consequences of this tripling of states depend on what parsing method is to be used. If the method only looks at states which have already been reached (say by keeping a list of reached states), then on a given parse. a left to right parser will always choose and subsequently look at just one of the three branches from determiner. On the other hand, if a parser looks at all states to see if they have been reached, then the more states, the more work the parser will have to do.

People can understand phrases like *this dogs* and *these dog* (an even more forgivable error is *truckers fees* for *truckers' fees*). Agreement errors are often not "fatal" errors. If one wants to design a parser to mimic this behavior, the use of separate states for singular and plural is not attractive.

In previous sections, we have dealt with a proliferation of states by suppressing constraints into the semantic component of the grammar. But in doing this, we may raise the computational cost of applying the constraints needed to limit the parser's search for syntactic structures. These constraints will no longer be implicit in the state reached, but will instead limit the semantic structures which can be constructed. To apply them, it will be necessary to attempt the construction of semantic structures. This can be a fairly costly operation.

While construction of semantic structures is usually fairly costly, it need not be expensive to apply constraints like determiner and noun agreement if one uses features. For example, we could define the binary features SINGULAR and PLURAL which are associated with determiners and nouns as shown in Figure 63. Note that since some determiners and nouns are not marked for number, a determiner or noun can be either singular, plural, or unmarked. To model these three values with binary features takes two features, e.g. SINGULAR and PLURAL. The use of must have and must not have values allows a fast implementation of the constraint that if a phrase has a SINGULAR value of 1, it must not have a PLURAL value of 1. Whenever a constituent is added to a phrase a check is first made to see if the phrase has a feature which the constituent says it must not have. Representing features values as bit vectors this check can be done by anding the two vectors and checking if the result is zero (which can be done in a single instruction on some computers). If the result is zero, all is well and the must have vector is anded into the vector of the phrase.

These features checks can be performed very quickly and with the use of relatively little storage.

Fig. 63. Features

	must have		must not have	
	Singular	Plural	Singular	Plural
the	0	0	0	0
this	1	0	0	1
these	0	1	1	0
dog	1	0	0	1
dogs	0	1	1	0
sheep	0	0	0	0

When features are not used, state (our memory for what has been found so far) is represented by position in the right side of a production rule. When features are used, state is factored into a position and a feature vector. This factoring makes sense when constraints may be factored into constraints on constituent order, plus constraints which (a) block some but not all sequences satisfying this constituent order and (b) may be easily determined from the input and represented as feature sets.

8.8 Level of Attachment

The following sentences and phrases are structurally ambiguous:[14]

(137a) Put the block in the box on the table.
(137b) He carried nothing to indicate that he was one of the group.
(137c) We sighted the man with the binoculars.
(137d) We never fought a bull with real courage.
(137e) He hit the man with the stick.
(137f) He seemed nice to her.
(137g) The stout major's wife
(137h) Jane's children's book
(137i) hot dog stand

For example, in (137c) *with the binoculars* can refer either to *the man* as in (138a) or to *sighted* as in (138b).

(138a) I saw [the man with the binoculars].
(138b) [I saw the man] with the binoculars.

The attachment problem is often thought of as a question of the level of attachment. Should we attach to the lowest possible phrase (e.g., [$_{\mathrm{NP}}$ *the man*]), or somewhere higher up in the tree (e.g., [$_{\mathrm{VP}}$ *saw the man*], [$_{\mathrm{S}}$ *I saw the man*])? In many cases, there is a strong bias toward a preferred interpretation. A number of psycholinguists (e.g., Kimball [46, 29, 28]) have attempted to characterize these biases in terms of syntactic structure. These authors have suggested that people adopt heuristics such as minimal attachment (adopt the structure with the fewest number of nodes) and right association (prefer [x [x x]] over [[x x] x]).[15]

There is a similar attachment problem in sequences of nouns (e.g., *hot dog stand*), which usually associate to the left

14. The second through sixth of these examples appeared in [105].
15. It may not be appropriate to introduce a left/right bias in the heuristic attachment principles. The tendency for right association in English (e.g., *yesterday* tends to attach to the right most clause in [*I said that* [*you said that* [*he said that ... yesterday*]]]) could be explained in terms of domination relations (lowest attachement) instead of precedence relations (right association). In right branching constructions (e.g., clauses in English), lowest attachment and right association make the same predictions. However, in left branching constructions (e.g., clauses in Turish or genitive phrases in English such as *The stout major's wife* and *Jane's children's book*), the two principles make opposite predictions. Such expressions usually associate to the left (lowest attachment).

(139a) [[[[hot] dog] stand] salesman] *left recursive*

but they can also associate from the right

(139b) [Boston [city [school [committee]]]] *right recursive*

It is claimed that while very long left associations are possible, e.g., *fire plug cap replacement plan*, it is rare to see a right association with more than three elements.

It should be clear from the examples that the choice of attachment frequently cannot be made on syntactic grounds alone. We would therefore like to write our grammar so as to eliminate the attachment issue from syntactic parsing. The question is whether the postponement of this issue leaves us with an inability to construct the rest of the syntactic structure. In the case of left attachment to a noun phrase there is no problem. We simply analyze the noun phrase with a rule like that shown in Figure 64.[16]

However, in the case of right attachment to a clause, it is not so clear how to forecast what can come next, independently of how the attachments are done. One issue is the satisfaction of certain verbs which have required arguments. Thus, (140a) is unambigous while (140b) is not

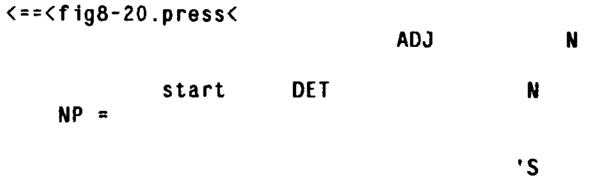
(140a) I put the block in the box. *unambiguous*
(140b) I saw the block in the box. *ambiguous*

because *put* requires a second argument, whereas *saw* does not.

It is not clear whether or not this argument requirement of *put* should be suppressed to semantics. In favor of suppression is the fact that *put* has a sense, *put the shot* which doesn't take a location. Requirements on arguments seem to apply generally to word senses, not to words. Thus to apply these constraints, we have to pick senses of words, which gets us into semantics. On the other hand, once a right attachment is deferred, only a limited amount of additional structure can be assembled with certainty.

One approach to this problem will be taken up in the chapter on parsing. It involves treating all the syntactic parses of a sentence as a single structure, with the unambiguous parts of the sentence represented only once.

Fig. 64.



16. A more general solution is required in practice to handle *King of Denmark's son*, which shows that the left argument of *'s* is an NP (e.g., $[_{NP}$ King of Denmark]), not an N (e.g., $[_{N}$ Denmark]).

8.9 WH-Movement

Section 8.3 showed how the relationship between a passive sentence and its corresponding active sentence could be captured in the semantic structure. A specific lexical item, the second (or past) participle, was stipulated to have joint constraints on its slots. These constraints stipulated that two different slots would be filled by the same lexical item. There are other types of sentence pairs whose relationship seems less amenable to explanation by this mechanism. Consider the following sentences.

(141a) Bob has told Harry to pat which dog?
(141b) Which dog has Bob told Harry to pat?

The difference between these two sentences is traditionally explained as follows. Starting with (141a), (141b) is produced by moving *which dog* to the left end and interchanging *Bob* and *has*. The interchanging of *Bob* and *has* is a local phenomena (called AUX-inversion) which presents relatively little trouble to a parser.[17] However, the movement of the WH-phrase[18] creates serious parsing problems. Unlike the passive, the WH-phrase can move up and to the left from arbitrarily deep in the syntax tree and there are no special lexical items (e.g., SECOND-PARTICIPLE) with which it is associated. Both of these problems make it difficult to treat WH-like passive.

The importance of finding the hole from which a WH-element was moved is illustrated by the following sentences.

(142a) *You must give the ball to.
(142b) Who must you give the ball to (who).

The final *to* in (142a) makes this sentence ungrammatical. The final *to* in (142b) is grammatical only because it is understood that *who* has been reinserted. Also, a sentence is bad if there is no place to put the WH-element, as in:

(143a) What did Bob play a game with you.
(143b) What did Bob play (what) with you.
(143c) What did Bob play a game with (what).

The syntactic structures generated by WH-movement can be generated by context free rules such as:

(144a) S $\rightarrow$ WH-NP AUX NP VP-with-NP-hole
(144b) VP-with-NP-hole $\rightarrow$ V NP-hole PP

17. Actually, AUX-inversion can be problematic when it is difficult for the parser to determine the boundaries of the subject. Consider, for example, the fragment Is the boy from France ... which should probably be transformed into *The boy is*.... However, if the fragment is followed by a complement as in *Is the boy from France happy?* then it should be transformed into *The boy from France is*.... One can imagine pathological cases where there might be a combinatoric number of ways to apply the transformation: *Is the block in the box on the table in the kitchen ...?*. The combinatorics of this construction are discussed in [18].
18. Wh-phrases are so-called because the question words which, what, who, when, where, why, etc. all start with *wh*

(144c) VP-with-NP-hole → V NP PP-with-NP-hole
(144d) PP → P NP
(144e) PP-with-NP-hole → P NP-hole

The technique used is to propagate the hole through the structure by writing new context free rules which explicitly mention the hole. The success of this method depends on the fact that, in English, you can only extract one WH-element out of a phrase.[19] That is, we (generally)[20] don't have phrases with two WH-holes:

(145) *Who$_i$ did Betsy ask what$_j$ [$_{\text{VP-with-2-holes}}$ Tom hit Δ_j to Δ_j]?

To complete the grammar in (144), it would be necessary to add context free rules for every syntactic category of phrase that a hole can pass into and for every syntactic category of hole that can pass into that category of phrase. So expanded out, this approach causes a massive reduplication of the rules, but when building a parser, it is not necessary to actually expand out the rules in this fashion. Just as we can add features to each phrase, we can add a slot which holds the hole, if any, being passed down. We fill in the appropriate syntactic category of hole as needed.

Linguists have observed that many syntactic structures have substructures from which a WH-element cannot be removed. For example, one can't say

(146a) *What$_i$ did John report the fact that Sally liked Δ_i?[21]
(146b) *What$_i$ dog did you take Δ_i's bone?

though there is no common agreement on the specificiation of these constraints. The most convenient way to implement such constraints is to build them directly into the parser. They come into play when the parser is filling WH-slots or building phrases out of constituents. One interesting research problem is to try to show that they follow from some psychological model of the parsing process.

8.10 Conjunction

Fundamental to conjunction is the notion of two constructions being "parallel". It is not exactly clear how "parallel" should be defined. Generally, both conjuncts must be constituents and must share the same part of speech:

(147a) The scene [$_{PP}$ of the movie] and [$_{PP}$ of the play] was in Chicago.

19. The facts seem to be different in Swedish; see Engdahl's thesis []
20. There are some interesting sentences with two holes, e.g., *Which violins are these sonatas easy to play* Δ *on* Δ*?* These sentences seem to be problematic for a context-free framework, though there is a possible straightforward (and unattractive) solution where each part of speech is hyphenated with -with-a-wh-hole and with -with-a-tough-movement-hole.
21. This fact was discussed extensively by Ross in his thesis [89]. Phrases like *the fact that...* of the form [$_{NP}$ NP S] are called complex noun phrases, and are considered to be wh-islands (Ross' term), because they generally block wh-movement. (Phrases with one hole already extracted are also called wh-islands, because it is impossible to extract a second hole as mentioned above.)

This is by far the most common kind of conjunctive construction, but it is not the only kind. First, it is possible, in certain cases, to violate the part of speech condition, as long as the conjuncts are semantically parallel. For example, consider (148) where *now* is a noun phrase and *in the future* is a prepositional phrase.

(148) We expect difficulties $\underline{\text{now}}_{NP}$ and $\underline{\text{in}}$ $\underline{\text{the}}$ $\underline{\text{future}}_{PP}$.

Secondly, (149) suggests that it is possible to violate the constituency condition

(149) [John] [wrote] and [Mary] [read] a detective story.

unless one accepts a somewhat unusual analysis of the syntactic structures of the form[22]

(150) $[_{\text{S-with-NP-hole}}$ John wrote $\Delta_i]$ and $[_{\text{S-with-NP-hole}}$ Mary read $\Delta_i]$ a detective story_i.

These structures assume that the noun phrase *a detective story* has been shifted to the right leaving a hole.[23] Support for this analysis come from sentences like:

(151a) John hummed Δ, and Mary sang Δ, the same tune.
(151b) John gave Mary Δ, and Joan presented Δ to Fred, books which looked remarkably similar.

which show a conjunction of two event descriptions in a way which supports the analysis given above. That is, when the descriptions of the two events are merged, the individuals whose descriptions were moved right are described by a single description which describes how they are related to each other independently of which of the two merged events they came from.

Another example using rightward movement of a constituent is

(152) John $[_{VP}$ $[_{\text{VP-with-NP-hole}}$ drove through $\Delta_i]$
and
$[_{\text{VP-with-NP-hole}}$ completely demolished $\Delta_i]$
$[_{NP}$ a plate glass window$]_i]$.

As analyzed, this sentence presents a difficulty for parsers not present in (150). In that sentence the conjoined syntactic structures had the same form. In (152), the hole is propagated down into a prepositional phrase [*through* Δ] in one conjunct, but not in the other. This makes it difficult for a parser to detect the end of the second conjunct. The first conjunct extends from the beginning of the sentence to the conjunction so it is easy

22. The suggestion that phrases like *John wrote* be considered constituents and be assigned the category (part of speech) S-with-NP-hole (or S/NP) is currently being advocated by Gazdar and his collaborators. In [31], he introduces the suggestion that conjuction provides additional motivation for these "slashed categories". That is, in Gazdar's framework (General Phrase Structure Grammar), it is only possible to conjoin constituents with the same (slashed) category. This condition subsumes the so-called across-the-board condition which blocks conjunction of an S/NP with an S. Gazdar's formulation of conjunction also blocks conjunction of S/NP with S/PP. See [108] for some objections to Gazdar's analysis.

23. This analysis assumes a tranformation like *right node raising* or *heavy NP shift*. See Postal [80, § 4.8] for some introductory discussion on right node raising.

to find. The second conjunct would be easy to find it we knew it was syntactically parallel to the first. Unfortunately, it sometimes isn't. One might try to rectify this by saying that the verb above is really *drove through*, not just *drove*.[24] However, there are other sentences which test this solution. For example,

(153) The tank drove through the side of and completely destroyed my greenhouse.

As written, this sentence is perhaps questionable English. It reads much better with commas after *of* and *destroyed*. To make the conjuncts syntactically parallel we would have to take *drove through the side of* as a verb. This seems implausible.

Rather than say that conjuncts must be strictly parallel syntactically, a more accurate explanation would probably run along the following lines. Conjuncts should be parallel. They should be semantically parallel or should represent a sequence. In addition, they should be nearly syntactically parallel to aid recognition and to support semantic parallelism. When conjuncts are not strictly syntactically parallel, they must have a strong semantic relationship, e.g., *now and in the future*, or there must be extra punctuation and meta-description to help the reader find the second conjunct, as in (154) vs. (153).

(154) The tank drove through the side of, and <u>thus</u> completely destroyed, my greenhouse.

If this is correct, it is somewhat discouraging for the designers of computer parsing algorithms. How is one to define the notion of approximately syntactically parallel? Fortunately, the types of phrases which one sees conjoined are much more limited than what is generated by a general rule. For example, "Mary's (uncle and aunt)" is OK, but "(Mary's uncle) and aunt" is not (with the possible exception of *aunt* meant like *father* in *ask father*), even though this latter phrase is a conjunction of noun phrases. Therefore, it makes sense, as a first approximation, to specify for each syntactic form, the syntactic forms with which it can be conjoined. (If one wants to say that, for example, a noun phrase and a prepositional phrase can be conjoined only when they are used adverbially, e.g., *now and in the future*, then the constraints must be placed on the use of conjoined constituents rather than on their formation.) This allows one to dispense with the question of nearness of match.

A type of sentence which requires assumption of one or more deleted constituents in order to get conjunction of constituents is:

(155) Mary expressed <u>costs</u> <u>in</u> <u>dollars</u> and <u>weights</u> <u>in</u> <u>pounds</u>.

This appears to have two constituents repeated, unless one analyzes it as containing a deleted repetition of the verb[25]

(156) Mary $[_{VP}$ expressed costs in dollars$]$ and $[_{VP}$ $[_{V}$ $]$ weights in pounds$]$.

24. This analysis is known as V-P *incorporation*. It is not an unreasonable position as Bresnan and her collaborators have demonstrated (e.g., [9, pp. 50-60]).
25. This transformation is known as *gapping*.

Postulation of a syntactic form containing deleted constituents is not particularly helpful in constructing a parser to find a syntactic structure. The recovery of deletions has not been a strong point of parsing algorithms. However, it does simplify the representation of conjoined structures if they always involve only the conjunction of constituents. Fortunately, the above example is perhaps the only syntactic structure with conjunction which both occurs frequently and requires an analysis using deletion. It would be wrong to design a parser to handle this kind of conjunction as the typical case.

The representation of conjoined structures as always involving only conjunction of single constituents is perhaps only an approximation. For example, some would argue that

(157) Smith loaned, and his widow later donated, a valuable collection of manuscripts to the library.

is a perfectly good sentence, but it can't be handled by the analysis given above. In designing a parser, thought should be given as to whether one is willing to ignore such rare constructions in order to get a simpler representation of syntactic structure.

8.11 Respectively

TRANSITION!!!

One of the sentence forms used to prove that English cannot be represented by a context free grammar was

(158) Betsy, Tom, and Mary, grew, cooked, and ate, peaches, plums, and strawberries, respectively.

This has the general form $a^nb^nc^n$, which is well known to be a language which doesn't have a context free grammar. Generating all strings which repeat once (i.e., ww) is also beyond a context free language. From this it follows that even

(159) Betsy, Tom, and Mary, grew, cooked, and ate, peaches, respectively.

is not context free if we insist that Betsy grew peaches, Tom cooked them, and Mary ate them. But there is no reason to make either the equality of the number of elements in each conjunct or their ordering a syntactic property, because we also have to accept sentences like

(160) Betsy, Tom, and Mary, grew and cooked, peaches, frequently.

where these constraints don't hold. The word *respectively* can be viewed as putting a constraint on the semantic structure, not the syntactic structure.

