SYNTAX DIRECTED INTERPRETATION OF NATURAL LANGUAGE

By
E. Stephan Coles

CENTER FOR THE STUDY OF INFORMATION PROCESSING
CARNEGIE MELLON UNIVERSITY
ADVANCED RESEARCH PROJECTS AGENCY
DEPARTMENT OF DEFENSE
SYNTAX DIRECTED INTERPRETATION

OF NATURAL LANGUAGE

By

L. Stephen Coles

Submitted to the Carnegie Institute of Technology
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

Pittsburgh, Pennsylvania
1967

This work was supported by the Advanced Research Projects Agency
of the Office of the Secretary of Defense (SD-146) and is monitored
by the Air Force Office of Scientific Research.
This dissertation presents a method called syntax directed interpretation which permits the use of semantic information in a syntactic analysis of sentences taken from a restricted domain of natural language. This method is used in the resolution of syntactic ambiguity of English sentences with respect to a variable but well defined universe of discourse.

The first chapter defines the problem and surveys previous work related to natural language and question-answering systems.

In chapter two we describe an integrated approach to natural language processing which combines both syntax and semantics within the framework of a natural inference system. The logical properties of completeness and consistency are demonstrated for such systems.

In chapter three a computer program (GRANIS) written in Formula Algol and using the method of syntax directed interpretation is described. A picture, which can be input directly via a graphic display console, serves as a context within which syntactic ambiguity of input sentences can be resolved.
Chapter four considers various extensions of GRANIS in the directions of greater habitability, inferential power, knowledge acquisition, and adaptability. Chapter five concludes with a summary of results and their implication for future work in this field. The program listing in Appendix 2 forms an integral part of this thesis, and some effort was taken to ensure that it would be readable.
I am indebted to Professors Alan J. Perlis and Herbert A. Simon for their criticism and guidance in this work. I am also grateful to Renato Iturriaga, Rudolph Krutar, and Tim Standish for their assistance with Formula Algol, which played an important role in the implementation of the ideas presented in this thesis as a running program.

The research reported here was supported by the Advanced Research Projects Agency of the Department of Defense under contract SD-146 to the Carnegie Institute of Technology.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Title Page</th>
<th>Abstract</th>
<th>Acknowledgments</th>
<th>Table of Contents</th>
<th>List of Figures</th>
<th>List of Tables</th>
<th>Chapter 1. INTRODUCTION</th>
<th>Chapter 2. AN INTEGRATED LINGUISTIC DESCRIPTION</th>
<th>Chapter 3. GRANIS, A COMPUTER MODEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>ii</td>
<td>iv</td>
<td>v</td>
<td>vii</td>
<td>viii</td>
<td>1</td>
<td>22</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.1 Definition of the Problem</td>
<td>1</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.2 Reasons for Wanting Natural Language and Picture Input</td>
<td>4</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.3 Related Systems</td>
<td>10</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Chapter 2. AN INTEGRATED LINGUISTIC DESCRIPTION</td>
<td>22</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.1 The Syntactic Component</td>
<td>23</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.2 The Semantic Component</td>
<td>26</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.3 The Relation Between Syntax and Semantics</td>
<td>29</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.4 The Pragmatic Component</td>
<td>33</td>
<td>54</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.5 Productions</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.6 Natural Inference Systems</td>
<td>38</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Chapter 3. GRANIS, A COMPUTER MODEL</td>
<td>53</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3.1 Program Structure</td>
<td>54</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3.1.1 The Production System</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3.1.2 The Dictionary</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3.1.3 The Universe of Discourse</td>
<td>63</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3.1.4 Evaluation</td>
<td>73</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3.2 Results</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td>Figure Description</td>
<td>Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. Basic NOR Circuit</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Geometric Figures</td>
<td>69</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Geometric Figures</td>
<td>79</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. Geometric Figures</td>
<td>80</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. Geometric Figures</td>
<td>81</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6. Geometric Figures</td>
<td>82</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7. Geometric Figures</td>
<td>83</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8. Diode Logic</td>
<td>84</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9. NOR Circuit</td>
<td>85</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10. NOR Circuit</td>
<td>86</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11. Low Level Logic</td>
<td>87</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12. Current Mode Logic</td>
<td>88</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13. Aniline</td>
<td>89</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14. Chlorobenzene</td>
<td>90</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15. Sulfapyridine</td>
<td>91</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16. Benzoic Acid</td>
<td>92</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17. Benzaldehyde</td>
<td>93</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18. Phenol</td>
<td>94</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19. Nine Geometric Patterns</td>
<td>95</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
LIST OF TABLES

Page

1. BNF Rules to Formula Productions .................. 56

2. Display Commands ..................................... 65

3. Logical Translations ................................. 77

4. Structure of English Tenses ......................... 123
Chapter 1. INTRODUCTION

1.1 Definition of the Problem.

The aim of the research reported here was to investigate how one might construct a computer program capable of communicating with people by means of natural language and elementary pictures taken from a restricted universe of discourse. During the course of this investigation a computer program called GRANIS (GRAphical Natural Inference System) was written. GRANIS accepts a restricted subset of English as input together with elementary line drawings which are input directly to the computer by means of a graphic display console. The problem areas considered for GRANIS were plane geometry, electrical circuits, and organic chemistry; these three diverse areas serve to demonstrate the generality of GRANIS in handling widely varying domains. The example shown in Figure 1 illustrates some of the communication and problem-solving capabilities of GRANIS in the area of electrical circuits.

There are two fundamental problems in the construction of a computer model that can make inferences about pictures and natural language. First, it is necessary to perform a syntactic analysis of the source information, i.e., both for the picture and the natural language text. Second, an internal representation for both the picture and text information must be found that facilitates problem solving in that domain. Although each of these problems is difficult in its own right,
INPUT SENTENCES:

1. Each resistor in parallel with a capacitor which is ten ohms is an input resistor.
2. Each resistor in parallel with a capacitor which is ten micro farads is an input resistor.

ANSWERS:
Sentence 1 is TRUE for the above circuit.
Sentence 2 is FALSE for the above circuit.

Basic NOR Circuit

FIGURE 1
the difficulty is further compounded because it is impossible to isolate completely the syntactic analysis problem from the semantic analysis problem. This unavoidable interaction between syntax and semantics in natural language is illustrated in the following two sentences suggested by Raphael [63, p. 141]

(1) Bring me the bottle of milk which is sour.
(2) Bring me the bottle of milk which is cracked.

Our knowledge of bottles and of milk and of what it means to be sour and cracked lead us intuitively to discover the intended parsing of the two sentences. That is to say, we unhesitatingly associate the relative pronoun "which" with its antecedent "milk" in the first case and "bottle" in the second. Indeed, the first example in Figure 1 exhibits this same kind of syntactic ambiguity, since anyone unacquainted with the properties of resistors and capacitors would be tempted to search for ten ohm capacitors in the circuit! The second example, however, does require one to look for ten microfarad capacitors. We will have more to say about the resolution of ambiguity in Chapter 2.

In summary, the main concern of this thesis is to explore the relationship between syntax and semantics in natural language as it relates to inference making in some restricted universe of discourse. Pictures provide a context within which the lexical and syntactic ambiguity of input text can be resolved.
1.2 Reasons for Wanting Natural Language and Picture Input.

Why would one want to talk with a computer in English? Many formal computer languages such as FORTRAN, ALGOL, LISP, and COBOL already provide a capability for communicating with the computer over a wide variety of problem areas. Indeed, some have even suggested that because of ambiguity, English would be an undesirable language for computer interaction (for example, see Zemanek [90, p. 141] and the discussion in [68]). In addition, the richness of natural language poses enormous memory requirements for the computer. As Paul Garvin recently pointed out:

As far as the structural characteristics of natural language go, perhaps one simple and completely trivial statement can cover it: They are rather complex. They consist of entities of different kinds. To give you a crude example, letters in script are one kind of entity; words with spaces on each side are another kind; sentences are a third kind. All these have describable characteristics, which are rather complex, and which, let us say, make natural language into a system of a much greater complexity than other communication systems, such as gestures or traffic signals.

The relative complexity of natural language such as English derives from various sources, including syntactic ambiguity, logical ambiguity (amphibolity), suppression of secondary premises (enthymeme), grammatical abbreviation (brachylogy), the fusion of different components of meaning, and the relative abundance of resources for paraphrase. A further type of complexity results from the fact that, viewed logically, a natural language such as English is not a single system. English, for example, includes both metalinguistic and object language expressions, and even the object language portion does not constitute a single extensional system. In spite of these difficulties,

however, there are several important reasons for wanting natural language as a mode of communication.

First, the computer would become immediately accessible as a problem solving tool for those people not conversant with any computer language appropriate to their needs. In addition, as Halpern [30] points out, the programming community itself would benefit. Since the very beginning of man-computer interaction there has been a steady movement away from lower-level to higher-level programming languages. The systematic evolution from machine-dependent languages to more sophisticated macro-assembly languages, culminating in machine-independent, compiler and interpretive languages points to a definite trend toward the full richness of natural language. It is this evolution which provides the perspective with which to view natural language processing.

A second equally compelling reason is that certain applications such as text-based information retrieval are best expressed in natural language; any other representations would require a manual translation process partially negating the advantage of using a computer in the first place.