8.12 Summary

(not yet written)

9. Parsing

9.1 Introduction

In the linguistics literature it is now popular to claim that English syntax can be described by a context free grammar, as long as syntax is not used to account for regularities better attributed to semantics. The thrust of the last chapter was to suggest that this is also an attractive strategy from the point of view of computational efficiency.

From a practical point of view, it is probably not very important whether every English construction can be described using a context free grammar. As long as most of English is context free it will probably be more efficient to use a parser designed for this sort of grammar, and then patch it up to handle the cases requiring greater than context free descriptive power. In designing algorithms, this approach is almost always better than using an algorithm which is powerful enough to handle the most difficult case even if most cases can be handled by some less powerful algorithm. The goal is to use the best algorithm powerful enough to handle most cases, so that the patches will be secondary to the algorithm and not dominate it.

The main thrust of this chapter, then, will be to explore the parsing of English as a context free language. At the end of the chapter, though, some attention will be given to parsers attributing a more powerful transformation grammar to English.

The parsing problem is one of attributing a structure to a pattern of words, given a description of possible legal structures. In such problems, search for an answer can be driven primarily by the model, primarily by the data, or by a mix of the two. Since syntax trees are generally drawn to branch downward, with the string of words at the bottom, model driven parsing is called top down parsing, data driven parsing is called bottom up parsing. The first task of this chapter will be to explore parsing algorithms which use various mixes of these strategies.

No matter what mix of model and data constraints is employed, the parser will reach numerous choice points. Faced with several alternatives, a search program has several basic questions:

(161a) Depth First Search: One option is picked and explored as far as possible. If this doesn't lead to the solution, the system backs up and tries a second option.

(161b) Breadth First Search: All options are explored in parallel.

(161c) Beam Search: A subset of the options is picked to be explored in parallel and the rest are permanently dropped. A limiting case of this strategy is the picking of one option.

(161d) Best First Search: The search program looks not only at the alternatives presented by the choice point, but at every unexplored alternative it knows about and picks the most promising one. Such a strategy often jumps around the problem, working first here, then there.

(161e) Table of Partial Results: Any of the above strategies can be combined with the formation of a table of partial results. Using such a table a search program can realize it has previously reached the same subproblem and avoid solving it twice.

The second task of this chapter will be examine how these strategies have been applied to the parsing of English.

9.2 Top Down Parsing

Parsing algorithms are perhaps best illustrated by showing how they work step by step given a grammar and a sample input. In order to show all the steps we will choose very simple examples. Recall that grammar (65) assigned the structure

(162) $[_{NP}\ [_{DET}\ a]\ [_{NP\text{-}ADJ}\ [_{ADJ}\ big]$
$[_{NP\text{-}ADJ}\ [_{NP\text{-}N}\ [_{N}\ dog]$
$[^{NP\text{-}N}\ [_{N}\ house]]]]]]$

to the phrase *a big dog house.* A top down parser could find this structure as shown in Figure 65. To function, the parser will need four data structures: the grammar rules, the string to be parsed, a push down stack, and a list of the syntactic structures completed or being built. This latter becomes the output of the parser when it is finished.

The parser begins by placing the distinguished symbol, NP, on the stack. Since NP is not a terminal symbol, it can't match an input symbol. So the parser looks at the grammar to see what can be derived from NP. It finds that there are two production rules NP $\rightarrow$ NP-ADJ and NP $\rightarrow$ DET NP-ADJ. Assuming no parallel computation, one of these possibilities must be chosen to be explored first and the other saved. If the parser wants to find all possible syntactic structures for the input, it must eventually try both. If it is content to find one structure, then the second possibility need only be looked at if the first doesn't lead to a successful analysis of the input.

One way to explore all possibilities in sequence is to put them all on the push down stack in a sequence. (In practice, a pointer to this sequence can be used.) The left most, right hand side in the sequence on the top of the stack is explored first, and this is explored by first examining its leftmost symbol. In Figure 65 a dot has been placed to the left of each symbol currently being worked on at each level of the stack. The leftmost symbol of the leftmost right hand side on the top of the stack in step 2 is NP-ADJ. This is not a terminal symbol, and so a further expansion is done, reaching step 3. Here the symbol under consideration is ADJ. We could expand this out, predicting each adjective. But at this point, even top down parsers generally look at the input and see if the next word can have the predicted part of speech. The parser finds that *the* cannot be an ADJ, and so it has to back up to the next alternative right hand side, as shown in step 4. This process of expanding, checking the input, and backing up continues until step 7 is reached, where the prediction DET is made. Since *the* is a DET, a match is found. The parser (a) begins the construction of a syntactic structure, using the stack as a guide, (b) advances the dot on the top of the stack one symbol to the right, and (c) advances the pointer to the current word of the input one word to the right.

This process continues until either (a) the input pointer passes the last word, or (b) the stack dot passes the last symbol of the current leftmost right hand side, at every level of the stack. When both of these events occur at once, as in step 18, a correct parse has been found. The parser saves away a correct syntactic structure

Fig. 65. Top-Down Parsing

```
<==<fig9-1.press<
        Stack                          Syntactic Structures        String

 1)   NP                                                        a big dog house

 2)   NP-ADJ or DET NP-ADJ
      NP

 3)   ADJ NP-ADJ or NP-N
      NP-ADJ or DET NP-ADJ
      NP

 4)   NP-N
      NP-ADJ or DET NP-ADJ
      NP

 5)   N NP-N or N
      NP-N
      NP-ADJ or DET NP-ADJ
      NP

 6)   N
      NP-N
      NP-ADJ or DET NP-ADJ
      NP

 7)   DET NP-ADJ
      NP

 8)   DET  NP-ADJ                           NP
      NP
                                   DET   NP-ADJ          a big dog house

                                    a

 9)   ADJ NP-ADJ or NP-N
      DET  NP-ADJ
      NP
```

```
10)   ADJ    NP-ADJ or NP-N            NP
      DET    NP-ADJ
      NP                        DET      NP-ADJ        a big dog house

                                 a         ADJ

                                           big
11)   ADJ NP-ADJ or NP-N
      ADJ    NP-ADJ or NP-N
      DET    NP-ADJ
      NP

12)   NP-N
      ADJ    NP-ADJ or NP-N
      DET    NP-ADJ
      NP

13)   N NP-N or N
      NP-N
      ADJ    NP-ADJ or NP-N
      DET    NP-ADJ
      NP

14)   N    NP-N or N                   NP                        a big dog house
      NP-N
      ADJ    NP-ADJ or NP-N     DET         NP-ADJ
      DET    NP-ADJ
      NP                               ADJ     NP-ADJ

                                       big      NP-N

                                                N    NP-N

                                               dog
15)   N NP-N or N
      N  NP-N or N
      NP-N
      ADJ  NP-ADJ or NP-N
      DET  NP-ADJ
      NP
```

```
16)   N  NP-N or N                      NP                   a big dog house.
      N  NP-N or N
      NP-N                           DET   NP-ADJ
      ADJ  NP-ADJ or NP-N
      DET  NP-ADJ                   a     ADJ   NP-ADJ
      NP
                                          big    NP-N

                                               N    NP-N

                                             dog   N    NP-N

                                                  house
17)   N NP-N OR N
      N  NP-N or N
      N  NP-N or N
      NP-N
      ADJ  NP-ADJ or NP-N
      DET  NP-ADJ
      NP

18)    N                                NP                 a big dog house
       N  NP-N or N
       NP-N                          DET    NP-ADJ
       ADJ  NP-ADJ or NP-N
       DET  NP-ADJ                         ADJ     NP-ADJ
       NP
                                           big      NP-N

                                                  N    NP-N

                                                 dog
```

```
19)    N                                NP                  a big  dog house
       NP-N
       ADJ  NP-ADJ or NP-N
       DET  NP-ADJ                   DET   NP-ADJ
       NP
                                      a   ADJ   NP-ADJ

                                          big    NP-N

                                                N    NP-N

                                               dog      N

                                                       house


                                        NP

                                     DET   NP-ADJ

                                      a   ADJ   NP-ADJ

                                          big
20)   NP-N
      DET  NP-ADJ
      NP

21)   N NP-N or N
      NP-N
      DET  NP-ADJ
      NP

22)  N
     NP-N
     DET  NP-ADJ
     NP
```

and either stops (if only the first correct structure is wanted) or backs up to find any additional correct structures. When only (a) occurs, the whole input string has been analyzed as a left segment of a predicted longer string. When only (b) occurs, a left segment of the input string has been analyzed as a whole string. In these cases the parser backs up.

A top down parsing algorithm has the advantage of simplicity, but for the typical grammar, the number of steps executed blows up badly with the length of the string. One can get some insight into what other parsers do by asking where the top down parser goes wrong.

First, note that if the grammar rule NP-ADJ → ADJ NP-ADJ had been written not as a right recursive rule, but as a left recursive rule, NP-ADJ → NP-ADJ ADJ, then the parser would go into an infinite loop, expanding NP-ADJ over and over without ever reaching a terminal symbol. This is what happened in the Harvard Syntactic Analyzer [51], one of the earliest parsers. They first stopped the looping by noting that every time the rule NP-ADJ → NP-ADJ ADJ is applied, a terminal symbol is predicted in the input, albeit not at the left. They stopped the expansion whenever the number of terminal symbols predicted exceeded the number of terminal symbols in the input. Later they discovered that any language having a grammar with left recursive rules also had a grammar with no left recursive rules, just right recursive and center embedding ones. This latter they called Greibach normal form grammar [35]. Greibach discovered an algorithm to convert grammars to Greibach normal form. This was not a practical approach. Greibach normal form grammars described the same language, but not with the same syntactic structures. It was hard to relate the new syntactic structures to the old and thus hard to construct the desired semantic structures. In Chapter 8, various methods of changing the grammar to accommodate parsing were explored. In doing this one has to walk the line between difficulty in parsing and difficulty in constructing a semantic structure, given the resulting syntactic structure.

It is possible for the above top down parsing algorithm to reach the same non-terminal at the same point in the input string by more than one path of prediction. When this happens, the exact same analysis of the input in terms of a phrase in the syntactic category of that symbol will be repeated. One could say that the top down algorithm does not distinguish between the formation of a phrase and its use. It makes separate predictions for every combination of use and formation. This is only required if different uses set different constraints for the formation of a phrase. A context free grammar doesn't do this, and so a faster parser will separate predictions of formation from predictions of use.

When semantics are used to control what phrases can be formed, one has to ask whether the semantic constraints used depend on where the phrase will be used in the syntactic structure. They may not. For example, predicability constraints generally are formulated to be independent of context. They can be employed without destroying the independence of formation and use.

A second problem with this parser is that, at any point in a sentence, it predicts all the possible constructions which could come next. This means that the larger the grammar gets, the slower the parser gets. This need not happen in a bottom up parser, which uses the sentence at hand to select the relevant grammar rules.

A top down parser wouldn't do so much extra work if the grammar always had a terminal symbol as the leftmost symbol on the right side of each production. Then every prediction would be followed by a comparison with the input. In this regard, it is useful to see how a top down parser would handle the alternative formulation of our grammar given in (130). The first few steps in the parsing sequence are shown in Figure 66. The parser starts out by putting the distinguished symbol on the stack as before. It expands this to the finite state graph, as shown in Step 2. There are three arcs leading out of the starting state and so the parser must choose one of these, and then back up for the others. In Figure 66 the arcs have been numbered to indicate the order in which they will be tried. If only the first parse is desired, the arcs can be ordered so as to maximize the chances of getting a parse with the least backing up. (The same ordering can obviously be applied to the production rules of the context free grammar.)

In any case, it is necessary to remember which arcs have been tried so that an untried arc can be picked when the parser backs up. We have shown this being done by pushing copies of the state transition network onto the stack, but it could be implemented by just pushing pointers into the state transition network onto the stack.

Note that with this grammar each prediction step is followed by a test, so the parser does less unnecessary work. In a full grammar recursion is still required, but its use can be restricted either to major phrases, such as the noun phrase, or in the limit, to constructions requiring center embedding.

A top down parser need not proceed strictly left to right. Winograd developed a parser based on systemic grammar which would, for example, look at the final punctuation mark when choosing an expansion of the sentence into a question, imperative, or interrogative. We shall have more to say about this later.

9.3 Bottom Up Parsing

A top down parser predicts by expanding production rules. A bottom up parser reverses this process. It looks for a production rule right hand side in the input and contracts it to a lefthand side. A bottom up parse of *a big dog house* is shown in Figure 67. Instead of using a stack for memory, we illustrate a bottom up parser which uses a well formed substring table, sometimes called a chart.

Starting with the individual words in the input string, the parser seeks to build up phrases which span successively larger portions of the input. This can be done in one pass or in multiple passes. The parser illustrated here uses multiple passes. First it finds all the phrases one constituent long. It does this in steps (1) through (5), by repeatedly using all the grammar rules of the general form X $\rightarrow$ Y. Next it applies rules of the form X $\rightarrow$ YZ to find all the phrases two constituents long which can be made with the phrases one constituent long. In step (7) it finds the new phrases one constituent long which can be made with the results of step (6). It repeats the above steps until no more constituents can be formed. Then it begins to look for phrases three constituents long. There are obviously various ways that this search can be ordered. For example, phrases formed can be ordered by the number of words spanned.

The problem with the bottom up algorithm is that it creates a number of analyses of subsections of the input string which could not possibly fit into a syntactic structure for the whole string. For example, a single noun is analyzed as a complete NP. In a larger grammar we often have a noun phrase which could be

Fig. 66. A Top-Down Parser (Revised)
<= =<fig9-2.press<

Fig. 67. A Bottom-Up Parser

```
<==<fig9-3.press<
1)  a big dog house


    DET  ADJ   N    N

2)   a   big  dog  house

           NP-N   NP-N

    DET  ADJ   N    N

3)   a   big  dog  house



                   NP-ADJ  NP-ADJ

                    NP-N    NP-N

    DET    ADJ        N       N

4)   a     big       dog     house



                      NP      NP

                    NP-ADJ  NP-ADJ

                     NP-N    NP-N

    DET    ADJ         N       N

5)   a     big        dog     house
```

```
   NP-ADJ   NP             NP

              NP-ADJ  NP-N  NP-ADJ

              NP-N          NP-N

    DET  ADJ    N             N

6)   a   big   dog         house

              NP

           NP-ADJ   NP  NP-ADJ   NP

                NP-ADJ  NP-N  NP-ADJ

                  NP-N          NP-N

    DET   ADJ       N             N

7)  the   big      dog          house



            NP             NP

         NP-ADJ    NP    NP-ADJ   NP

                 NP-ADJ  NP-N  NP-ADJ

                   NP-N         NP-N

    DET    ADJ      N             N

8)   a     big      dog          house
```

```
          NP      NP    NP-ADJ    NP

                NP-ADJ  NP   NP-ADJ    NP

                    NP-ADJ    NP-N  NP-ADJ

                    NP-N            NP-N

    DET     ADJ       N               N

9)   a      big       dog             house
```

```
                           NP

                 NP        NP       NP-ADJ     NP

                        NP-ADJ  NP       NP-ADJ   NP

                             NP-ADJ   NP-N      NP-ADJ

                             NP-N               NP-N
```

```
          DET    ADJ       N                 N

10)        a     big      dog               house
```

analyzed as two noun phrases, e.g., *send the chinese teachers* and *send the chinese teachers.* But we never have *chinese teachers* as a noun phrase following *the*, and this the bottom up parser, but not the top down parser, will find.

9.4 Earley's Algorithm and Variations

A parser which combines the virtues of both the top down and bottom up approaches was discovered by Earley [20]. It parses left to right and finds all parses in parallel.

First, it predicts as much as it can top down. As opposed to the top down parsers presented, it separates phrase formation from use. It only predicts a given right hand side will start at a given word in the sentence once, no matter how many ways the corresponding left hand side might be used.

To be more specific, it scans an input string from left to right. As each symbol is scanned, a set of states is constructed which represents the condition of the parsing process at that point in the scan. Each state in the set represents (1) a production such that we are currently scanning an instance of it, (2) a point in that production which shows how much of the production we have scanned, and (3) a pointer back to the position of the input string at which we began to look for that instance of the production. We will represent this triple as a production with a dot in it followed by an integer.

Figure 68 shows the steps in the application of Earley's algorithm to "a big dog house" using the grammar in (68). The position to the left of the first word will be called position 0.

The first step in computing the state set for position 0 is to place the state NP 0 in this set. We then compute the closure. This is the prediction step. The closure of a state set is computed as follows: for each state in the set with a non-terminal N immediately to the right of the dot, add one state for each alternative of N. Put the dot in this state immediately to the left of the first constituent and make the pointer point to the current position. This adds to the state set all predictions which we might begin to look for at this point.

After computing the closure of state 0, the algorithm looks at the first input word (call it "a") and constructs state set 1 as follows: For each state in the state set (state set 0) which has a part of speech of "a" (i.e. DET) immediately to the right of the dot, put a state in the new state set which is the same as the old state except that the dot is moved one symbol to the right, so that that part of speech (DET) is now immediately to the left of the dot. Now we have a new state set which contains all the states which still match the input string. Now we must compute the closure of this new state. The closure is computed as for state 0, but now there is an additional step. For each state which has the dot to the right of the production

(163) $D \rightarrow C_1 ... C_n$ f

Fig. 68. Earley's Algorithm
<= =<fig9-4.press<

take each state in the closure of the state set to which f points of the form

(164) $E \rightarrow \alpha.D\beta\ K$

where α and β are strings. Add

(165) $E \rightarrow \alpha D.\beta\ K$

to the closure of state set f. Intuitively, state set f is the state set we were in when we went looking for that D. Now we have found it, so we go to all the states in f which caused us to look for a D and we move the dot over the D.