Third, today's programming languages are process oriented rather than description oriented. This makes it impossible at present for a human to discuss his problem with a computer in its formative stages, i.e., when it is not yet well defined. Indeed, for most pure mathematicians, a problem which has been sufficiently formalized to permit computer assistance is no longer an interesting problem! With natural language input on the other hand it would be possible to engage the computer in a dialog, permitting the
user to describe his problem, while the computer questions assumptions and aids in the process of jointly converging on a mutually acceptable well defined problem for later solution. Engelbart [22], Licklider [42], and Yershov [89] each give their conception of what such a dialog might be like, and have provided much of the underlying motivation for the work undertaken here. The inherent ambiguity of natural language should not be thought of as a disadvantage in such a dialog. To the contrary, ambiguity that is resolvable within the context of the conversation is a virtue, since it permits greater parsimony in the source statements. Ambiguity is one of the major features that distinguish natural from formal languages, and it actually contributes to the expressive power and flexibility of our language. Even ambiguity that is not currently resolvable can be used to direct the conversation in the further specification of the problem to be solved.

Finally, natural language can contribute in an important way to the generality of artificially intelligent systems. There is much evidence to indicate that no one global internal representation is suitable for widely varying problem domains (for example, see Newell and Ernst [51]). Thus, rather than continually expanding the problem-solving power of one representation, another approach to generality, sometimes referred to as the "big switch theory" of artificial intelligence, requires some device immediately beneath the surface that will call upon the correct representation once the context has been determined. This approach frequently gives the impression of a system of considerably greater capability than one in which the switching must be done manually be the user. It is suggested that natural language frequently functions in the role of establishing
contexts and switching between representations, and therefore can be used to improve the generality of problem solving systems by providing a framework within which such switching can be performed automatically by the system. Moreover, natural language is essential if we ultimately hope to solve significant artificial intelligence problems such as the "Imitation Game" originally proposed by Turing [80] in 1950, and which is now commonly referred to as the "Turing Test".

Aside from natural language text, what about pictures? The importance of pictorial input is clearly evident in the technical journals that humans use to communicate with each other. Blackboards again illustrate the same point. Two important applications of pictorial and natural language input to a computer lie in the areas of information retrieval and automated teaching. Information retrieval on a large pictorial data base is a significant possibility for the future. Four immediate examples of such a large pictorial data base are

1. a complete set of architectural blueprints for a complex structure.
2. a set of schematic circuit diagrams for a city-wide electrical power distribution system.
3. a set of design blueprints for an automobile.
4. a set of flow charts for a business data-processing program.

In any particular instance each of these examples may involve a collection of several thousand schematic diagrams. At present locating a relevant diagram can frequently be an extremely time consuming process. One can envisage structural design of the future, however, as a large number of engineers creating
blueprints on display terminals that are dynamically incorporated into a large body of related design information stored in the mass memory of a computer. This would virtually eliminate the present need for draftsmen. Moreover, the consistency of proposed interfaces and conventions would be established immediately during the design itself, since the system would dynamically verify the consistency of each proposed increment to the data base with respect to its current state. In this fashion the design process itself would be considerably accelerated.

The ability to interrogate the pictorial data base in terms of simple English sentences means that the designer need not learn a special computer language to operate on the data. This is important for two reasons. Not only is it more convenient for those designers not already acquainted with appropriate computer languages, but for all designers English would provide a superior vehicle for interrogating the data than any formal computer oriented language.

In automatic teaching programs the ability of the student to interact with the system in terms of a dialog opens up a host of new possibilities. Present automated teaching systems suffer from the limitation that they require the student to select from among a small number of prescribed alternatives in replying to any particular question. A static decision about whether to present the next question in the series, return to an earlier question, refresh the student with some previous material, skip to new material, and so on, is made on the basis of the formatted reply given by the student, and no opportunity exists for him to interject any qualifications of his own that were not explicitly anticipated by the programmer.
who originally formulated the course material as a programmed text. A future automated teaching system need not be conditioned by such constraints. In teaching an introduction to plane geometry for grade school children, for example, such a future automated teaching system might permit the child to draw his own geometric figures and make simple English statements about them in conjunction with some prompting from the program. Such a system would provide much greater latitude in exploring the ideal environment for individual learning.

Although many arguments to demonstrate the inseparability of natural language text and pictures could be marshalled, no important consequences would follow from this observation if it were not for the fact that, from the point of view of computer processing, these two kinds of information sources can be handled with the same techniques. We will have more to say about this in Chapter 3.
1.3 Related Systems.

Various approaches have been employed in inference making and natural language processing on a computer. An excellent review of techniques for syntactic analysis can be found in Bobrow [6]. Similarly, an extensive survey of question-answering systems can be found in Simmons [71]. A more recent and critical survey of data-retrieval systems, prepared by Kasher [32], points out some of the major weaknesses of previous work, even though it is perhaps unduly pessimistic about future prospects.

As indicated by Bobrow, there are several dimensions along which question-answering systems can be evaluated: syntactic, semantic, and deductive. Along the syntactic dimension one can measure the complexity of the format for acceptable input sentences; along the semantic dimension one can measure the extent to which relations among the defined primitives can be represented in the data structures of the system; along the deductive dimension one can measure the extent to which implicit relations among the inputs can be made explicit upon interrogation by the user. In addition, the ability of the system to interact with the user on a conversational basis and the growth potential of the system are important considerations. We will use these criteria in an evaluation of the work of Kirsch [36] and the work of Simmons and Londe [72], since both of these systems like GRANIS are natural language inference-making systems that operate on a graphic data base.
1.3.1 Graphic Data Base Systems.

1.3.1.1 Description of PLM.

In 1963 R. Kirsch, D. Cohen, B. Rankin, and W. Sillars [35] devised a collection of programs called Picture Language Machine (PLM) at the National Bureau of Standards which was designed to accept geometric pictures as input. PLM translates both the pictures and an English statement into a common intermediate logical language and determines whether the statement about the picture is true. PLM is composed of three subsystems--a parser, a formalizer, and a predicate evaluator. Its language is limited to a fragment of English suitable for making statements about three geometric figures. The parser is based on an unordered context free phrase structure grammar which includes a discontinuous constituent generator. Parsing is accomplished by means of a recognition routine which successively substitutes symbols in the dictionary for words in the sentence or for an intermediate symbol string until the top of the parsing tree is attained.

After the sentence has been parsed, the formalizer, developed by Sillars [69], translates the parsed sentence into the first order predicate calculus using a small number of constant predicates. The primitives of the language include brackets, parentheses, the terms "and", "if...then", "not", "identity", universal and existential quantifiers, variables, and three types of predicates. Typical one place predicates are
Cir(x)  x is a circle
Tri(x)  x is a triangle
Bk(x)  x is black

Examples of two and three place predicates are
Bgr(x,y)  x is bigger than y
Lf(x,y)  x is to the left of y
Smc(x,y)  x is the same color as y
Bet(x,y,z)  x is between y and z
Mort(x,y,z)  x is more to the right of y than z
Mmid(x,y,z)  x is more to the middle of y than z

The translation process involves substituting formalization symbols for grammar symbols beginning at the top of the parsed tree and working down. To each rule in the grammar there corresponds a rule in the formalizer. In the insertion of implication and quantifiers more than a simple substitution is required, but the result is always a unique, unambiguous, well formed formula. An example sentence

(3) Each triangle is to the left of a black circle.
translates into the well formed formula

(4) (∀x)[Tri(x) ⊃ (∃y){Bk(y) ∧ Cir(y) ∧ Lf(x,y)}].

The relationships between the geometric variables are well defined and the truth value may be tested by the predicate evaluator.

The predicate evaluator translates from pictures to the formal language. It is designed to accept inputs that have been processed by SADIE, a scanning device used as input to the computer. The inputs are restricted to three sizes of triangle, square, or circle, each of which may be black.
or white. A technique called blobbing is used to distinguish objects resulting from the scan and each such object is then circumscribed within a rectangle. Maximum and minimum x-y coordinates are computed and the ratios of these serve to distinguish triangles from circles or squares. Circles are distinguished from squares on the basis of covering less area. A black figure is one whose area is filled in, while a white figure is just an outline. These relatively simple computations serve to generate the valid predicates from the picture.

1.3.1.2 Limitations of PLM.

The syntax of PLM is quite restricted, thereby limiting the ability of the user to communicate comfortably with the system, unless he has first become quite familiar with the specification of the grammar. Because the grammar is restricted to a context free, phrase structure form, it can be extended only with considerable difficulty. Even if the grammar could be extended with comparative ease, side effects reflected in the semantics, particularly with respect to the scope of quantifiers and the binding of free variables, would continue to present serious problems. The semantics in PLM, however, is probably the more serious limitation. The superficial restrictions, such as limits on the number and range of predicates, are not as serious as the difficulty in initially fitting a wide variety of natural language statements into the predicate calculus format (for example, see Reichenbach [64], Chapter VII). Once the input text has been translated into the predicate calculus, the deductive capabilities of the system are quite powerful, as one might expect.
One of the chief drawbacks of the implementation is lack of conversational capability, preventing a dialog that can be used to help the user converge on a mutually acceptable input statement. In spite of these limitations, and the fact that it never really worked as an integrated operational system, PLM appears to be well conceived and was certainly the first system of this kind to be described in a formal manner. In addition, the possibility for converting to a transformational grammar and the extension to domains other than geometry were so inviting as to provide inspiration for some of the original work on GRANIS.