In Figure 68, state "NP → .DET NP-ADJ 0" is the only state in state set 0 which can advance to state set 1. Applying the first part of the closure step causes us to get the production rules for NP-ADJ and these cause us to get those for NP-N. The second part of the closure step is not applicable since no production has been completed by state 1. We now compute state 2. Since *big* is an ADJ, we advance "NP-ADJ → .ADJ NP-ADJ 1" to state set 2. We then compute the closure as before. In computing state set 3, "NP-N → N. NP-N 2" and "NP-N → N2" are advanced by *dog*. The first step of closure finds "NP-N→.N NP-N 3" and "NP-N → .N 3." The remaining states are found by the second step of closure. Since "NP-N →N.2" is finished, we have found an NP-N from 2 to 3. We use this to finish "NP-ADJ → .NP-N" and this in turn lets us finish "NP-ADJ → ADJ. NP-ADJ 1," which in turn lets us finish "NP → DET.NP-ADJ 0," which gives us an NP. extending from 0 to 3. State set 4 is computed in a similar fashion.

Earley's algorithm is another example of a well formed substring or chart parser. In addition to making top down predictions which are independent of use, it uses another computation saving technique not used by the bottom up parser presented earlier. Suppose we have a rule of the form NP → NP-ADJ. The earlier parser will enter this rule in the table once for each way that NP-ADJ can be formed. Earley's parser will make only one entry. In English, it is quite common for a string of words to be analyzable as a certain syntactic category in more than one way. For example, this can happen because of the attachment problem as illustrated in

(166a) the man [in the park with a telescope]
(166b) [the man in the park] with a telescope

Given a sentence like *Find the man in the park with the telescope*, the first parser would find two distinct syntactic structures. Earley's parser would find one structure *find NP*, with two choices for NP. One can see that this merging of syntactic structures which have the same use offers one approach to the attachment problem. The explosion of search caused by having to take both branches of an attachment decision is reduced when different courses of action can later be merged back into one. There is thus a choice between trying to delay the attachment decisions completely and trying to limit the explosion of effort caused by branching at the choice point.

A version of Earley's algorithm can also be made to work on networks. The steps are shown in Figure 69. As before, we start out state set zero with the state ".NP 0". Computing the closure, we use the one production. We place an X in the state we are in and a dot in front of each symbol on an arc leading out of this state. We then examine the input symbol, advance, and compute the closure as before (the second step of the closure algorithm is invoked whenever we reach a final state).

Here, at last, is a parsing algorithm which looks pretty reasonable. One doesn't have the feeling that a lot of needless work is being done. The state sets in this example are particularly simple because we have eliminated recursion. In practice, some recursion is required in a full grammar. For example, Figure 69 shows how an NP can be of the form DET N PP. A PP itself is of the form P NP, producing a recursion through NP and PP.

The amount of recursion controls the number of production rules. (With no recursion you have only one rule.) Using a grammar with state transition network right hand sides, one can get the total number of rules down to under 30. Doing this, it is possible to assign each rule a bit in a bit vector. The advantage of this is to avoid some recomputation which Earley's algorithm does when it computes the first step of closure.

In the first step of closure, Earley's algorithm finds, for each non-terminal symbol with a dot in front of it, all the productions which can lead to constructing a syntactic structure in the category of that symbol. This calculation is the same for a given non-terminal symbol, no matter where or how often that symbol has a dot in front of it. This calculation can be greatly simplified by doing some advance calculation of bit vectors. We calculate for each non-terminal symbol a bit vector which has a one bit for every production which is used in computing the non-terminal symbol's closure. Now to get the first step of closure, we simply AND together the bit vectors for all the non-terminal symbols which have a dot in front of them.

Given this vector, we can compute the state set for the first step of closure by setting up the appropriate state for each one bit. But why not save even more by postponing the setting up of a state until that state is needed. We compute, once and for all, the productions which each terminal and non-terminal symbol can start. Whenever we look at an input symbol (use a terminal symbol) or compute the second step of closure (use a non-terminal symbol), then for each production rule which that symbol can start we check the bit vector of the appropriate state set to see if that production rule can be added to it. (We have the option of creating once and for all a bit vector describing the phrases a symbol can start and then ANDing this with the state set bit vector. If the result is zero, we know there is nothing we can start in that state set with that symbol. One's in the result say what can be started.) Using this method, rules are added only when both (a) they are needed and (b) at least the first constituent has been found.

Note that this last technique lets us choose whether we want to add the new states immediately and then look through them along with all the other states in a state set when we are computing the second step of closure or accepting an input symbol, or whether we want to use the technique to trade off a shorter list of states in the state set for looking through the states which a given symbol can start. Since a given symbol usually starts only one or two states, the second alternative looks better, and it gets more attractive as the grammar gets bigger because the number of forecasted states in a state set grows, but the number of states a given symbol starts tends to stay small. This is in part because big grammars involve rare constructions and symbols which are left out of small grammars.

Fig. 69. Earley's Algorithm on Networks
<==<fig9-6.press<

The algorithm illustrated in Figure 69 is again a well formed substring or chart algorithm. The substrings are indexed first by the position of their right ends in the input, and second by production rule. It is also possible to run this algorithm indexing first by production rule and second by word position. This is shown in Figure 70. (In this figure the states have been numbered for purposes of exposition). In this approach a single copy of each state diagram is progressively annotated.

In step 0 nothing has to be done, except to set up the bit vector of needed phrases. After this, the algorithm reads the first input symbol *a*. Finding that this starts a noun phrase and that one is needed, it begins annotating the NP state transition diagram. At state 2, it lists the left and right ends of the string which reached state 2 and a description of the string.

The algorithm then looks at the next input word *big*, which it finds to be an adjective. An adjective, it knows, can take a transition out of state 1 or state 2. Since *big* is the second word, it goes to state 1 to see if there is a string reaching this state and ending at the first word. There isn't. Next it checks state 2 and finds "(0 1 a)", so it further annotates the state to produce step 2. The algorithm continues in this manner.

As with Earley's algorithm, it is frequently the case that a state is reached by two different strings which span the input. Great efficiency is gained if these are merged and a single tail computed for both of these strings.

Is is better to index first by word position, as in Figure 69, or by state, as in Figure 70? Indexing by word position, one goes to that state set and then somehow looks through all the arcs with a dot on them to see if any have a syntactic category of the next word or phrase. This is usually done by maintaining a list of the arcs with a dot and searching through them linearly. Presumably the arcs could be sorted or placed in a hash table. But since few probes are made, this might take longer than linear search. (The only parser I know of where this was done ran very slowly.) This approach will slow down as the grammar expands, if expansion means there are more arcs leading out of each state. It generally does. Also, things slow down when a given word position can be reached many ways. On the other hand, increasing the number of arcs on which a given symbol is used doesn't have much effect.

Indexing by state, it is necessary to look at every state where a given symbol is used. Thus, this method slows down if the same symbol is used on many different arcs. It doesn't make any difference how many arcs come out of a given state and so this method is particularly good for handling rare constructions with special parts of speech. Since the notations are generated left to right, they are naturally sorted by the word positions of their rightmost word. Therefore, only the second phase of update requires much searching through the notations on a given state.

9.5 Representing the State of an Incomplete Search

All the parsing algorithms presented above employ two components, (a) a data structure which describes the grammar, and (b) a data structure which starts off with the input string and is augmented until it eventually contains the parser's output as a separable part. At any point during the parse, the state of the parser is represented by this later structure in three ways:

Fig. 70. A Chart Parsing Algorithm
<= =<fig9-7.press<

(167a) as pointers into the grammar
(167b) as structures which have been built as potential components of the final output
(167c) as a structure built to hold the elements in (a) and (b) in some particular arrangement.

It is easy to see that there are trade-offs possible among these three components. For example, to consider an extreme case, suppose we list, as our grammar, all strings of words less than 50 words long and associate with each string the appropriate parser output. By merging these strings at the left, as shown in Figure 71, we can create a parser whose entire state consists of a single pointer into the grammar. This pushes the wait and see strategy to the limit. All possible strings with a common left segment are identified ahead of time and the state of having seen such a string is represented by a point in the grammar. This requires too many strings to be a practical approach. It would take too much memory to store them. Note that even though each node would branch 1000's of ways, hashing could be used to cheaply find the branch leaving a given node and associated with a given word. The parser would be extremely fast.

So-called <u>semantic</u> <u>grammars</u> go as far in this direction as is practical (if not farther). Figure 72 shows an interaction with the LIFER [39] system for building semantic grammars. Examples of sentences to be handled are:

(168a) What is the age of Ivan Frymire?
(168b) What is the occupation of Jewell Fleming?
(168c) What are the age and occupation of Ivan Frymire?

These sentences can all be recognized as instances of the pattern:

(169) What is/are the *attributes* of *person*?

In creating this pattern it was recognized that in answering the question all attributes are handled the same and all persons are handled the same. Therefore, we can generalize on a semantic basis. The words *is* and *are* are generalized because we choose to ignore the singular/plural distinction. In Figure 72b) the sets <PERSON>, <ATTRIBUTE>, and <IS/ARE> needed to support these generalizations are created.

Fig. 71.
<==<fig9-8.press<

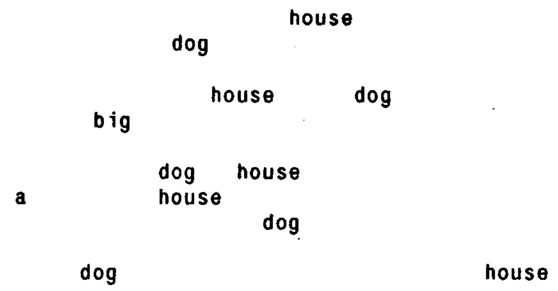


Fig. 72. **Interaction with LIFER**

(170a) Set up data to be queried

```
SETPROPLIST (JEWELL.FLEMING (AGE 35
                              OCCUPATION TEACHER
                              HEIGHT 5.5
                              WEIGHT 105))

SETPROPLIST(IVAN.FRYMIRE (AGE 40
                          OCCUPATION FARMER
                          HEIGHT 6.2
                          WEIGHT 225))
```

(170b) Create sets of words related semantically in this application

```
MAKE.SET (<PERSON> (JEWELL.FLEMING IVAN.FRYMIRE))
MAKE.SET (<ATTRIBUTE> (AGE OCCUPATION HEIGHT WEIGHT))
MAKE.SET (<IS/ARE> (IS ARE))
```

(170c) Define legal patterns of constituents and how to construct the semantic structure of a pattern from the semantic structures of its constituents

```
PATTERN.DEFINE ((WHAT <IS/ARE> THE <ATTR-SET> OF <PERSON>)
                 (MAPCONC <ATTR-SET>
                          (FUNCTION
                           (LAMBDA (A) (LIST A (GETPROP <PERSON> A))))))
PATTERN.DEFINE (<ATTR-SET> (<ATTRIBUTE>) (LIST <ATTRIBUTE>))
PATTERN.DEFINE (<ATTR-SET>) (<ATTRIBUTE> AND <ATTR-SET>)
                            (CONS <ATTRIBUTE> <ATTR-SET>))
```

(170d) Invoke parser and type in example

```
(LIFER.INPUT)
What is the occupation of jewell.fleming
parsed!
(OCCUPATION TEACHER)
```

In Figure 72c) the pattern

(171) WHAT <IS/ARE> THE <ATTR-SET> OF <PERSON>

is defined. This pattern mentions the sets <IS/ARE> and <PERSON>. It also refers to the subpattern <ATTR-SET> which matches one or more attributes separated by *and.* Two further pattern definition statements are used to define this subpattern.

In summary, the semantic grammarian starts out with all the sentences to be accepted and generalizes on semantic categories. Sets and subpatterns are used to recognize when something lies in a generalization.

Using this system, the data base query program LADDER [39] was created. The response time of the LADDER parser was good, even though it was strictly top down, but it became difficult to add new patterns which would not interact with old ones once the system contained roughly 1000 patterns. Also, every application requires different semantic generalizations and so the whole system has to be rewritten for a new application.

The semantic grammar makes a fast parser because it reduces parsing uncertainty. If enough memory is available, nothing can be faster. Unfortunately, the semantic grammar is not a good formalism for presenting a large grammar to a machine. This doesn't rule out generating a semantic grammar from a more traditional description of syntax and semantics.

A syntactic grammar for a parser based on some variation of Earley's algorithm is quite compact. It is easy to store and relatively easy to write. But an intermediate state of Earley's algorithm will typically contain many structures which will not ultimately turn out to be part of the output. Earley's algorithm wastes time constructing what will ultimately turn out to be unneeded structures.

Is there any intermediate ground between the semantic grammar and Earley's algorithm? There is one approach which has been tried several times in different variations, but is not yet completely explored. This approach is based on the observation that when sentences are processed left to right, it is often possible to resolve potential ambiguities by looking ahead only a few constituents. That is, if one elects to wait and see, there is a class of ambiguities where one doesn't have to wait very long to resolve them. In Chapter 8, the notion of wait and see was introduced with the sentences:

(172a) What $\underline{\text{is}}_V$ an ai?
(172b) What $\underline{\text{is}}_{AUX}$ an ai doing in here?

In (172a), *is* is the main verb, while in (172b) it is generally treated as an auxiliary. The significance of the distinction is that it determines whether *what* should be treated as a WH-element to be sent down to a hole.

(173a) What Δ is an ai?
(173b) What is an ai doing Δ in here?

Moving from the left, a parser will not know how to treat *is* when it is encountered, as a main verb or as an auxiliary. Only the word after the NP, *an ai*, will resolve this. In Chapter 8, two alternatives were suggested to deal with this situation. The first was to explore each route in turn. Doing this, one could delay the construction of syntactic structure until one is past the point where back up is likely. That is, one can write the grammar to advance through states collecting constituents and then combine them at points where back up is less likely.

The second alternative given in Section 8.6 was to treat *be* as a special case, taking transitions into a unique series of states which rejoin the main path after the ambiguity is resolved.

A third method has also been proposed [60] which divides the state between position in the grammar and the data structures. The idea is to write grammar rules specifically for recognition which look not only at the next k input symbols, but also at some number of constituents which have already been assembled. A parser using this method would not branch when it reached *is* in (172a-b), neither would it go into a special series of grammar rules. Rather, it would proceed to assemble the noun phrase *an ai* without deciding the use of *is*. Then it would apply pattern-action rules of the form:

(174a) if [WH-NP] [is] [NP] [verb-ing] then <action>
(174b) if [NP] [is] [NP] then <action>

Typically, these rules would be ordered so that if (174a) applies, it would override (174b).

By looking at previously formed constituents, this method avoids having to encode so much information into positions in grammar rules. It pays for this by using rules which look at previously formed constituents more frequently. The first two methods encode the information in the grammar rules. Of these, the first method trades off back up (or non-determinism) for simplicity and conciseness of grammar representation. The second uses a bigger more complex grammar without so much back up.

No experiments have yet been done to show whether it is better to carry everything forward in the grammar state or to match against previously formed constituents.

It might be worth mentioning one final strategy which has not been implemented. Langendoen [54] suggests using a finite state transition network for the entire grammar. But he notes that although languages generated by non-center embedding grammars can be accepted by a finite state machine, that machine can't produce the syntactic structure. Roughly stated, it can't parenthesize the input into phrases because having only a finite number of states, it can't remember for arbitrarily long sentences how many left parens it has already used in order to balance them with right parens. Of course, it can be made to count to some finite depth. In particular, it could count to 1. Beyond this, it will have to use a stack to keep the count. Note that if a semantic structure could be constructed from a syntactic structure with only the left ends of phrases marked, a finite state machine without an auxiliary stack would be useful.

9.6 The Determinism Hypothesis

No one believes that limited look ahead will resolve all the ambiguities which arise in parsing a sentence left to right. For example, in

(175) The cotton clothing is made of grows in Mississippi. *Garden Path*

cotton clothing can be parsed as a compound noun or *clothing is made of* can be taken as a relative clause modifying *cotton*. The correct parse doesn't become apparent until the WH-hole after *of* is reached. Since a WH-hole can be arbitrarily far away from the left edge of a relative clause, a parser would, in general, have to wait arbitrarily long to resolve an ambiguity like (175).

Among those who advocate the wait and see strategy, there are two schools of thought on what should be done. Proponents of the <u>determinism</u> <u>hypothesis</u> claim that it should be possible to parse a sentence left to right (a) without forming any constituents which will not appear in the output, and (b) without resorting to an expansion of the grammar rules of the type used by a semantic grammar, <u>but only if people can also parse the sentence left to right without being aware of taking a garden path</u>. Most people do take a garden path on sentence (175). Thus, no one is claiming that this sentence can be parsed deterministically. The two strategies proposed are:

(176a) Guess, and follow one path. If wrong, recover using a different parsing algorithm (Marcus [60], Shipman [97], Church [17]).
(176b) Never guess, branch whenever a decision can't be made with the look-ahead allowed (Ginsparg [33])

The proponents of guessing are the first to admit that syntax alone will not permit good guessing. Therefore, they propose to follow only one path, but to do as much semantic processing as necessary on that path as the parse proceeds. A guess is made by asking which alternative best fits the syntactic and semantic constraints. Given that there is only one path, they note, matching against several constituents will not be so expensive.

The opponents of this view (e.g., Ginsparg) note that, given the current state of the art, semantic processing is more costly than syntactic processing. Therefore, it is better to find all the syntactic parses before applying the more expensive semantics to further constrain the possibilities.

To implement his deterministic parser, Marcus used a stack of unfinished nodes and a finite buffer. Grammar rules were only allowed to look at constituents in the buffer. Shipman proposes that only a stack is needed if grammar rules are constrained to a finite window of nodes on the top of the stack. Shipman's proposal is illustrated in Figures 73 and 74. Figure 73 shows how items (i.e. *saw* and *yesterday*) which are in the window can be moved onto a constituent and the stack collapsed to fill the holes left by the window. Figure 74 shows how *the* is used to trigger the search for a noun phrase with a corresponding shift in the window.

Fig. 73. Overflow of Buffer Operation
<= =<fig9-10.press<

Fig. 74. Effect of Attention Shifting Rules
<= =<fig9-11.press<

9.7 Parsing Conjunction

Woods [118] discusses an algorithm for parsing conjunction which is appropriate to a top down algorithm like that illustrated in Figure 75. When the algorithm reaches a conjunction, its general state will be as illustrated in Figure 75. Woods' algorithm handles sentences like

(177a) John drove his car through and completely demolished a plate glass window.
(177b) Print best methods to grow and properties of alkali iodates.