1.3.2.1 Description of NAMER.

In 1964 also inspired by PLM, Simmons and Londe at the System Development Corporation programmed a system called NAMER which could generate natural language sentences from line drawings displayed on a matrix. The primary intent of this research was to demonstrate that pattern recognition programs could be used to identify displayed figures and to identify the relations among them. After this had been established, a language generator was used to generate simple sentences such as "The square is above, to the right of, and larger than the circle." The sentences generated could be construed as answers to the various relational questions that might have been asked.

When a picture is presented on the input matrix, a set of 96 characteristics is computed. The algorithms or operators compute these as functions of the size, shape, and location of the pattern in the matrix. Typical characteristics that are derived include one bit indications of the presence or absence
of parts of the figure in sections of the matrix, of protuberances, of holes in the pattern (as in a circle), and of indentations as in a "u". A first level learning stage of NAMER selects a small subset of the 96 characteristics-- those which correlate most highly with correct recognition of the name by which the experimenter designates the pattern.

The second level of NAMER operates in a comparable fashion to obtain characteristics of the sets of coordinates representing two patterns. At this level the operators generate characteristics of comparative size, separation, density, height, etc. Subsets of these 96 characteristics are learned in the same fashion as at the earlier level to correlate with such relation terms as above, below, thicker than, to the right of, etc.

The language generator uses a very small phrase structure grammar to generate simple sentences which are true of the picture. Some sample sentences are

(5) The dog is beside and to the right of the boy.
(6) The circle is above the boy.
(7) The boy is to the left of and taller than the dog.

There is a great variety of drawings that can be learned and once a relation is learned between any two figures it usually generalizes successfully to most other pairs of figures.

1.3.2.2 Limitations of NAMER.

As Simmons admitted, NAMER is not strictly a question answerer. In order to answer English questions selectively it would be necessary to match the valid statements that can be
generated against an analysis of the specific question. Thus, it makes no sense to discuss the syntax of input statements. The generative grammar for the output sentences in NAMER seems to be tacked on as an afterthought, and it is characterized by a difficulty common to all generative grammars, viz-- the ability to generate syntactically well formed sentences which are semantically nonsensical. Thus, significant expansion of NAMER's grammar would probably require further direction from the semantics than is currently provided in order to ensure meaningful sentence generation. NAMER's semantics on the other hand is quite general, since the system is potentially capable of representing arbitrary spatial relationships among an arbitrary collection of primitive objects. This generality derives from NAMER's statistical pattern recognition techniques based on the work of Uhr and Vossler [81]. The inference-making capabilities of NAMER seem mainly inductive rather than deductive, and it is unlikely that deductive machinery could be introduced without extensive reprogramming. NAMER currently has no interactive potential, although it is not inconceivable that this could be added considering the highly conversational work of Simmons in connection with Protosynthex II [73].

1.3.3 Other Systems.

In addition to the two graphic data based systems described above, a number of other programs have been written which to varying degrees attempt to answer questions based on natural language inputs. We will briefly mention some of the more important ones and postpone any discussion until the next section.
For his doctoral dissertation at Carnegie Institute of Technology Robert Lindsay [43] wrote a program called SAD SAM (Syntactic Appraiser and Diagrammer - Semantic Analyzing Machine). The input to the system is a set of sentences taken from Basic English, a subset of English devised by Ogden [54], while the universe of discourse for SAD SAM is the class of family relationships. Bert Green, et al [29] constructed Baseball at the Lincoln Laboratory. Its data base was composed of the month, day, place, teams, and scores for each game in the American League for one year, and it also accepted a limited collection of questions from English. The largest example of a fact retrieval system is the work of Robert Simmons [70] at the System Development Corporation. The ProtosyntheX I data base is the sixteen volume Golden Book Encyclopaedia, and it answers questions as a function of the cooccurrence of content words in the input question with content words in the dictionary.

SIR (Semantic Information Retriever) is a system written by Bertram Raphael [63] in connection with his doctoral thesis at MIT. It is much more independent of a specific data base, but is still limited in the sentence formats that determine what storage and retrieval functions the system will perform. The relations that the system is capable of handling are set inclusion, set membership, equivalence, ownership, part-whole, and left-to-right position. Also for his doctoral dissertation at MIT Daniel Bobrow [7] programmed the STUDENT system which accepts algebra word problems phrased in English similar to that found in high school texts. These problems are translated into a set of equations which can then be evaluated arithmetically. Roger Elliot [21] for his dissertation at the University of Texas
programmed the GRAIS System which is an improvement over SIR in that the properties of relations such as transitivity, symmetry, and reflexivity can be input by the user. In addition the system has checks for the redundancy and consistency of the input data.