Its operation is illustrated by Figure 76. Parsing the sentence proceeds in normal top down fashion until the conjunction is encountered. At this point, the history of the computation looks like the illustration in Figure 75 — that is, the parser is several levels down in recursive computations, each of which is trying to build some constituent, and each of which has made some sequence of transitions up to the constituent it is seeking. The conjunction algorithm selects a configuration from this history (unless filtered out by special rules they must all be tried) and restarts it looking at the string following the conjunction. The configuration which the machine was in when it encountered the conjunction is suspended, to be continued later. The restarted configuration (indicated by the first dashed bracket in Figure 76) will be working on some constituent X (and, indirectly, so is the suspended configuration, although perhaps several levels deeper — see Figure 75).

Fig. 75. Woods' Model of Conjunction
<= =<fig9-12.press<

Fig. 76.
<= =<fig9-13.press<

The restarted configuration will process the string following the conjunction until some point (represented by the second dashed bracket) where it and the suspended computation begin to share the same portion of the input before they each complete a constituent of type X (at the right bracket in Figure 76), at which point they can build a single conjoined X and be combined into a single configuration again. The difficulty of deciding where the suspended configuration is to be continued is handled very nicely in the system by inserting on top of the restarted configuration (chosen from the history) a special CONJOIN configuration which will gain control when the resumed configuration has completed the construction it was building (i.e., the X ending at the right bracket). The parser can thus be turned loose to operate on the restarted configuration in the normal way until it completes the constituent X and transfers control to the CONJOIN configuration. CONJOIN will then look at the path of the new computation between the conjunction and the right bracket and select (all possibilities must be tried) a point in that path at which to continue the suspended computation.

If the point at which the suspended computation is continued has been improperly chosen, then at some point it will not be able to proceed along the trail (i.e., it will not permit one of the transitions that was made on the other half of the conjunction), and the computation will block. In this case, one of the alternative places for continuing the conjunction will be considered. If the computation is successful, it will complete a constituent of type X at the same point in the string where the second conjunct has already completed an X (i.e., at the right bracket) and the two configurations will now be identical except for the type X constituents which they have just completed. At this point the two type X constituents are conjoined into a single one, thus merging the two configurations into a single one for the rest of the parsing.

What this algorithm does to a sentence like (177a) is to form *drove his car through a plate glass window* and *completely demolished a plate glass window* and then conjoin them. In Chapter 8 the advisability of always doing this was questioned, using the example

(178) John gave Mary △, and Joan presented △ to Fred, books which looked remarkably similar.

Note first that this example can't be handled by the algorithm unless we take the gap to be after *Fred* instead of after *presented.* This could actually be justified by noting that even in the absence of conjunction we have a phenomena called heavy NP shift, which allows certain very long noun phrases to be moved to the right end of the sentence. Compare

(179a) Joan presented the biggest book I ever saw in my life to Fred.
(179b) Joan presented to Fred the biggest book I ever saw in my life.
(179c) Joan presented to Fred the book.

Sentence (179b) reads smoother than (179a), but (179c), with a short noun phrase shifted right, is strained. In order to parse (179b), we would need a grammar which could take the hole after *Fred* in (178).

But given this capability, Woods algorithm would still create *John gave Mary books which looked remarkably similar* and *Joan presented to Fred books which looked remarkably similar.* This is incorrect. It is an example of a general problem which arises when one tries to *untransform* a sentence before understanding its logical form.

Clearly, the Woods algorithm is also combinatorially explosive. Unless a well formed substring table is used, many constituents will be refound over and over. In this regard, a similar facility has been constructed for the algorithm in Figure 77. This algorithm is more capable than the Woods algorithm in that it handles conjunctions of phrases within a constituent of a conjunction and the interaction between conjunction and WH- movement. This later is illustrated by the fact that one can say (180a), but not (180b).

(180a) What vegetables$_i$ did Betsy grow and eat Δ_i?
(180b) *What fork$_i$ did Betsy use a knife and Δ_i?

As with the first algorithm, normal parsing is done up to the conjunction. The second conjunction algorithm goes to every state, S_i, which can be the first state of a phrase started by the word following the conjunction. It examines for each S_i every phrase (or partial phrase) constructed by the parse so far, which passes through S_i. For each of these phrases, two cases are of interest. First, the phrase may continue uncompleted out to the conjunction. This is a phrase of the type identified by the Woods algorithm. We *fold it back* to the starting state. This is best illustrated by an example. Suppose we have the phrase

(181) The black and the red balls

to be parsed with the grammar:

(182) <= =<fig9-14.press<

When the conjunction is reached, state 0 will contain the phrase (0 0 NIL) and state 1 will contain the phrases (0 1 the) and (0 2 the black). The word following the conjunction is *the* which is a determiner. A determiner is only used by arcs out of state 0. Going to state 0 the parser finds the phrase (0 0 NIL) which continues out to the conjunction. Thus, it folds this back, adding the phrase (0 3 * the black * and *) to state 0. Normal parsing is then resumed. When the parser picks up the *the* following the conjunction, it goes to state 1 and finds that it can continue the phrase (0 3 * the black * and *) and puts the phrase (0 4 * the black * and * the) on state 1. Next it continues this to (0 5 * the black * and * the red). At this point it needs to know that it has reached a possible right end to the second conjunct. It can know this if it associates with the folded back state (0 3 * the black * and *) all the states where the right end of the second conjunct could reasonably be. Within the limits discussed in Chapter 8, these are fixed by the state where the right end of the first conjunct was. All we need do is associate a list of such states with each state which might contain the right end of the left conjunct. Using this information, the parser can form the phrase (0 5 * the black * and * the red *). It then adds *balls* to this to complete the parse.

In order to handle nested conjunctions with this scheme, one only need generalize it to take more than one list of right end finishing states and to make the rule that right ends must be finished in a first in last out order.

In order to accept (180a) but not (180b), the folding back mechanism checks to see if a WH-element was used in the left conjunct. If it was, the parser obtains from the point of use a list of states where the WH- could be used in the right conjunct and stores this list and the WH- in the folded back phrase. Otherwise, it doesn't permit a WH- in the folded back phrase and thus blocks (180b).

A phrase passing through a state S_i which can be the first state of a phrase started by the word following the conjunction, may not reach the conjunction. It may either die or be completed before reaching the conjunction. If it dies, it is ignored. If it completes, then the parser finds every state S_i which this phrase could start a phrase in, and proceeds as above for these S_i as well.

10. Lexical Syntax and Semantics

10.1 Building Syntactic and Semantic Structures

In this chapter the construction of a grammar of English will be explored. Our goal will be to construct a grammar which makes it easy to translate an input string into a semantic structure. In this regard, it may be helpful to start with a very simple example of how this translation can be done.

The example will use the formalism known as an Augmental Transition Network, or ATN [113]. The ATN parser is top down and the grammar uses finite state right hand sides. A simple ATN grammar is shown in Figure 77. The output of an ATN right hand side can be the string of words and phrases which it has accepted, or, alternatively, the ATN can be made to compute a more complex function of this string. Some possible system structures are illustrated in Figure 78. The structure on the left consists of three separate phases, lexical, syntactic, and semantic. Woods [119] has termed this structure Cascaded ATNs (presumably after cascade filters, where the output of one is the input to the next and each applies additional constraint to the signal.)

Each stage of a cascaded ATN can accept output as it is generated by the stage above. Frequently, a left portion of an input string will be analyzable in more than one way by an ATN stage. When this happens, the next stage will receive multiple inputs. If the next stage can find no analysis for one or more of these inputs, it can send this information back up as constraints to the stage above. Alternatively, each stage can be run to completion before input is given to the next.

To illustrate the operation of an ATN, we will use the single stage form shown in Figure 78b. The grammar processed by our example ATN system is shown in Figure 78. Note that in this grammar (a) a function has been associated with each arc, and (b) a set of registers has been associated with each phrase. Processing of the sentence *the boy patted the dog* may be described informally as follows.

Parsing is strictly top down. The distinguished symbol S is expanded. A copy of the registers to be used in this traversal of the sentence is made. The first arc of the sentence phrase requires a NP and so an NP phrase is started. A copy of its registers is made. Since the first input word, *the* is a determiner, the first arc of the noun phrase can be traversed if the arc function f_5 can be executed successfully. This function puts *the* in the DETERMINER register. Next, *boy* is seen to be a noun, so the arc function f_6 is executed. This function puts *boy* in the NOUN register and SINGULAR in the NUMBER register. Since a final state has now been reached, function f_7 is executed. This function sends the values of the PERSON and NUMBER registers up to the corresponding sentence registers. It uses the contents of the NOUN and DETERMINER registers to construct [BOY-1 #C THE]. A noun phrase having been found, function f_1 is executed. It places the result of the noun phrase construction, [BOY-1 #C THE], in the SUBJECT register. Parsing proceeds in this fashion until function f_4 constructs the final result. To do this it must obtain from memory appropriate senses of *boy, pat* and *dog* and build the final frame.

To flesh out this description, we must build a bigger grammar, and we must discuss the problem of obtaining from memory, and combining, frames for particular senses of words.

Fig. 77. A Simple ATN Grammar
<==<fig10-1.press<

Fig. 78. ATN Output
<= =<fig10-2.press<

10.2 Syntactic Phrases and Semantic Structures

In discussing the construction of a grammar of English, we will provide two sorts of information, examples of phenomena which any grammar might be expected to capture, and a discussion of how these phenomena would be treated by the approach being developed here. Broadly speaking, our approach is to suppress into semantics phenomena which cannot be expressed by a context free grammar. Indeed, to go farther, by using finite state transition networks for the right sides of rules it is possible to have only one rule for each of the traditional phrases.

The principal syntactic phrases are the clause, noun phrase, and prepositional phrase. However, there is no need to postulate a small set of phrase types. There can be a large number of infrequently occurring phrases. For example, *per diem* and *per cent* are now used like nouns, but they still are formed like the prepositional phrases they used to be. One way to handle this is to have a special phrase for this case, as shown in Figure 79. Using a parsing algorithm which starts phrases bottom up, this phrase will only be invoked when the word *per* is encountered. Its presence won't slow down normal processing at all.

Just a few words receive most of the use, there are many words which occur infrequently. The same can be true of grammatical constructions.

Syntactic phrases should be distinguished on the basis of syntactic construction, not on the basis of use. For example, *spreading rumors* has a different use in

(183a) Spreading rumors yesterday wasn't nice of you.
(183b) I hear you were spreading rumors yesterday.

In (183a), *spreading rumors yesterday* refers to a kind of spreading, a process, which is bad manners. In this use, *spreading* would usually be called a gerund, *spreading rumors yesterday,* a gerund phrase. In (183b), *spreading* follows the auxiliary verb *be*. In this use, *spreading* is usually called a first (or present) participle. In (183a), *spreading rumors yesterday* is used like a noun phrase. In (183b), it is used as a complement of *be*, or, more traditionally, one would say that it is a verb phrase minus the auxiliary. Despite these different uses, the order of words in these phrases and their rules of syntactic and semantic combination are the same in (183a) and (183b). Therefore, we should have only one ATN phrase for recognizing this sequence.

Fig. 79. Per diem and Per cent
<==<fig10-3.press<

It follows from this that *spreading* should have the same part of speech in both (183a) and (183b). That part of speech, whatever we call it, will cause *spreading* to be considered for the construction of the phrase *spreading rumors yesterday*. This phrase can then be said to have more than one part of speech if that is the easiest way to explain its use. That is, suppose we have rules X → a A, X → a P, Y → b B, and Y → b P for some phrase P. Then if we agree that P can be either an A or a B (i.e., A → P and B → P are rules), we need only the ATN rules X → a A and Y → b B. The rules A → P and B → P are implemented by the forecasting bit vectors and the lists of parts of speech of the phrase, P.

If two phrases have some initial left segment in common, then these common segments can be merged. For example, suppose we have the rules:

(184a) N → per diem
(184b) PP → per day

These could be merged to form:
<= =<fig-pre-diem.press<

A technical difficulty with this merge is that the combined network should be traversed whenever either an N or PP is needed, but only up to the branch point. Beyond this point, it matters which was needed. This difficulty is avoided if the two forecasts are always made in parallel, or, as happens here, the initial word (i.e., *per*) is only used in this network, so that it is always correct to start this network when the word is seen.

Words in a syntactic phrase may combine semantically in more than one way, e.g., in *chinese teacher*, the word *chinese* either characterizes the teacher or describes the subject matter. But given the syntactic structure, one can give rules for each of the ways (and there aren't many) its words can combine semantically.

10.3 Word Senses

When the parser has determined that a string of words can be assigned the syntactic structure of a particular phrase, we then use that syntactic structure as a guide in assembling a semantic structure. Building a semantic structure involves picking appropriate senses of the individual words.

With these thoughts in mind, let us examine the intransitive verb *run*. As was mentioned in Part I, one of the most basic notions in English appears to be the idea of instantaneous change as expressed mathematically by the first derivative; although, as pointed out by Miller [70], no verb seems to convey this notion alone. *Come* and *go* are quite abstract, but imply direction with respect to a deixis point. We thus postulate a predicate TRANS which means its subject has a first derivative. RUN we take to be a specialization of TRANS which adds to the first-derivative an implication of relative speed, and the presence of some additional systematic, recurrent, or in some sense predictable behavior, e.g., when a machine is *running*, it has a first derivative and a systematic behavior.

A typical dictionary lists over twenty meanings of intransitive *run*. Some of these are shown in Figure 80. The diversity of these *run* meanings shows clearly that RUN is a supra-level concept. so abstract as to convey only the little meaning discussed above. In fact, there are no slots in a case frame which can be usefully specified at the level of RUN. it is too general. The best we could do is to say that the SUBJECT must be a noun. Case frames are assigned to the base level concepts which are specializations of RUN. An important insight is that the fillers of one or two slots often serve to pick the appropriate sense. The *run* in Figure 80 use the follow slots:

(186a) SUBJECT-COMPLEMENT: run aground, run third
(186b) SUBJECT and FREQUENCY: conveyance and every hour
(186c) SUBJECT: stocking runs, tongue runs, line runs, family line runs, vine runs, sam runs, stream runs tears run, butter runs, engine runs, herring run, animal run
(186d) SPECIFIC-PLACE: run in a race
(186e) SUBJECT and TRAJECTORY: ideas run through mind
(186f) DESTINATION: run into debt, run into trouble

The different senses of a verb which are distinguished by a slot can conveniently be notated with a triple (ilk tie cue), when ilk is the verb, tie, the slot, and cue the filler. For example, (RUN SUBJECT BUTTER).

Fig. 80. Dictionary Meanings of Intransitive Run

(187a) stocking runs
(187b) ideas run through the mind
(187c) tongue runs on
(187d) run-into-state: run aground, run into debt, run into trouble
(187e) ~ extend: line runs East, family line runs back, vine runs up wall
(187f) ~ flow: sap runs, stream runs, tears run
(187g) face runs with tears
(187h) ~ melt-and-flow: butter runs
(187i) run-machine: engine runs, boat runs
(187j) ~ make-trips: boat runs every hour
(187k) run-animal
(187l) ~ flee: he ran to Mexico
(187m)~ run-in-race
(187n) place-in-race: he ran third
(187o) run-free: we let Rido run
(187p) run-for-quick-visit: run over to my mother's
(187q) run-fish: the herring are running

It is important to remember that the cue does not define a concept: at most it permits the inheritance of one or more properties. The main role of the cue is to distinguish concepts with the same ilk from one another. For example. the running of streams is quite different from the running of animals. Given *the stream ran* and *the animal ran*, the listener must pick the correct meaning of *run* for each. Since these phrases differ only in the substitution of *stream* for *animal*, these words must be used to pick the correct *run* meaning.

Usually, the specializations of a predicate will be productive. When the listener hears *the milk ran*, he might not have *run* specialized by *milk* in his semantic model, but he would probably have a productive specialization of *run* by *liquid*. Sometimes a sentence will be ambiguous because more than one specialization could apply. For example,

(188) The stream runs to the bottom of the hill.

could be taken parallel to either

(189a) The street runs to the bottom of the hill.
(189b) The stream flows to the bottom of the hill.

depending on whether we recognize a stream as something which has a direction and an extent like a street, or as something which flows. That is, ambiguity can arise when different characterizations of stream pick different meanings of run.

Suppose people identify several prototypes and examples with a particular sense of a word. That word may be used for anything which is a good match for one of the prototypes. When a new thing comes along which doesn't match any of the prototypes too well, we may extend the sense of an existing word.

Wittgenstein [112] has pointed out that senses of a word have a family resemblance - while any two have much in common, little is common to them all. A given sense can be extended in different directions and each of these extensions can be further extended.

In thinking about word sense extension it is helpful to have examples of how different senses of a word are related. Examples of word sense change may be found in Bloomfield [6 §24.1]:[26]

(190a) <u>Narrowing</u>:
Old English *mete* 'food' > *meat* 'edible flesh'
Old English *deor* 'beast > *deer* 'wild ruminant of a particular species'
Old English *hund* 'dog' > *hound* 'hunting-dog of a particular breed'
(190b) <u>Widening</u>:
Middle English *bridde* 'young birdling' > *bird*
Middle English *dogge* 'dog of a particular (ancient) breed' > *dog*
Latin *virtus* 'quality of a man (*vir*), manliness' > French *vertu* (> English *virtue*) 'good quality'
(190c) <u>Metaphor</u>:

26. The greater-than symbol (>) should be read *changed into as the result of a historical processes.*

Primitive Germanic *['bitraz] 'biting' (derivative of *['bi:to:] 'I bite') > *bitter* 'harsh of taste'

(190d) <u>Metonymy</u> (the meanings are near each other in space or time):
Old English *ceace* 'jaw' > *cheek*
Old French *joue* 'cheek' > *jaw*

(190e) <u>Synecdoche</u> (the meanings are related as whole and part):
Primitive Germanic *['tu:naz] 'fence' (so still German *Zaun*) > *town*
pre-English *['stobo:] 'heated room' (compare German *Stube*, formerly 'heated room', now 'living room') > *stove*

(190f) <u>Hyperbole</u> (from stronger to weaker meaning):
pre-French **ex-tonare* 'to strike with thunder' > French *etonner* 'to astonish' (from Old French, English borrowed *astound, astonish*)

(190g) <u>Litotes</u> (from weaker to stronger meaning):
pre-English *['kwalljan] 'to torment' (so still German *qualen*) > Old English *cwellan* 'to kill'

(190h) <u>Degeneration</u>: Old English *cnafa* 'boy, servant' > *knave*

(190i) <u>Elevation</u>: Old English *cniht* 'boy, servant' (compare German *Knecht*) 'servant') > *knight.*

Bloomfield's examples depict changes of meaning over time and are called diachronic comparisons. But senses related in these ways also exist at the same time. For example

(191a) <u>Abstraction</u>: Living things can be divided into plants and animals, animals into people and animals, animals into bugs fish birds reptiles and animals, animals into dogs cats and (real) animals.