DEACON (Direct English Access and Control) is another large and continuing research project designed originally by Thompson [17] at the TEMPO Facility of the General Electric Company. The data base consists of some operational data on a portion of the Navy fleet such as ships, their movements and commanders, etc. It is of special interest because it demonstrates clearly the relation between syntax and semantics using syntax directed techniques, and we will have more to say about this later. Several programs for deductive question-answering have appeared based on the original work of McCarthy [47] with the "Advice Taker". Among them are Fisher Black's work [5] for his doctoral dissertation at Harvard and the work of James Slagle [76] on DEDUCOM (Deductive Communicator), but neither of these systems has been oriented to natural language input.

Kondô and Murata [37] at Tokyo Metropolitan University describe work toward a natural language inference-making system similar in spirit to the work of Kirsch, but not enough is known about the program to give any definite evaluation at this time. Cooper [16] and Darlington [18] both have programs which translate a subset of English into the propositional calculus. Cooper translates simple English sentences into Aristotelian logic by carefully restricting his grammar, whereas Darlington's program translates certain English riddles into logical form which may then be tested for validity. A program by Ross Quillian [61] in connection with his doctoral dissertation
at Carnegie Institute of Technology, although not strong in natural language input, does achieve some general inference-making capabilities by means of a careful structuring of his internal data representation.

1.3.4 Discussion.

In the preceding section a number of question-answering systems were mentioned quite briefly. Are there any limitations which all of them share in common? If we disregard for the moment those systems, such as Simmons'Protosynthex, which attempt merely to retrieve direct facts from the data base, and concentrate on those which have as a goal the answering of questions not explicitly stated in the input, then we observe that they all rely internally upon a formal logical language for obtaining inferences. Now if we further restrict ourselves to that subset which genuinely permits natural language inputs, we discover that the variety of natural language permitted is highly limited. That is to say, although a conversation with the system may "read" like English, even allowing for a restricted vocabulary, it does not "speak" or "write" like English. The reason for this is that in order to get from natural language input into a formal language suitable for inference-making, a translation is required, which as a prerequisite requires some kind of syntactic analysis of the input. In turn such an analysis requires a grammar, whether stated explicitly or implicitly, and getting a grammar sufficiently large to handle more than a mere fragment of natural language seems to be a major stumbling block. But are there any theoretical reasons why this should be the case, disregarding for the moment the practical problems of handling a large grammar on a computer? Certainly there is evidence to
indicate that large grammars can indeed be undertaken (for example, see the original work of Kuno [38] with the Harvard parser, Lehman and Pendegraft [40] at the University of Texas, the work of Watt [84, 85] at the National Bureau of Standards, Robinson [67] at RAND, the more recent work of Zwicky [91] with transformational grammars, and Clark [10] at IBM).

One theoretical problem which does arise with large grammars is the problem of ambiguity. This is evidenced by the large number of parsings produced by the Harvard Multiple Pass Syntactic Analyser on seemingly innocuous sentences. For example, the sentence

\[ (8) \text{A whip can be a stick with a cord or leather fixed to the end of it, used to give blows in driving animals, etc., or as punishment.} \]

produces over 120 parsings (see Quillian [61, p. 206). And this is not an uncommon occurrence, since similar sentences have been known to generate over one thousand parsings! The reason for these large numbers is due to the manner in which the individual ambiguities may interact; they combine in a multiplicative rather than an additive manner. The parsings are listed by the program in an essentially random order, while the particular parsing that a human would unhesitatingly choose as correct is in no way distinguished from the other irrelevant parsings. Occasionally, the natural parsing does not even appear on the list.

Thus before one can provide a genuine deductive, question-answering system with a wide range of natural language input, one must face the problem of ambiguity inherent in English. This entire problem is generally avoided in existing systems by tacitly considering only an unambiguous fragment of natural
language which in some sense is isomorphic with the logical language. The problem of translating a possibly ambiguous English sentence into the logical language which does not directly permit the representation of ambiguity is thereby circumvented. But how might one provide a question-answering system with a larger grammar? Before exploring this question and the problem of representing ambiguity further, we must examine more closely the relationship between syntax and semantics in natural language. This is the subject of the next chapter.
Chapter 2. AN INTEGRATED LINGUISTIC DESCRIPTION

The present chapter seeks to develop for natural language an integrated conception of the nature of linguistic description suitable for computer modeling. It builds on work accomplished in the area of syntactic analysis, and extends it to include recent work in semantic theory. According to this conception, a linguistic description consists of three components: syntactic, semantic, and pragmatic. The distinction between these three components was first made in the 1930's by Charles Morris [50] when he developed a science of signs which he called "semiotics" based on the earlier work of the American logician Charles Peirce. Morris defined these three terms as follows:

syntactics-- the study of the formal relation of signs to one another without regard to their specific significance;
semantics-- the study of the relation of signs to the objects to which these signs are applicable;
pragmatics-- the study of the origin, uses, and effects of signs within the behavior in which they occur.