(191b) <u>Part</u> <u>Whole</u>: A door can be the frame, and the and swinging part, or just the swinging part. Boston can be Greater Boston or Boston proper.

(191c) <u>Nearness</u> <u>in</u> <u>Space</u> <u>Time</u>: Watergate can refer either to a building or one event which occurred there.

The above discussion makes clear that only some relations between word senses will be amenable to systematic description at any level of detail. It makes clear that while one can guess an approximate sense from a knowledge of a few senses and general principles, to really know a language, one must know a great many senses of its common words.

10.4 The Verb Phrase Nucleus

In the last section it was pointed out that the fillers of one or two slots of a verb often serve to select the appropriate sense. This leads us to advance the hypothesis that some slots are more central to determining the sense of a verb than others. Stripping a simple English sentence of all but the most essential adjuncts, one finds several forms of surface structure, e.g.

(192a) Subject Verb Direct-Object
(192b) Subject Verb Indirect-Object Direct-Object
(192c) Subject Verb Particle Direct-Object
(192d) Subject Verb Particle Prepositional-Phrase
(192e) Subject Verb Complement

(192f) Subject Verb Direct-Object Complement

It seems helpful to view specializations of the verb in terms of these surface categories.

Quirk and Greenbaum [79] distinguish the following classes of verb specializations, (although they do not describe them in precisely these terms).

(193a) The children were sitting down. *Verb, particle, subject*
Did he catch on?
The plane has now taken off.
The prisoner finally broke down.
When will they give in?

(193b) We will set up a new unit. *Verb, particle, direct-object*
Find out whether they are coming.
Drink up your milk quickly.
They turned on the light.
He can't live down his past.

(193c) They called on the man.[27] *Verb, prepositional phrase*
They looked at the picture.
He asked for the waiter.
He lived on rice.
He referred to the dictionary.

(193d) He puts up with a lot of teasing. *Verb, particle, prepositional phrase*
We look forward to your next party.
He stood up for his rights.
She checked up on him.
He broke in on our discussion.
He got away with it.

(193e) The hostess showed me to the door *Verb, direct-object, prepositional phrase*
John put the car into the garage.
We kept them out of trouble.

Mary took advantage of him.[28]
They made good use of the house.
He gave way to the truck.
They made allowance for his age.
They made a fuss over him.
We lost touch with him.

---

27. What is the difference between a preposition and a particle? When the verb is specialized by a prepositional phrase it is not possible to move the preposition after its object (e.g., *They called on the man* $\not\rightarrow$ *They called the man on*). In contrast, when the verb is specialized by a particle and object, the particle can occur after the object (e.g., *They called up the man* $\rightarrow$ *They called the man up*).

28. The analysis of these idiomatic verb phrases is controversial. On the one hand, one can assign them the syntactic structure [$_{\text{V}}$ *took*] [$_{\text{NP}}$ *advantage*] [$_{\text{PP}}$ *of him*] and account for the idiomatic meaning in semantics. Alternatively, once could assign them the syntactic structure [$_{\text{V}}$ *took advantage of*] [$_{\text{NP}}$ him].

(193f) My stocking ran *Verb, subject*
The bill died in committee.
The sun rose at 5:00.

(193g) We shot pool. *Verb, direct-object*
We shot the rapids.
We shot a gun.
We shot a rabbit.
We hit the jackpot.

(193h) He looks good. *Verb, subject complement*
He came a croper.
He turned traitor.

(193i) He put his best foot forward. *Verb, direct-object, object-complement*

In summary we have a rather complete set of possibilities:

(194a) <= =<fig10-7.press<

Whether the filler of a particular slot also picks a sense of the verb depends entirely on the verb and slot in question. One can identify different degrees of dependence between a verb sense and a filler of one of its slots.

For example, the complement of a verb may refer either to the subject or the object. Examples are:

(195a) He felt tired. *current predicate on subject*
He awoke very tired.
He showed up at work very tired.
He showed up at work with a tie on.
He left town a poor man.
Melvin struck me as honest.

(195b) The dog went mad. *resulting predicate on subject*
His brother fell ill.
All my misgivings came true.
Our provisions ran short.
Morris turned socialist.
He became President.
He got into trouble.
I started to go.
(195c) I consider John a good fellow. *current predicate on subject*
(195d) We elected John chairman. *resulting predicate on object*
I forced John to go.
We painted the barn red.
I put the ball in the box.

Although all these sentences may be described as having a complement, the degree to which the complement selects the sense of the verb varies greatly. In some cases the complement plays a central role, while in others it could just as well be left off. Zandvoort [120] cites the progression:

(196a) He awoke <u>very</u> <u>tired</u>. *complete*
(196b) He felt <u>very</u> <u>tired</u>.
(196c) He was <u>very</u> <u>tired</u>.
(196d) He tired! *reduced*

In (196a), *he awoke* would be a complete thought by itself. The complement *very tired* further describes the subject, but it doesn't change the meaning of the verb and it isn't required. In (196b), *he felt* is incomplete. Further *he felt the animal's fur* uses a different sense of *felt*. Here *very tired* both selects a sense of the verb and completes it. In (196c), the verb is reduced in strength so that all of the meaning comes from the complement, while in (196d), the verb is not even present.

We postulate that in (196b) and (196c), the complement is part of the verb phrase nucleus, while in (196a), it is not.

The sentence *He looked at the girl* is an example of specialization by a prepositional phrase. We know it can't involve specialization by a particle and object because the preposition cannot be interchanged with the object.

(197) *He looked the girl at.

However, as Quirk and Greenbaum point out [79], *the girl* intuitively seems to be the object. Indeed, we have the passive construction

(198) The girl was looked at by everyone.

We will model this by saying that *at the girl* is used to select the appropriate specialization of *look*. This specialization treats *the girl* as an OBJECT. Bresnan [10, p. 19] argues for this approach and points out that

(199a) sounds much better than (199b).

(199a) The solution to the problem was arrived at by the class.
(199b) ?The station was arrived at by the train.

This indicates that the more abstract sense of arrive in (199a) takes the *at* prepositional phrase in the nucleus while the other does not.

10.5 Finding the Appropriate Sense of a Verb

Syntactic structures corresponding to the verb phrase nucleus can have the forms and interpretations shown in Figure 81. This figure does not list all of the forms the nucleus can take. Missing are subject and object complements which are not noun phrases or prepositional phrases. Complements will be discussed separately below. Those omitted here aren't confusable with these forms since they have different parts of speech. Also, as mentioned above, a particle can jump over a noun phrase which isn't a pronoun, but we have listed forms with particles only in one position.

There are several things which Figure 81 has to tell us about how to access verb senses. First, all noun phrases which describe the verb are either

(200a) used in the nucleus
(200b) describe time or location in a semi-idiomatic way, e.g., *home, yesterday, Mondays.*

Therefore, it makes sense to locate all the noun phrases describing the nucleus. The ambiguities which do arise will be of the form

(201) John chose yesterday.

Fig. 81.

Example	Syntactic Form	Interpretation
My stocking ran.	NP V	Subject V
He caught on.	NP V P	Subject V P
She checked up on him.	NP V P PP	Subject V P PP
The dog bit the man	NP V NP	Subject V Object
He became a rich man.		Subject V Subject-Complement
Mary made Sam a cake.	NP V NP NP	Subject V Object
Mary made Sam a poor man.		Subject V Object-Complement
Mary made Sam a good wife.		Subject V Subject-Complement
He can't live down his past.	NP V P NP	Subject V P Object
He lived on rice.	NP V PP	Subject V PP
John put the car in the garage.	NP V NP PP	Subject V Object Object-Complement
He gave way to the truck.		Subject V NP PP
I'll make you one up.	NP V NP P NP	Subject V Dative P Object

where *yesterday* is either the direct object of *chose* or it describes the time of the choosing.

Second, a given verb sense may require a noun phrase or prepositional phrase which describes a particular slot, or it may go further and require the use of a particular head noun or preposition. For example. compare

(202a) I put the x {in, on, under} y.
(202b) I gave way to the truck.

The illustrated sense of *put* requires an object and a specific place, but it will take many different descriptions of the object and require only that the specific place describe a place where the object can be put. By contrast, in (202b), the illustrated sense of *gave* requires the use of *way* and *to*.

Third, from Figure 81 we can see that a given syntactic form, such as NP V NP NP, can be interpreted in more than one way.

Fourth, it is extremely rare that more than two constituents after the verb are used in choosing the nucleus. Sentences like

(203a) ?The secretary sent the stockholders out a notice.
(203b) ?The secretary sent the stockholders a notice out.

are pretty rare and many people consider them bad English.

A procedure for finding appropriate senses of the verb might work roughly as follows. For each of the syntactic forms in Figure 81 which pertain to the syntactic analysis of the sentence at hand, check

(204a) whether there is a sense requiring the specific head noun, particle, or prepositions present.
(204b) whether there are one or more senses with each of the possible configurations of grammatical relations.

For each of these senses so found, discard those which reject the particular slots descriptions. For example, consider

(205a) The idea arrived yesterday.
(205b) John chose yesterday.
(205c) John chose a ball.
(205d) *John became yesterday.
(205e) John sneezed a big sneeze.
(205f) John sneezed yesterday.

In (205a), *arrived* is intransitive, so the only analysis is Subject V. In (205b), *choose* is transitive and so we have both Subject V and Subject V Object. In (205c), however, we have only Subject V Object since *a ball* is not a known time or location description; and unlike *yesterday* must be used as the object. Since *became* requires a subject complement, *yesterday* must fill this role in (205d), but since *John* can't be described as *yesterday* this sentence is anomalous. As a transitive verb, *sneeze* requires an object with a head noun of *sneeze*, so (205e) can only be intransitive *sneeze*.

There are two complications which must be observed in carrying out the above process. First, a single time or location can be described by more than one phrase. Thus, we can have

(206a) Put the rug on the floor by the window.
(206b) Put that rug here by the window.
(206c) Here on the floor will be a good location for that rug.

In these sentences there is only a single specific place, but there is more than one phrase describing it. A second complication is that it may be necessary to re-view a description in order to understand how it can fill a slot. For example,

(207) The father leaped 20 feet at a bound.

requires us to assume it is not silly for there to be something which is both a father and can leap. Similarly, the sentence

(208) The ice cream ran across the floor.

requires us to know that ice cream has a liquid phase but no legs. These examples show that it can be difficult to go from a slot filler to a verb sense which takes that filler, since we don't know what properties of the filler the sense requires.

When a verb is transitive, the object usually plays a greater role in selecting the sense than the subject. If there is a particle, it is more important than the object in sense selection. It should be possible to improve the efficiency of sense selection by studying such relationships.

10.6 Productively Related Verb Senses

It is generally assumed that

(209a) The dog bit the man.
(209b) The man was bitten by the dog.

reference the same sense of the word *bite*. Chapter 8 showed how these can be related either by syntactic transformation or lexically through the second participle. The question arises whether the following sets of sentences should be similarly related, even though there is no explicit part of speech like the second participle to signal a transformation.

(210a) I sent a package to Mary. *dative shift*
(210b) I sent Mary a package.

(211a) John broke the window with a rock.
(211b) A rock broke the window.
(211c) The window broke.

(212a) I rode the horse.
(212b) The horse rode well.

(213a) Bees are swarming in the garden.
(213b) The garden is swarming with bees.

(214a) I loaded the truck with hay.
(214b) I loaded the hay on the truck.

(215a) I emptied the bag of groceries.
(215b) I emptied the groceries out of the bag.

(216a) John ate dinner.
(216b) John ate.

(217a) Twenty people can sleep in this car.
(217b) This car sleeps twenty people.

There are four arguments for treating pairs like (210a-b) as referencing separate senses of the verb stored in the lexicon, which senses are known to be related by a productive rule.

(218a) By explicitly storing both forms, we trade space for time in the sense finding algorithm.
(218b) The two uses don't mean exactly the same thing. Using two senses is one way to record this meaning difference. Of course, it can also be recorded by modifications to one sense made at each use, but this probably takes more computation.
(218c) As pointed out by Bresnan [11], certain affixing rules apply to one of the uses but not another. Again, multiple senses make this easier to express.
(218d) Sometimes a verb will have one of the uses in a set but not the other. This is easy to express when all senses used are explicitly placed in the lexicon.

We can illustrate these points with some of the examples given above. Sentences (210a-b) illustrate the relationship known as <u>dative</u> <u>shift</u>. The difference in meaning caused by dative shift is seen in

I sent a package to Houston.
(218e) I sent Houston a package.

Sentence (218b) implies that Houston was the recipient of the package, not merely the location to which it was sent. This implication is not present in (218a). Examples of sentences with a dative but no shifted form are

(219a) Mary bore Mike a son.
(219b) Mary asked Tom a question.

A sentence with a *to* phrase but no dative is

(220) Mary introduced Sam to Mike.

The significance of exceptions is that they don't cast anything extra to handle when everything is explicitly listed.

The prefix *out-* (in the sense of surpass) can be applied to intransitive verbs, but not transitive ones, as illustrated by (221a-b).

(221a) The lamp outshines the candle.
(221b) *The Brownies outfound the Girl Scouts in the treasure hunt.

In summary, since we are already committed to storing and searching a large number of senses of many words, it seems most straight forward to treat the above examples as involving separate senses, for which we understand a general relationship which exists among them. The general relationship will play a role in the learning, not the daily use, of the words.

10.7 Semantic Structure of the Clause

We propose classifying the phrases in a clause as follows:

(222)

Phrase Type	Example
Nuclear	He shot a picture.
Lexical Adjunct	He shot a picture from the hill top.
Adverbial Adjunct	He shot a picture from the hill top using a footstool.
Disjunct	Surprisingly, He shot a picture.
Conjunct	He shot a picture because he was a photographer.

This is classification is based on use. The use of nuclear phrases to pick verb senses has been discussed. Lexical adjuncts describe slots, often present in some verb senses and not others, which are associated with the frame for that verb sense, even though they aren't used to pick the sense. For example, shooting a gun and shooting a rabbit are quite different senses of shoot. They are disambiguated by their OBJECT slot. Both take a description of the SOURCE of the action (*I shot the x from the hill.*) as a lexical adjunct. Description of this SOURCE is optional and doesn't disambiguate the senses.

Adverbial adjuncts describe slots assumed to be present in the referent of the verb sense frame, even though they are not in the verb frame itself. That is, an adverbial adjunct is added to a verb frame to build a larger description, rather than being used to instantiate the frame and it makes sense in terms of the frame of the referent. For example, while it is possible to use a footstool in the process of shooting a picture, one would be surprised to see mention of a footstool in a typical scenario for picture shooting. Indeed, out of context, one is left guessing just how a footstool would be used.

The order of adverbial adjuncts can change the meaning. Compare,

I went home to get a cookie to kill time.
(222c) I went home to kill time to get a cookie.

An important constraint applying lexical adjuncts, but not to adverbial adjuncts, is that, unless conjunction is used, only one filler of a given lexical slot can be mentioned in a given clause. For example, in shooting a picture any number of objects can be used instrumentally, but only one can be picked to be mentioned in the instrument slot of *shoot picture.* Thus (223a) cannot mean that both the flash

(223a) I shot the picture with the flash with the tripod.
(223b) I shot the picture with the flash and the tripod.
(223c) I shot the picture with the flash, using the tripod.

and the tripod were used instrumentally. This meaning can be conveyed either by conjoining two lexical adjuncts, as in (223b) or by using one lexical and one adverbial adjunct, as in (223c).

In his famous paper *The Case for Case,* Fillmore [25] proposes

> The sentence in its basic structure consists of a verb and one or more noun phrases, each associated with the verb in a particular case relationship. The 'explanatory' use of this framework resides in the necessary claim that although there can be compound instances of a single case (through noun phrase conjunction), each case relationship occurs only once in a simple sentence.

Here we propose a slightly different approach than Fillmore. First, we take the existence of one referent per slot as part of the definition of the lexical structure of the verb, and second, we assume that many types of phrases, not just noun phrases, can describe these lexical slots. A sentence element can describe only one slot; however, it may be ambiguous. For example, in

(224) The sandwich was eaten by John.

by John could describe the AGENT, or *by John* could be the PLACE where the sandwich was eaten. We know that John could be the AGENT because the preposition *by* is always used to mark the AGENT in the case frame of the SECOND-PARTICIPLE.

Fillmore's hypothesis is that if two phrases describe the same slot, they must be joined by a conjunction, e.g., in the first of

(225a) The sandwich was eaten by John and the sea.
(225b) The sandwich was eaten by John by the sea.

the sea and John are both AGENT's or both PLACE's, while in the second they must necessarily be one an AGENT and one a PLACE. The requirement for conjunction provides a means of testing properties of slots and proposed new slots. If a sentence element thought to describe a new slot must be conjoined with an element describing a known slot, then the proposed slot is in fact identical with the known one. For example, suppose we felt that a restriction on the AGENT slot was that it had to be animate. The *by the acid* in

(226) The sandwich was eaten by the acid.

would describe a slot other than AGENT while retaining the notion of consumption of the sandwich, acid not being animate. However, by adding *by John* to this,

(227) The sandwich was eaten by John by the acid.

we see that the AGENT sense of *by John* cannot be maintained unless *by John* is conjoined with *by the acid.* Therefore, *by the acid* describes the AGENT slot, no new slot exists and it is false that the AGENT must be animate.

An apparent exception to the rule that only one element describes a slot occurs in the case of TIME, PLACE, and TRAJECTORY. One can say

(228a) I went on Friday at 5 o'clock in the afternoon.
(228b) The ball is on the floor under the table.
(228c) He went through the field along the fence.

Here, however, one is using more than one element to describe the same segment of time, place, or trajectory. The segment is constrained to be in the intersection of the descriptions, e.g., it must be on a Friday, and also at 5 o'clock. This leaves two possibilities, but *in the afternoon* brings it down to one. These slots, then, are an exception only in the sense that more than one element can be used to describe a given time or space segment. If something was done at two distinct times, places, or segments of trajectory, a conjunction must still be used.

A given slot can be described by more than one part of speech. For example:

Example	Part of Speech	Slot Described
I treated her immediately.	adverb	TIME
I treated her yesterday.	noun	TIME
I treated her by noon.	preposition	TIME
I treated her when she came in.	clause	TIME
I treated her surgically.	adverb	MEANS
I treated her by surgery.	preposition	MEANS
I treated her by surgical means.	preposition	MEANS

The reader can verify that the slot conjunction rule holds between different parts of speech as well as between the same parts of speech.

Elements which describe a specific slot such as the subject may be constrained to appear only at specific locations within a given sentence type. This is a great help in resolving ambiguities as to what slot an element describes. Further, slots other than TIME, PLACE, SPECIFIC-SPACE, and TRAJECTORY are described by only one preposition. A sample list of the slots and the prepositions they take is given in Figure 82. Just as with Figure 82. other types of linguistic elements (e.g., words, syntactic structures), it appears that a few slots get most of the use but that there are many rare slots used only by certain verb senses.

Fig. 82. Slots and the prepositions they take

DATIVE	none
SUBJECT-COMPLEMENT	none
TRAJECTORY	(many preps.)
SOURCE	from
ORIGIN	of
DESTINATION	to
INTENDED-DESTINATION	for
METHOD	by
MEANS	by
INSTRUMENT	with
SPECIFIC-PLACE	(many preps.)
SUBJECT	none
AGENT	by
CONSTITUENT	with
TOPIC	about
PLACE	(many preps.)
DURATION	for
TIME	(many preps.)
PURPOSE	to
INTENSITY	none
FREQUENCY	none
QUANTITY	none

Conjuncts and adjuncts can be seen as lying at the extreme of the progression introduced earlier. Recall that while in (229b-c) the main verb is incomplete without the expression *very tired,* in (229a) it can stand alone.

(229a) He awoke very tired.
(229b) He felt very tired.
(229c) He was very tired.
(229d) He tired!

In (230b) the description of his being tired is made a conjunct. As such, it is set even farther apart from the description of waking up. Instead, of describing the waking up, it is introduced as a factor which the reader could have expected would prevent or make less desirable his waking, but didn't prevent it. A sentence like (230a) can be viewed semantically as three frames. A frame describing his waking, one describing his being very tired, and one describing the relation between these two.

(230a) Surprisingly, he awoke.
(230b) He awoke although he was very tired.

Other examples of this sort are given below. The connective gives the order of the arguments and can negate one or both of them.

(231a) IMPLIES (E1, E2)	
If it looks like a rose then it is a rose.	E1,E2
It is a rose if it looks like a rose.	E2,E1
It is a rose unless it looks like a rose.	E2,E1
It looks like a rose therefore it is a rose.	E1,E2
(231b) DID-NOT-BLOCK (E1, E2)	
He was fat, nevertheless, he played well.	E1,E2
He was fat, yet he played well.	E1,E2
He played well, even though he was fat.	E2,E1
He played well although he was fat.	E2,E1
He played well despite being fat	E2,E1
I was tired; however, I kept working.	E,E2
I will have fun whether or not I go to the movies.	E2, or E2, E1
(231c) SET-GOAL-FOR (E1,E2)	
I eat candy because I like it.	E2,E1
I like candy so I eat it.	E1,E2
I like candy, thus I eat it.	E1,E2
I like candy, consequently, I eat it.	E1,E2
I like candy, hence, I eat it.	E1,E2
I like candy, accordingly, I eat it.	E1,E2
I eat candy rather than starve.	E2,E1
I eat candy because of my appetite.	E2,E1
(231d) XOR (E1,E2)	
I can go to Boston or I can go to New York.	E1,E2
Either I can go to Boston or I can go to New York.	E1,E2
(231e) AND (E1,E2)	
I eat and I sleep.	E1,E2
I didn't eat, rather, I kept working.	E1,E2
Boys like girls, conversely, girls like boys.	E1,E2
Boys like girls, similarly, roosters like hens.	E1,E2

Sentence (230a) illustrates the use of a disjunct. A disjunct translates most naturally into a frame which is a comment on the state or process described by the main sentence.

10.8 Semantic Structure of the Prepositional Phrase

The semantics of prepositional phrases admit the same general classification used to describe clauses in (222).

Phrase Type	Example
Nuclear	in spite of the hotel
Lexical Adjunct	within 5 miles of here
Adverbial Adjunct	straight down the road
Disjunct	even in the bedroom

Sometimes one sees the suggestion that a string like *in spite of* should be taken as a single preposition. The approach advocated here brings out the similarity of the different phrase types (the proposition that they have a syntactic similarity is referred to as Chomsky's X-Bar Theory [16, 41]. It points out that *in spite of* is a nuclear phrase, a semantic, not a syntactic, idiom. The prepositional phrase does not take lexical and adverbial phrases as freely as the clause. The possible configurations of modifiers in a prepositional phrase are:

Configuration	Examples
P N	on board
P N NP	on board the ship
P N PP	in front of the house
P NP	at the ship
P NP PP	within 5 miles of here
P PP	out of the house
P	out

These configurations can be modified on the left by sequences of the form (EVEN) (RIGHT) (ADJ). As can be seen in the table above, the prepositional phrase never takes more than two modifiers on the right. If one of these is a noun, it is always nuclear. A prepositional phrase is always lexical, with the lexical slot forcing the preposition to a single choice, e.g., *of* is always used with *within* to give the ORIGIN.

10.9 Description of Space and Time

A primary use of prepositional phrases is the description of spatial and temporal relationships. It is possible to propose a general framework for English spatial and temporal relationships. In this framework an entity is located in time and space by referring it to other entities. I may say that someone on a train stayed where he was (in the seat next to me) even though the train is moving. Every such description is with respect to some frame of reference which is defined by some collection of entities.

Consider physical bodies. At any point in time a physical body has a location. It also has some number of surfaces and enclosures as attributes. For example, *in the house*, *in the back of the house* and *under the house* all refer to enclosures which are attributes of a house. The distinction between a surface and an enclosure is not a question of dimension [3]. Something is *in the front yard* but *on the front lawn* because a yard is an enclosure but a lawn isn't. Something can be *in a sphere* or *in a circle* despite the difference in dimensions. A point is *on a line* or *on a surface.*

One body A can be related to another B by indicating that:

(232a) A and B have the same location; e.g., A at B
(232b) Surface contact; e.g., A on B
(232c) Location of A in some enclosure of B; e.g., A in B

The precise portions of objects which are usually in the specified relationship depends on the individuals involved. A fly can be *on a table* by being on the side. A book could be glued to the side of a table, but we understand a book to be on top when it is *on a table.* A man can be *on the floor* either standing up or lying down, with different parts of his person actually in contact with the floor in the two cases.

A body can also be used to define coordinates, natural bodies can have spherical symmetry (e.g., a ball), cylindrical symmetry (e.g., an icicle), or bilateral symmetry (e.g., a person). When a body has bilateral symmetry it is natural to identify a top and bottom, a front and back, and left and right sides. Having done so one can orient one or more coordinate systems with respect to the body (note the difference between the front of the interior of a church and the front of the exterior). An additional coordinate is provided by the direction of the force of gravity. Using coordinates, a body can be placed a distance in front of, behind, to the left of, etc. another. There is some subtlety in the choice of coordinates for a given English expression. If a girl is lying on her back on the beach, a fly two inches above her knee may be in one of two positions, either two inches over her knee (using a coordinate system based on gravity) or two inches toward her head from her knee (using a coordinate system defined by the girl's body). The girl usually has here long axis lined up with the gravitational up vector, so this can be considered her direction even when she is lying down. Someone may be simultaneously *in front of a chair* with respect to the coordinate system of the chair and *behind the chair* with respect to the coordinate system of someone chasing him.

In addition to locations, surfaces, enclosures, and coordinates, there are directed paths. Directed paths may also be specified using any entity which has a principal dimension, like a river, a wire, or an edge. Any physical body which is translating also defines a directed path, its trajectory, which is the successive locations it occupies in time. Recall that an individual has a temporal structure. The trajectory is a way of viewing the body's translation as an attribute of the body. A *curve ball* can be considered an attributive characterization of a ball with a particular translation attribute.

A directed path may be either in time or in space. The sentence

(233) The lawns intermittently overlap the curbs.

can be taken either *as the weeks go by* or *as we go down the street.*

A directed path has a source and a destination as attributes. Examples of how prepositional phrases describe directed paths are:

phrase	interpretation
to A	destination is location of A
from A	source is location of A
out of A	destination is in exterior of A
from inside A	source is in interior of A
along A	path is defined by a salient dimension of A
through A	path is in an enclosure of A

Examples of how prepositional phrases are used to describe locations are:

A at B	A and B have same location
A against B	A and B have a common boundary; typically B is constrained by A.
A on B	location of an exterior surface or exterior point of A is a surface of B.
A in B	location of A in an interior enclosure of B
A below B	A in a space below B
A under B	A in a space below B and typically constrained by B
A in front of B	A in the space defined by the front of B

To make things more complicated, directed paths may be used to describe locations, and locations may be used to describe directed paths.

One can describe a location with a directed path by indicating that if the directed path is followed, the location to be described will be reached. For example,

(234a) The ball is over the fence. (not hovering)
(234b) The dog was across the street from the little girl.
(234c) The store is straight ahead.

A second method of describing a location with a directed path is illustrated by the sentence

(235) He spread the paint from the window to the door.

The implication is that the object is located over the entire path.

One can describe a directed path with a location by specifying a location through which the path passed or its destination. For example,

(236) Tad jumped on the table.

is ambiguous depending on whether the path of the jump or the location of the jump is being specified. Note that

(237) Tad ran behind the car.

could mean that the location of the running, the destination of the running, or even just a segment of the path of the running, was behind the car.

10.10 Spatial and Temporal Description in the Clause

There are several ways that location and directed path descriptions are used in a clause. They can serve as a SUBJECT.

(238a) In front of the house would be a nice location for the new tree.
(238b) From here to the park is too far to run.
(238c) How far is it from here to the park?
(239a) How far is it from here? *implied destination*
(239b) How far is it to the store? *implied source*

Sentences (239a-b) show that when the SOURCE or DESTINATION is left off of a TRAJECTORY, a deictic location may be assumed.

Some verb senses assume action at a distance described by a path.

(240a) He saw the ship from the rooftop.
(240b) He shot the rabbit from the model airplane using a remote control gun.

Some verb senses assume that the subject or object follows a trajectory.

(241a) The vase fell from the table to the floor.
(241b) He bought the package from Mary.
(241c) He reached Boston from New York.

An ambiguity between action at a distance and trajectory of object is seen in:

(242) He pushed the vase from the table.

A verb sense may mention a specific place occupied by the subject or object.

(243a) He rode in a car.
(243b) Bob threw a smoke screen around himself. *specific place*

A state or process may have a time, location, and duration.

(244) They play baseball for an hour on Sunday afternoon in Central Park.

The subject or object complement of a verb sense may describe a location or a trajectory.

(245a) Bob found a smoke screen around him. *object complement*
(245b) He stayed in the closet.

Bresnan [11] points out that the difference between (243b) and (245) is brought out by the pronouns. In (245), *him* can refer to Bob, while in (243b) *himself* must be used in order to refer to Bob.

Frequently, a verb sense will incorporate part of the description of a place or a trajectory. For example, *disembark* implies motion off of a vessel. This can be further refined by naming the vessel.

(246) We disembarked from the Kennedy.

In some cases, the incorporated information is a default which can be overridden. For example, (247a) implies that Sally went up.

(247a) Sally climbed the ladder. *up*
(247b) Sally climbed down the ladder. *down*

This default direction is overridden in (247b).

10.11 Complements

In Chapter 8 the passive was handled by specifying that two slots shared the same slot filler. In this section, we suggest that this same mechanism can be applied to the subject and object complements. A number of other constructs using this mechanism will then be illustrated.

The frame for the second participle was defined in Chapter 8. It is repeated here for convenience.

(248) <==<fig8-5.press<

(248) says that the OBJECT of the PROCESS which is the SUBJECT-COMPLEMENT of the SECOND-PARTICIPLE is the SUBJECT of the SECOND-PARTICIPLE. Using this same notation, verbs with subject and object complements are described as shown in (249a-b).

(249a) <= =<fig10-14.press<

<= =<fig10-15.press<

Sentence (250) is generally said to contain an existential *there*, which is usually

(250) There is a fly in my soup.

analyzed as a meaningless place marker. In (251) we define a sense of *be*, EXISTENTIAL-BE, which requires *there* for a subject and which has an object complement. This sense of *be* would be selected in (250).

(251) <= =<fig10-16.press<

Sentences (252a-c) require some discussion.

(252a) John was able Δ to please.
(252b) John was easy to please Δ.
(252c) The garden was easy to play games in Δ.

Note that in (252a) John does the pleasing while in (252b) John is pleased. Obviously, this difference must come from a difference between the senses of *able* and *easy*. Sentence (252a) can be handled by defining a sense of *able* which takes a to infinitive, like *to please* as a subject complement. The sentence can then be represented as

<==<fig10-17.press<

The decision to treat PAST-TENSE as the top node will be discussed in the next section.

Sentences (251b-c) show that *easy* requires WH-movement[29] in order to make the appropriate slot identities. If WH-movement is considered a syntactic phenomenon (as has been advocated above), then it will be necessary to define a syntactic relative construction headed by a to infinitive. This will be needed anyway to handle the *know something* sense of

(253) He would be the man to know.

One specific suggestion is to define a sense of *to,* RELATIVE-TO, which takes a bare infinitive as OBJECT and sends down an NP hole which is equated with its SUBJECT. The sense of *easy* in (251b-c) then requires this RELATIVE-TO as its SUBJECT-COMPLEMENT. Thus, (251c) would look like:

29. This movement rule is also known as *tough movement.*

(254) <= =<fig10-18.press<

10.12 Partial and Complete Conversion

In a traditional grammar book, such as Zandvoort [120], it is customary to discuss <u>conversion</u> of a word from one part of speech to another. Thus, for example, from *smoke*, the noun, we get the verb, *to smoke.* From *to smoke a cigarette* we get in turn the noun in *a smoke.*

Zandvoort distinguishes <u>partial</u> <u>conversion</u> from <u>complete</u> <u>conversion</u>. A completely converted word functions entirely as if it were in the new word class. A partially converted word is formed and inflected as if it were in its old word class, but is used in its new word class. Examples of partial conversion are *the poor, her best,* and *my brother's.*

We can argue that complete and partial conversion apply not just to words, but also to phrases. Indeed, Zandvoort notes two examples of partial conversion of a complete sentence

(255a) Young What's his name has got into a bad scrape, I hear.
(255b) All this 'how d'ye doing' is really a bit absurd.

In the second example, the suffix *-ing* is used to effect the conversion. English is an unusual language in permitting so many conversions without affixation, but often an affix is used.

The suffix *-ing* can be used to either partial or complete conversion of a verb to a noun. Look at (256a-d):

(256a) Bob playing the piano was good.
(256b) Bob's softly playing the piano was good.
(256c) Bob playing of the piano was good.
(256d) Bob's playing of the piano was good.

In (256a). *Bob playing the piano* is formed as if playing were a verb and then used like a noun phrase. In (256b) *softly playing the piano* is formed like a verb phrase and then used like a noun phrase without the determiner. On the other hand, in (256d) *playing* is used like a noun in forming a noun phrase. In (256c) *playing* is also used like a noun, and so *Bob* cannot be added as if *playing* were a verb. In the above uses, *playing* is called a gerund. We will refer to the partial and total gerunds.

The suffix *-ing* can also be used to convert or partially convert a verb to an adjective. Partial conversion to an adjective forms what is commonly called the present (or first) participle. Thus, in

(257) Bob is playing the piano.

playing the piano is formed like a verb and used like an adjective. In the usual sense of *Mary is charming* we have complete conversion, in *Mary is charming snakes*, partial conversion.

It might be useful to note that except for the subject noun phrase, the partial gerund and the first participle are formed exactly alike.

There are straight forward, although slightly intricate, rules for locating the subject and object of a partially or totally converted verb. When the verb is totally converted, the subject and object are specified by the *of* and *-'s* genitives as shown in (258a-c):

(258a) the shooting of the hunters	*subject or object*
(258b) the hunter's shooting	*subject or object*
(258c) the lion's shooting of the hunters	*subject and object*

Phrases (258a-b) illustrate how the *of* or *-'s* genitive alone can be interpreted either as the subject or the object. Phrase (258c) shows that when both are present, the *of* genitive must be taken as the object, the *-'s* genitive as the subject.

To find the subject and object of a partially converted gerund, one must first be aware of three ambiguities. Jespersen [42] notes that (259a) can mean either that I remember my grandfather as he was giving me a sovereign or I remember the giving. In the first interpretation we have a noun, *grandfather,* modified by a first participle phrase, *giving me a sovereign.* In the second we have a partial gerund phrase *my grandfather giving me a sovereign.*

(259a) I remember my grandfather giving me a sovereign.
(259b) I laughed at the thought of me going to that hideous chapel.