To illustrate the distinction between these components consider the following three sentences:

(1) "It is cold" is a sentence.
(2) "It is cold" is true.
(3) I believe it is cold.

The first is a syntactic assertion; the second a semantic assertion; the third a pragmatic assertion. Indeed, if we
assume that it is cold, then by substituting the word "hot" for the word "cold" in each of the above sentences, the first remains true, the second becomes false, while the truth-value of the third cannot be determined.

Katz and Postal [34] have made an important theoretical contribution to our understanding of syntax and semantics. Martin [49] has developed an impressive formalization of pragmatics. In the remainder of this chapter we will examine how these three components relate in the development of a computer model.

2.1 The syntactic component.

The syntactic component of a linguistic description of a natural language must be a system of rules which describes the abstract structure underlying the sentences of a language. For the purpose of this thesis a transformational grammar in the sense described by Chomsky [11, 12, 13] will be assumed as the form of the syntax.

The initial part of such a grammar is a set of phrase structure rules. The form which such rules take is illustrated in the following over-simplified example:

1. By using a Backus-Naur Form (BNF) notation we have departed from the standard notation of an immediate constituent grammar generally used by linguists. This has been done for the sake of consistency with the later presentation and to emphasize the analytic rather than the generative aspects of the grammar.
The above grammar generates such sentences as

(4) John loves Mary.
(5) Mary loves the small white house.
(6) They love a big house.

Conversely, one may view the grammar as a recognizer in which sentences (4)-(6) would be said to be well-formed with respect to the grammar, whereas

(7) Mary loves the house.

is not well-formed, due to an idiosyncrasy of this particular grammar. Note that the third rule specifies that at least one adjective must precede the noun in any nounphrase with a determiner. Also observe that a recursive definition such as for adjective string in the fourth rule is permitted. Such a recursive construction provides for an arbitrary number of adjectives preceding a noun. The **structural diagram** (SD) of a sentence is a tree that exhibits the phrase structure of the sentence. For example, the SD for sentence (5) is as follows:
The metasyntactic classes enclosed in angular brackets, such as "sentence," "subject," "nounphrase," and so on are the non-terminal symbols of the grammar, whereas "John," "Mary," "house," "big," etc. are the terminal symbols associated with the grammar.

The second part of such a grammar is a set of transformations that may be used to convert one sentence in the language to another, where both sentences are related by a strong intuitive relationship. One such transformation is the transformation from active to passive voice. For example, "John loves Mary" into "Mary is loved by John." Another such transformation concerns the transformation from interrogative to declarative sentences. For example, "Does John love Mary?" into "John loves Mary." Still another important class of transformations concerns the mapping of compound sentences into their simpler kernel sentences and vica versa. For example, "John loves Mary and Mary loves the small white
house" into "John loves Mary" and "Mary loves the small white house." With the introduction of transformation rules, the basic phrase structure grammar can be simpler. Only a very simple set of "kernel sentences" of a language need be considered. All other complex sentences can be first decomposed into their kernel sentences, and then parsed with respect to the phrase structure grammar to obtain their structural diagrams.

2.2 The Semantic Component.

The semantic component of the linguistic description is taken to be a set of projective rules similar in spirit to that proposed by Katz and Fodor [33]. Such a projective device consists of two parts: first, a dictionary which provides a meaning for each of the terminal symbols in the language, and second, a set of semantic rules. The semantic rules assign an interpretation to the source statement in terms of the structural diagram produced by the syntactic component. This interpretation will be an expression in a formal language which corresponds to the meaning of the source statement.