A second ambiguity is illustrated by

(260) shooting hunters

This can be either a kind of shooting, in which case we have a partial gerund, or a kind of hunters, in which case *shooting* is a totally converted first participle. The third ambiguity is more subtle. The phrase

(261) the hunter shooting

can be read either as *the hunter-shooting* or as *the-hunter shooting*. In the first case, we have a generic type of shooting described by modifying a totally converted gerund with *hunter*. In this total conversion case, hunter can be either the subject or the object. In the second case, we have a partially converted gerund phrase. In this partially converted case, *the hunter* describes the subject as in verb phrase formation, the subject comes before the verb. Note that

(262) The hunter shooting of the lions

has only the first sense of (261). This may be explained by noting that only a total gerund can take *of Tom*, so the partial gerund sense is eliminated.

The above facts are summarized below:

(263a) <u>Partial</u> <u>Gerund</u>		
	easily shooting the lions	object
	the hunters easily shooting	subject
	the hunters easily shooting the lions	subject and object
	the hunter's easily shooting the lions	subject and object
(263b) <u>Total</u> <u>Gerund</u>		
	the easy shooting of the hunters	subject or object
	the hunter's easy shooting	subject or object
	the hunter's easy shooting of the lions	subject and object
	the easy hunter shooting of the lions	object of verb sense *hunter shooting*, in which hunter is subject or object

The facts about the total gerund can be represented by the identification of slots, if we are willing to allow three senses of the total gerund as shown below

(264a) the shooting of [$_{\text{subj}}$ the hunters]	*intransitive*
(264b) [$_{\text{subj}}$ the hunter's] shooting	
(265a) [$_{\text{subj}}$ the hunter's] shooting of [$_{\text{obj}}$ the lions]	*transitive*
(265b) the shooting of [$_{\text{obj}}$ the lions]	
(265c) [$_{\text{obj}}$ the lion's] shooting	

The four way split is based on (a) whether a transitive or intransitive sense of the verb is used and (b) on whether an *of* prepositional phrase is required. The presence or absence of an *of* phrase is easily detected, but the fact that the verb sense can be transitive or intransitive explains the ambiguity of *the hunter's shooting* and *the shooting of the hunters*.

Treating noun phrases in the same way verb phrases and prepositional phrases have been treated, we would have, for example,

(266a) Nuclear	shooting of the lions
(266b) Lexical	the hunter's shooting of the lions

The transitive *of* sense of the gerund, for example, is shown below

(267) <==<fig10-20.press<

The other senses would be represented accordingly. Note that for entities high on the anthropomorphic scale the *-'s* genitive is preferred to the *of* genitive, and vice versa. For example, *Bob's dying* is more natural than *the dying of Bob, the fading of the cloth* more normal than the *cloth's fading*. This can be captured by restrictions on the appropriate slots of the gerund senses.

If one adopts a wait-and-see strategy, the senses of *-ing* might be disambiguated as shown in Figure 83. First, to form phrases, a choice must be made between partial conversion, noun, and adjective. Then when the phrases are used, the partially converted phrase must be used as a noun or adjective, and the total gerund may be disambiguated to count or mass by the inflection *-s* or determiner. Finally, appropriate senses must be picked.

Fig. 83. A Taxonomy of -ing

partial-conversion
- participle: *She is charming snakes.*
- partial-gerund: *That is running a risk*

total-gerund
- *mass*
  - *action: a little crying*
  - *result: a little stuffing*
- *count*
  - *action: a wedding*
  - *result: sweepings*

total-adjective: a charming person

In closing, the reader is asked to compare the TRANSITIVE-OF-GERUND of (267) with the SECOND-PARTICIPLE below:

(268) <==<fig10-22.press<

This sense of the second participle takes a lexical *by* phrase. Just as there is partial and total conversion of the *-ing* form to a first participle or adjective, so there is partial and total conversion of the *-ed* form to a second participle or adjective.

(269a) Mary was easily pleased by John.
(269b) Mary was pleased with John.
(269c) Mary was easily pleased with John.

We see from (269b-c) that the totally converted form can take prepositional phrases other than *by*. Sentence (269c) shows how only the partially converted form can take an adverb and illustrates how the adverb causes the two forms to take different senses of pleased, an unknown agent being implied by (269c) but not by (269a).

10.13 Tense and Aspect

Affixes are divided into prefixes and suffixes. Suffixes which apply *across the board* are called inflections. This is perhaps a weak definition of inflection, but there is no generally accepted definition which is better. To indicate tense and aspect, English uses a system of verb inflections and auxiliary verbs. A version of this system compatible with our development here is presented below:

FRAME	SUBJECT COMPLEMENTS
TENSE	AUX-BE, AUX-HAVE, INFINITIVE
MODAL	AUX-BE, AUX-HAVE, INFINITIVE
AUX-BE	FIRST-PARTICIPLE, SECOND-PARTICIPLE
AUX-HAVE	PERFECT
FIRST PARTICIPLE	AUX-BE, INFINITIVE
SECOND-PARTICIPLE	INFINITIVE
PERFECT	AUX-BE, INFINITIVE

Although this system of inflections is highly regular, different words show forms of the inflections. For example, the normal past tense marker is *-ed*, but some verbs have an irregular past, e.g., *run ran*, and the modals occur only in irregular tensed forms, e.g., *can could*. Using the system above, a sentence like (270) would be analyzed as shown in Figure 84.

(270) The sandwich must have been being eaten.

10.14 The Noun Phrase

The basic syntactic structure of the noun phrase is:

(271) NP → (adverb) (pre-determiner) (DET) (ADJ)* (N)* HEAD-NOUN ({PP, Relative})*

as illustrated by the noun phrase

(272) only half the red fire plugs in Boston I saw
adv pre-det DET ADJ N HEAD-NOUN PP Relative

Semantically, the noun phrase shows the same structure as the other phrases. The head noun serves as the core of the nucleus. It may take elements on its left or right to find the frame which is the semantic nucleus. This frame is lexically modified by elements lying to the left and right of the nucleus, and adverbially modified by elements lying beyond these. This order of elements is:

(273) adverbial < lexical < nuclear < head noun < nuclear < lexical < adverbial

It is important to note that, except for the head noun, the boundaries between syntactic parts of speech do not correspond to the boundaries between nuclear, lexical, and adverbial use of phrases. Examples of each use are shown below.

(274a) Nuclear: *point of view, bull dog, polar bear, horse's ass*
(274b) Lexical: *weather in England, wheat farmer, electrical engineer, the hunter's shooting*
(274c) Adverbial: *book on the table, woman lawyer, useful result, my book*
(274d) Disjunct: *just a puppy*

Ambiguities arise when a phrase can be interpreted as more than one type, e.g.,

Phrase	Nuclear	Lexical	Adverbial
cat house	of ill repute	for cats	
fat man	in the circus		who is fat
old friend		known 50 years	90 years old
woman doctor		doctors woman	is a woman
horse's ass	fool		rear end of
chinese teacher		teaches chinese	is chinese
big top	circus tent		which is big

Fig. 84. The sandwich must have been being eaten.
<= =<fig10-24.press<

brass rat	MIT ring	which is brass

The difference between lexical and adverbial modifiers is brought out nicely by the expression *old friend.* When *old* is understood lexically it modifies the friend frame, describing the length of the friendship. When it is understood adverbially, it modifies the referent of *friend,* saying that he/she is old.

The phrase *in England* is taken as lexical in *the weather in England* since the adverbial form *the weather which is in England* sounds odd. The analysis of Chapter 5 would make weather an attribute of a place. One individuates weather by individuating its place. This makes it more plausible that *weather* takes a lexical location. In Chapter 5 many additional lexical examples were cited from Chomsky [16], who used them to argue for the distinction between lexical and adverbial phrases.

Modifying phrases to the left of the head noun need not be a single word. For example the noun phrase *horseback riding school cafeteria breakfast menu substitution list* has either of the following structures:

(275a) [[[[[horseback riding] school] cafeteria] [breakfast menu]]
[substitution list]]
(275b) [[[[horseback riding] school] cafeteria]
[[breakfast menu] [substitution list]]]

These structures illustrate some typical behavior of such expressions. They branch more to the left than the right (which has the effect of reducing the memory load for a left to right processor), and are sometimes open to alternative interpretations (as above), each of which seems to describe the same entity. At other times the alternative interpretations have different meanings, e.g., *old-home magazine* vs. *old home-magazine, stout major's-wife* vs. *stout-major's wife.* The same precedence between nuclear, lexical, and adverbial modifiers holds within each subphrase.

As stated, adverbial modifiers apply to the referent, not the frame. Recall, however, that some functions use the frame in describing the referent. For example, even though Fred may be a basketball player and a man, to say he is a small basketball player is not the same as to say he is a small man. This is because SMALL uses its subject as a default description of the peer group to be used in computing the distribution of sizes with which the size of the referent of its subject is to be compared. A curious property of English is exhibited by expressions like *big European butterflies.* This can be understood as big for European butterflies, or just big for butterflies. That is, the adverbials *big* and *European* can be understood to apply either in a nested order, or independently. One interpretation of this fact is to say that just as lexical modifiers can help select a sense (in which case they are considered nuclear) so can adverbial modifiers. Thus, one always has the possibility that something in memory will be accessable from European butterflies which does not follow from what is known about European and butterflies independently.

Section 6.8 postulated three mechanisms for understanding a compound noun:

(276a) recognize it, e.g., *bull dog.*
(276b) have a concept for which this would be an appropriate name, e.g., *shoestring end.*
(276c) use analogy, e.g., from *dog house* and *bird house* understand *grasshopper house.*

Let us see how the different modes of understanding are invoked by each type of compound: nuclear, lexical, and adverbial. Nuclear compounds are just recognized, e.g., *bull dog* or else examples like *dog house* and *bird house* are generalized to give the word *house* a slot which creates a new sense for each way it is filled in. Thus, recognition and analogy can be used at the nuclear level. Nuclear compounds truly involve word sense formation.

Lexical compounds, such as *afternoon movie*, assume that the sense of the head has a slot filled by the sense of the modifier. Such a slot can be created by generalization over many cases, but in some instances (like time) it might even be innate. Note that in *student power* the students have the power, while in *apple cake* the cakes have the apples. One might argue that *have* has been deleted from these examples, and in the second case the order of words is reversed. We suggest that, instead, POWER has a slot for what has that power, POWER being an attribute, while CAKE has a slot for its principal flavoring ingredient, this being of key interest in choosing a slice of cake.

The compound *snake poison* can mean either poison for snakes or poison from snakes. It seems reasonable to take the first sense as lexical, many poisons are for killing some particular class of things. The second sense is probably adverbial for most people, as in *poison which is from snakes.* If poison doesn't have a slot for what it is from, how do we find this sense. *Snake poison* falls into a class with *snake skin*, *rabbit foot*, and *horse hair.* One way to capture this generality is to do in the adverbial case what we rejected in the lexical case, postulate a deleted relation. This relation would relate any species of animal to one of its components.

The possibility of deletion is seen in

(277a) She liked her servants to be Church of England.
(277b) I'm the coke.

The latter might be used as a waitress was bringing a tray of drinks to a table of several persons.

In summary, we have proposed three mechanisms for string generalizations used in understanding compounds.

(278a) nuclear generalizations stored in the head word;
(278b) lexical generalizations as slots on word senses;
(278c) adverbial generalizations as relations one element of which is satisfied by a sense of the modifier.

10.15 Multiple Descriptions in the Noun Phrase

A single noun phrase may contain more than one description of the referent, related adverbially. The syntactic relation between these descriptions can take several forms. Examples are:

(279a) Apposition: *my friend, the doctor; the fact you won; Professor Martin*
(279b) Of Genitive: *the state of Texas; the fact of your winning*
(279c) 's Genitive: *Dublin's fair city*

(279d) Compound: *woman lawyer*
(279e) Determiner: *that dog; the dog*
(279f) Conjunction: *the miner and sapper*

When apposition is used, the first member must be recognizable as a form which supports it. For example, after so-called factive nouns like *idea, fact, report,* etc. one can have a clause which spells it out. The genitive constructions are now relatively rare in the language. They can be viewed as apposition where one member takes case markings to indicate the relationship. In a compound one member becomes even more syntactically subservient to the other and this reaches its maximum when one member is reduced to a determiner.

10.16 The Partitive

The partitive construction is used to indicate part of a whole. Examples (280a-b) show that the partitive applies to both count and mass terms.

(280a) some of the little water remaining	*mass*
(280b) some of these 10 strong men	*count*

In each of these examples two entities are mentioned, the whole and a part of it. Significantly, all that can be said about the part is its quantity. The part takes all its other properties from the whole and so it is semantically subservient to it. Example (281) can be viewed as a reduced form of (280b).

(281) some strong men

where the whole is a generic, not separately determined and quantified. We have two descriptions, *some* and *strong men.* Examples (282a-b) move in the opposite direction, attributing properties to both entities.

(282a) a bit of the little water remaining
(282b) a group of these 10 strong men

Such entities as group can be treated either as singular or plural. Semantically related to collectives like group are the container expressions, e.g., *glass of wine.*

The above examples represent taking a physical part from a whole. An analogous type of expression deals with identity. For example, (283a-b) show that *kind of cherries* can be treated as either singular or plural.

(283a) What kind of cherries is best.
(283b) What kind of cherries are best.

10.17 Complementizers and Relatives

It has already been mentioned that the gerund provides a method of converting a verb to a noun. There are in fact several forms which can be constructed from a verb:

Form	Example
partial gerund phrase	Bob describing Mary
gerund phrase	Bob's describing of Mary
nominalization	Bob's description of Mary
bare infinitive phrase	describe Mary
to infinitive phrase	to describe Mary
for complementizer phrase	for Bob to describe Mary
tense	Bob describes Mary
that complementizer phrase	that Bob describes Mary

Each of these phrases can be used in certain syntactic constructions and each has certain possible semantic interpretations. While the syntactic uses are pretty well described in a good handbook of grammar, the semantic interpretations are not so well understood. Nevertheless, a rough sketch can be given.

Syntactically, gerunds and nominalizations behave as noun phrases. Each of the other forms can be used in some subset of the positions which a noun phrase can take.

Since the to infinitive is the only construction where a preposition is followed by a bare infinitive, it seems reasonable to treat this use of *to* as a separate part of speech. This use clearly derives in syntax and semantics from the prepositional *to*, and it would be possible to use the transition network shown in Figure 85 to avoid starting two networks with to. We wait and see what follows *to* before picking its part of speech.

Similarly, the uses of *for* in the for complementizer phrase and *that* in the that complementizer phrase are generally considered a distinct part of speech called a complementizer. Typical uses of these phrases are shown below:

(284a) I know that he is wise	*Object Complementation*
(284b) I prefer for you to speak English	
(285a) That he was alone was obvious from the report	*Subject Complementation*
(285b) For you to leave right now would be inconvenient	
(286a) The idea that nobody will survive is appalling	*Multiple Description*
(286b) The command for all troops to move out was given Friday	
(287a) For his son to enjoy the army, he would have to try very hard	*Adjoint Complementation*
(287b) That his son would not have to join the army, he joined himself	
(288a) In that the command was given	*Prepositional Phrase*

Fig. 85.
<==<fig10-33.press<

To try to make some semantic sense of the above forms it seems worthwhile to have some distinction between a description of an entity and a description of a description of an entity - i.e., a meta-description. The traditional way of doing this is to distinguish <u>events</u> from <u>propositions</u>.

Vendler [102] notes that while one can say either

(289a) The collapse of the Germans was in 1944.
(289b) The collapse of the Germans is a fact.
(289c) The collapse of the Germans was an event.

one cannot say

(290a) *That the Germans collapsed was an event.
(290b) *That the Germans collapsed was in 1944.

but only

(291) That the Germans collapsed is a fact.

In this way, he argues for a distinction between events and facts. He says, for example, that although some people followed the collapse of the Germans, they did not follow the fact that the Germans collapsed.

The phrase *that the Germans collapsed* contains a tensed (or finite) verb. Joos [43] maintains that the finite verb is distinguished in that it <u>makes</u> <u>an</u> <u>assertion</u>. The Kiparskys [47] point out that in comparing

(292a) That the Germans collapsed is tragic.
(292b) That the Germans collapsed is true.

we see that *tragic* presupposes that its subject is a fact, while *true* does not.

From this we are lead to understand Joo's distinction to say that a finite verb describes a <u>proposition</u> which may or may not be presupposed true. The above examples show that a nominalization can describe either a proposition or a fact.

Traditional intuition holds that the same proposition can be described in more than one way. This is why, it is said, that it is important to distinguish between propositions and descriptions which refer to them. However, as was described in Section 2.10, close to right? people in fact don't always realize when two descriptions describe the same proposition, nor does there seem to be an easy way to effect this, and so the idea of using propositions instead of descriptions hasn't been very helpful in reasoning about knowledge and belief.

Perhaps it is better just to say that a tensed verb asserts a particular description. That description can itself be described by using it as a subject or object as shown above in (288)-(292).

The bare infinitive is used to present a generic state or process for consideration or action, as in

(293a) See it!
(293b) See it and you will believe it.
(293c) Me fly?

It is not used to refer to an instance of the state or process, as gerunds and nominalizations do.

The to infinitive is used to present a generic state of being-the-subject-of-a-state-or-process. For example,

(294) To see her is to love her.

means that to be the subject of seeing her is to be the subject of loving her. The for complementizer is used to specify a particular subject for the to infinitive. Thus

(295a) Your leaving right now would be inconvenient.
(295b) For you to leave right now would be inconvenient.

differ subtly in that the first focuses on the leaving being inconvenient while the second focuses on you being the subject of the leaving. Thus the second says more clearly that it is your leaving and not just any leaving which would be inconvenient.

There are three ways that the above phrases can be combined with a noun phrase: multiple description, WH-movement and anaphora. Multiple description has already been discussed in section 10.15; examples are:

(296a) The idea of Sally kissing Bill.	gerund or nominalization (with of genitive)
(296b) The plan to kiss mules is preposterous.	to infinitive phrase
(296c) The plan for Sally to kiss mules	for complementizer phrase
(296d) the suggestion Sally kissed mules	tense
(296e) the suggestion that Sally kissed mules	that complementizer
(296f) the idea of Bob in short pants	of complementizer phrase

This last case can be analyzed as an example of the of genitive followed by what Jespersen calls a nexus. The placing together of two phrases in subject predicate relationship. We prefer to say that both phrases, rather than their combination, are governed by *of*, since no other explicit sign of combination, like tense, is present. Since this is a special use of *of*, it is made a complementizer as were *for* and *to* above.

In WH-Movement, a hole is moved out of the phrase and filled by the noun phrase. Examples are:

(297a) the man to know Δ	to infinitive phrase
(297b) the man for you to know Δ	for complementizer phrase
(297c) the man I know Δ	tense
(297d) the man that I know Δ	that complementizer phrase

In anaphora, a phrase is placed right after the noun phrase. Something in the second phrase refers to the first by anaphora. Examples are:

(298a) the box in which he put the ball Δ
(298b) a man with his wife gone
(298c) a man with five dollars on him

A subtle distinction exists among:

(299a) Albert didn't know what Sally knew Δ.	2 interpretations
(299b) *Albert didn't wash what Sally knew Δ.	0 interpretations
(299c) Albert didn't wash what Sally washed Δ.	1 interpretation
(299d) Albert didn't know what Sally washed Δ.	1 interpretation

Sentence (299a) is ambiguous. It can mean either that Albert didn't know, for example, chemistry, the way Sally did, or it can mean that Albert didn't know the areas of Sally's knowledge. The ambiguity arises because the phrase *what Sally knew* can be taken as an indirect question, or *Sally knew* Δ a modifier of *what.* In either case the noun phrase *what Sally knew* refers to information, and since you can't wash information, (299b) doesn't make sense. For the same reason, (299c-d) each have only one interpretation.

10.18 Comparatives

Examples of comparative constructions occur in:

(300a)
(300b) John is more helpful than Betsy is.
(300c) John is more helpful than Betsy.
(300d) John is a more helpful boy than Tom is.
(300e) *John is a more helpful boy than Betsy is.
(300f) John is as helpful a boy as Tom.
(300g) *John is as helpful a boy as Betsy.

The word *more* indicates a greater quantity or intensity than some reference value. The reference value can be implicit, e.g., *I want more,* or it can be introduced by *than,* e.g., *more than 5, Mary has more friends than just Bill and Pete.* As shown in (300a) what follows *than* need not be an explicit description of the reference value. Instead, the reference value can be specified by a clause from which a hole has been extracted. The hole has the same form as the phrase following the *more* (excluding the plural inflection). That is, in (300a) the hole is *helpful,* while in (300c-d), the hole is *helpful boy.* This causes (300d) to imply that

Betsy is a boy.

This analysis in terms of holes is complicated by examples like (301a-b).

(301a) Mary has better looking dogs than Sam has friends.
(301b) Fido is a bigger dog than Midnight is a cat.
(301c) Mary has more dogs than Sam has friends.

The parallel construction in these sentences allows us to recognize that only the scale being compared is deleted.

Sentence (300b) shows that the comparative clause can also have words deleted. Clearly comparative constructions are difficult to process. Fortunately, as with conjunction, a few deletion patterns make up most of the cases.

Many modifiers describe a scalar attribute. For example *much* refers to quantity, while *long* refers to length. Such modifiers can themselves be determined or quantified as shown below:

how much bread	how useful	how long
as much bread	as useful	as long
too much bread	too useful	too long
that much bread	that useful	that long
so much bread	so useful	so long
very much bread	very useful	very long
more bread	more useful	longer
enough bread	useful enough	long enough
bread enough		

Observe that when *more* modifies a noun it incorporates the sense of quantity.

Several of these determiners, *as, too, so,* and *more*, can take lexical phrases (e.g., *as much as, too much as, not so much as so much that, more than, enough to*) and some of them can be determined or quantified in turn (e.g., *as much too much, as much more*).

The specification of quantities or intensities in terms of a reference amount obviously involves some of the more complex constructions in English. Note the similarity between (300e) and the use of predeterminers in English, as illustrated by

(302a) {half, twice, all, a fraction} the money
(302b) {so helpful, as pretty, too big} a girl

In this construction, a multiple or fraction is followed by a definite noun phrase. The argument to the multiple or fraction is taken as the quantity inferred from the noun phrase. In the similar comparative construction, the adjective describes the scale on which the comparison is specified.

11. Conclusions

(not yet written)

References

1. "Artistotle's 'Categories' and 'De Interpretatione' ", **translated with notes by J.S. Ackrill, Oxford University Press, Ely House, London W.1., 1963.**

2. Benacerraf, P., "What Numbers Could Not Be," *The Philosophical Review*, p. 47, 1965.

3. Bennett, David C., "Spatial and Temporal Uses of English Prepositions. An Essay in Stratificational Semantics," Longman, Green & Co. Ltd, London, 1975.

4. Bierwisch, M., "On Classifying Semantic Features," in D.D. Steinberg and L.A. Jackobovits (eds.) *Semantics*, Cambridge University Press, London, (1971).

5. Black, M., "Identity of Indiscernibles," *Problems of Analysis: Philosophical Essays*, Greenwood Press, 1954, Science 4, pp. 427-455, 1937.

6. Bloomfield, Leonard, "Language," Holt, Rinehart & Winston, New York, 1933.

7. Bobrow, D. G. and Winograd, T., "An Overview of KRL, a Knowledge Knowledge Representation Language", Technical report AIM-293, Artificial Intelligence Laboratory, Stanford University, 1976.

8. Bochvar

9. Bresnan, Joan, "The Passive in Lexical Theory," in Bresnan, Joan (ed.), *The Mental Representation of Grammatical Relations*, MIT Press, Cambridge, MA., 1982.

10. Bresnan, Joan, *A Realistic Transformational Grammar*, in Halle, Bresnan and Miller (eds.), *Linguistic Theory and Psychological Reality*, 1978.

11. Bresnan

12. Carnap, R., "Empiricism, Semantics, and Ontology," *Revue Internationale de Philosophie* 11, (reprinted in L. Linsky (ed.) *Semantics and the Philosphy of Language*)

13. Charniak

14. Chisholm, Roderick M., "Person and Object," Open Court Publishing Co., LaSalle, Illinois, 1976.

15. Chomsky, Noam, "Formal Properties of Grammars," in Luce, Bush and Galanter (eds.), *Handbook of Mathematical Psychology*, Volume II, John Wiley and Sons, New York, pp. 323-418, 1963.

16. Chomsky, Noam, "Remarks on Nominalization," in R. Jacobs and P. Rosenbaum (eds.), *Readings in English Transformational Grammar*, Ginn, Waltham, Massachusetts, 1970.

17. Church, K., "On Memory Limitations in Natural Language Processing," MIT/LCS/TR-245, 1980 (also available from the Indiana University Linguistics Club).

18. Church, K. and Patil, R., "Coping with Syntactic Ambiguity or How to Put the Block in the Box on the Table," MIT/LCS/TM-216, 1982, and AJCL, to appear.

19. Donnellan, K., "Reference and Definite Descriptions," *The Philosophical Review* 75, pp. 281-304, 1966.

20. Earley, Jay, "An Efficient Context-Free Parsing Algorithm," Unpublished Ph.D Thesis, CMU, 1968.

21. Earley, J., "An Efficient Context-Free Parsing Algorithm," Communications of the ACM, Volume 13, Number 2, February, 1970.

22. Eaton, Ralph Monroe, "Symbolism and Truth: An Introduction to the Theory of Knowledge," Harvard University Press, Cambridge, 1925.

23. Evans

24. Fahlman, S.E., "NETL: A System for Representing and Using Real World Knowledge," MIT Press, Cambridge, Ma., 1979.

25. Fillmore, Charles, "The Case for Case," in Emmon Bach and Robert T. Harms (eds.), *Universals in Linguistic Theory*, Rinehart and Winston, 1968.

26. Fikes, Richard E. and Nilsson, Nils J., "Strips: A New Approach to the Application of Theorem Proving to Problem Solving," Artificial Intelligence2, pp. 189-208, 1971.

27. Fodor, J., "Three Reasons for Not Deriving 'Kill' from 'Cause to Die' ", *Linguistic Inquiry*, 1:4, pp. 429-438, 1970.

28. Ford, Marilyn, Bresnan, Joan and Kaplan, Ronald M., *A Competence-Based Theory of Syntactic Closure*, in Bresnan (ed.), *The Mental Representation of Grammatical Relations*, 1982.

29. Frazier, Lyn, *On Comprehending Sentences: Syntactic Parsing Strategies*, PhD thesis, University of Massachusetts, (avaiable from the Indiana University Linguistics Club), 1979.

30. Funk & Wagnalls, *Standard College Dictionary*, Funk & Wagnalls Co., New York, 1963.

31. Gazdar, G., "Unbounded Dependencies and Coordinate Structure," *Linguistic Inquiry*, 12.2, 1981.

32. Gazdar, G., "Phrase Structure Grammar," to appear in P. Jacobson & G. Pullum (eds.) *The Nature of Syntactic Representation.*

33. Ginsparg

34. Graham, S., Harrison, M. and Ruzzo, W., "An Improved Context-Free Recognizer," *ACM Transactions on Programming Languages and Systems*, pp. 415-462, Vol 2, No. 3, July 1980.

35. Greibach, S. A., "A New Normal-Form Theorem for Context-Free, Phrase-Structure Grammars," *Journal of the Association for Computing Machinery*, Vol. 13, pp. 582-587, 1965.

36. Haack, Susan, "Philosophies of Logics," Cambridge University Press, Cambridge, U.K., 1978.

37. Hayes, Patrick J., "The Frame Problem and Related Problems on Artificial Intelligence," Memo 153, Stanford Artificial Intelligence Laboratory, Stanford, Ca., 1971.

38. Hendrix, G., "Encoding Knowledge in Partitioned Networks," in N.V. Findler (ed.), *Associative Networks - The Representation and Use of Knowledge in Computers*, Academic Press, New York, N.Y., 1978.

39. Hendrix, G., Sacerdoti, E., Sagalowicz, D., and Slocum, J., "Developing a Natural Language Interface to Complex Data," ACM Transactions on Database Systems, Vol. 3, No. 2, pp. 105-147, June 1978.

40. Herzberger, 1970

41. Jackendoff, Ray, "X-Bar Syntax: A Study of Phrase Structure," *Linguistic Inquiry Monograph Two*, MIT Press, Cambridge, MA., 1977.

42. Jespersen

43. Joos, M., "The English Verb: Form and Meanings," University of Wisconsin Press, Madison, 1968.

44. Karp, D.J., "General Ontology," Ph.D. thesis, Philosophy Department, Massachusetts Institute of Technology, 1975.

45. Keil, Frank C., "Semantic and Conceptual Development, An Ontological Perspective," Harvard University Press, 1979.

46. Kimball, J., "Seven Principles of Surface Structure Parsing in Natural Language," *Cognition* 2:1, pp. 15-47, 1973.

47. Kiparsky, Paul and Kiparsky, Carol, "Fact", in Steinberg, D. D., and Jakobovits, L. A., *Semantics*, Cambridge University Press, 1971.

48. Kleene

49. Koton, P., SB Thesis, 1980.

50. Kripke, Saul A., "Outline of a Theory of Truth", *Journal of Philosophy*, Vol. 72, 1975.

51. Kuno, Susumo, "The Predictive Analyzer and a Path Elimination Technique", *CACM* 8, 1965.

52. Labov

53. Lakoff, George, "Hedges: A Study in Meaning Criteria and the Logic of Fuzzy Concepts," University of Michigan and Center for Advanced Study in Behavioral Sciences

54. Langendoen, T., "Finite-State Parsing of Phrase-Structure Languages and the Status of Readjustment Rules in Grammar," *Linguistic Inquiry* 6-4, (1975).

55. Lyons, (probably the text book)

56. (Ryle & Mackie)

57. Mathlab Group, "Macsyma Reference Manual," Laboratory for Computer Science, MIT, 1977.

58. Malhotra, Ashok, "Design Criteria for a Knowledge-Based English Language System for Management: An Experimental Analysis," MIT/LCS/TR-146, 1975.

59. Marchand, H., "The Categories and Types of Present-Day English Word Formation," 2nd Edition, Verlag C.H. Beck, (1969).

60. Marcus, Mitchell P., "A Theory of Syntactic Recognition for Natural Language," MIT Press, 1980.

61. Marr, David, *Vision*, W. H. Freeman and Company, San Francisco, 1982.

62. Martin, William, "Interactive Systems - Theories of Implementation," unpublished ms., MIT., date unknown.

63. Martin, William, "Roles, Co-Descriptors, and the Formal representation of Quantified English Expressions," *American Journal of Computational Linguistics*, 7:3, 1981.

64. McCarthy, John, "Recursive Functions of Symbolic Expressions", *CACM*, pp. 184-185, 1960.

65. McCarthy, John, "Epistoemological Problems of AI", *IJCAI-5*, pp. 1038-1044, 1977.

66. McCarthy, John and Hayes, Pat, "Some Philosophical Problems from the Standpoint of Artificial Intelligence", pp. 463-502, Meltzer, B., and Michie, D., (eds.), *Machine Intelligence 4*, Edinburgh University Press, Edinburgh, 1969.

67. McDermott, D.,

68. McDermott, Drew V., and Sussman, Gerald Jay, "The Conniver Reference Manual," AI Memo 259a, 1974

69. McLeod, Dennis and Hammer, Michael, *SDM: A Semantic Database Model*, University of Southern California and Massachusetts Institute of Technology, June 10, 1980.

70. Miller, G.A., "English Verbs of Motion: A Case Study in Semantics and Lexical Memory," in *Coding Process in Human Memory*, A.W. Melton and E. Martin (eds.), V.H. Winton & Sons.

71. Minsky, critique of Schank ???

72. Minsky, probably a k-lines paper

73. Minsky, Marvin, "A Framework for Representing Knowledge," in P.H. Winston (ed.) *The Psychology of Computer Vision*, McGraw Hill, New York, 1975.

74. Montague, R., "On the Nature of Certain Philosophical Entities," *The Monist*, 1969.

75. Moore

76. Moses, Reference to integration, ???

77. Pratt, V. R., "A Linguistics Oriented Programming Language" *IJCAI-3*, 1973.

78. Pratt, V., "Lingol - A Progress Report," *IJCAI-4*, 1975.

79. Quirk, Randolph and Greenbaum, Sidney, "A Concise Grammar of Contemporary English", Harcourt Brace Jovanovich, Inc., New York, 1973.

80. Postal, Paul, "On Raising," MIT Press, Cambridge, MA., 1974.

81. Putnam

82. Quine, Willard van Orman, "From a Logical Point of View," Harper and Row, New York, N.Y., (1953).

83. Quine, 1963, is this the same as Quine53? I think Beth Levin found something written by him with "web of belief" in the title

84. Quine Quine, Which one? Word and Object (1960) From a Logical Point of View (1953), Philosophy of Logic, 1970, Web of Belief?

85. Rescher

86.

87. Rosch

88. Roesch these must be same

89. Ross, H., "Constraints on Variables in Syntax," PhD Thesis, MIT, (available from Indiana University Linguistics Club), 1970.

90. Rhyne, James R., "Lexical Rules and Structures in a Computer Model of Nominal Compounding," Ph.D. Thesis, Computer Science Dept., Univ. of Texas, Austin, 1976.

91. Russell, B., "On Denoting," reprinted in H. Feigl and W. Sellars (eds.) *Readings in Philosophical Analysis*, Appleton-Century-Crofts, Inc., New York, N.Y., 1949.

92. Ruzzo, Walter L., "General Context-Free Language Recognition," unpublished PhD Thesis, University of California, Berkeley, 1978.

93. Ryle & Mackie

94. Schank

95. Schank

96. Schank

97. Shipman, David and Marcus, Mitchell P., "Towards Minimal Data Structures for Deterministic Parsing," *IJCAI-79*, pp. 815-817, 1979.

98. Sommers

99. Strawson, P. F., "Individuals: An Essay in Descriptive Metaphysics," Anchor Press Books Edition, 1963.

100. Sussman, G.J., Winograd, T., and Charniak, E., "Micro-Planner Reference Manual," AI Memo 203, MIT AI Laboratory, December 1971.

101. Tarski, A., "The Semantic Conception of Truth," *Philosophical and Phenomenological Research*, 4, reprinted in L. Linsky (ed.), *Semantics and the Philosophy of Languages*, University of Illinois Press, Urbana, Illinois, 1944.

102. Vendler, Z., "Linguistics in Philosophy," Cornell University Press, Ithaca, N.Y., 1967.

103. Whorf, Benjamin Lee, "Language, Thought and Reality," MIT Press, Cambridge, MA., 1956.

104. Tversky, Amos, "Features of Similarity," *Psychological Review*, 84:4, July 1977.

105. Wales, Roger and Toner, Hugh, "Intonation and Ambiguity," in Cooper and Walker (eds.), *Sentence Processing*, Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1979.

106. Waltz, David, "Understanding Line Drawings of Scenes with Shadows," in Winston, P., (ed.), *The Psychology of Computer Vision*, McGraw-Hill Book Company, New York, 1975.

107. Weizenbaum, J., "ELIZA," *CACM*, pp. 36-45, 1966.

108. Williams, Edwin S., "Transformationless Grammar," *Linguistic Inquiry*, 12:4, pp. 645-654, 1981.

109. Winograd, Terry, "A Computer Program for Understanding Natural Language," TR-17, MIT Artificial Intelligence Lab., Cambridge, Mass., 1971.

110. Winston, P. H., "Learning Structural Descriptions from Examples," PhD Thesis, MAC-TR-76, MIT, Cambridge, MA., 1970.

111. Wittgenstein, L., "Philosophical Investigations," MacMillan, New York, (1953).

112. Wittgenstein, chapter 10

113. Woods, William, "Transition Network Grammars for Natural Language Analysis," *CACM*, Volume 13, Number 10, October, 1970.

114. Woods, W., *What's in a Link*, ???

115. Woods, W., "Semantics and Quantifications in Natural Language Question Answering," Bolt, Beranek and Newman Report 3687, 1977.

116. Woods, W., 1978

117. Woods, Hendrix

118. Woods

119. Woods, William A., "Cascaded ATN Grammars," *American Journal of Computational Linguistics*, 1980.

120. Zandvoort. R.W., "A Handbook of English Grammar," Longman, Green & Co. Ltd., London, 1957.