There has been much discussion as to what kind of formal language, if any, should be used internally to represent the semantic information contained in the source text. If one's goal is merely to extract direct information from the given text, then tree or ring data structures may be adequate. If on the other hand, one expects the system to make deductions about the truth-value of complex propositions concerning varying domains, then a general mechanism for such deductions
must be independent of any particular domain and a formal logical language is suggested. During the time of the Greeks linguistics and logic were one general subject matter studied by the same philosophers. However, owing to certain historical developments, linguistics became divorced from logic, particularly in this country. The result of this separation has been extremely unfortunate. As Bar-Hillel [3] has pointed out, a separation of logic from linguistics is inherently wrong. Any attempt to understand even simple inferential mechanisms in natural language requires a recognition of the logical principles which underlie these mechanisms. Consider the following two examples: First, a linguist in attempting to describe the semantics of the phrase "smaller than"-- and countless other phrases in English-- must account in some way for the conditions that permit one to deduce that "A is smaller than C" given that "A is smaller than B" and "B is smaller than C." A dictionary with entries for "small" and "than" by itself is not sufficient for making such an inference. Some additional semantic rule based on the logic of "transitivity" is needed.

The second example derives from the relationship between the so-called "deep structure" and "surface structure" initiated by Chomsky [11]. As Bohnert [9] suggests, the two sentences

(8) Barking dogs don't bite.
(9) Parallel lines don't meet.

have the same surface syntax. Yet the transformation which changes (8) into

(10) No barking dog bites.

when applied to (9) yields
(11) No parallel line meets. which clearly is unintelligible. This is said to reveal a difference in deep structure in the given sentences (8) and (9). In order to describe precisely the difference between them, a systematic way of representing deep structures must be found. It is suggested that the notation of the predicate calculus in symbolic logic provides such a representation in a natural way. Sentence (8) is represented as 

\[(\forall x)(Dg(x) \land Bk(x) \supset \neg Bt(x)).\]

That is to say, for every x, if x is a dog and x is barking, then x does not bite. Sentence (9) is represented as 

\[(\forall x)(\forall y)(L(x) \land L(y) \land P(x,y) \supset \neg M(x,y))\]

or for every x and for every y, if x is a line and y is a line and x is parallel to y, then x does not meet y. This notation reveals among other things the different relational demands of "parallel" and "meets," and clearly illustrates the deep structure underlying each term.

For these reasons the predicate calculus has been selected as the formal language to represent the meaning of the source text. The dictionary now takes on the role of a collection of predicates to be used in the construction of expressions in the quantification calculus.
2.3 Relation between Syntax and Semantics.

Before continuing to describe the semantic component, we must examine further the relation between the syntactic and semantic components. First, let us return to the phrase structure grammar and take note of some of its deficiencies. Some of these deficiencies can be handled in the transformational part, but others cannot. Consider the following two examples. First, the phrase structure grammar could as easily generate the sentence

(14) They loves Mary.

as "They love Mary." Our intuition tells us that (14) is a grammatical deviation rather than a semantic one, and should be accounted for in the syntactic component. But such a simple rule as "the verb agrees with the subject in gender, person, and number" is not easily expressed in phrase structure grammar, and such agreement rules abound in natural language. If one were to approach this problem in the phrase structure grammar, then one would need a separate BNF rule for each agreeing sequence. This particular kind of difficulty, however, can be handled quite well in the transformational part of the grammar.

Now consider a second problem. We can also generate the sentence

(15) The big house loves John.

Now our intuition detects some semantic deviation. The sentence appears to be a reasonably well-formed English sentence. The difficulty is just that it doesn't make any
sense. One way of circumventing this problem within the grammar is to refine our notion of what it means to be a noun, i.e., by partitioning nouns into two categories: animate nouns and inanimate nouns. Then we may alter the BNF rule for <noun> to

\[
\text{<anoun> ::= John I Mary I } \cdots \\text{<inoun> ::= house I school I}\cdots
\]

making appropriate changes in the transformational part to ensure that love and its conjugations are in the class of verbs which permit only animate nouns as subjects. This solution is only temporary, however, since as soon as the grammar grows, a similar problem will arise between abstract and concrete nouns. This will require refining the concept of noun still further. Still later we will have to deal with such anomalies as "drink concrete" or "eat sincerity." As the grammar continues to grow, where this process will terminate is not clear. Of course, it must ultimately terminate when each lexical item in the general class of nouns forms a syntactic category by itself. But this solution violates our intuitive notion of what it means to be a syntactic category. The only distinction between such one-word categories is semantic and not syntactic. Thus, to approach basic semantic problems through the syntactic component leads to an explosion of the grammar, rendering it unmanageably complex.

On the other hand an attempt to approach essentially syntactic problems via the semantic component is equally doomed to failure. This mistake was first made, it appears, by the medieval grammarians who mechanically postulated a
class meaning for each grammatical category. In modern linguistics some grammarians have sought to find a semantic justification to explain why negated verbs in English require the auxiliary verb "do". Such an effort tends to confuse essentially meaningless grammatical transformation with the fundamentally meaningful semantic relations in the language. If these domains are not distinguished, no significant understanding of language can be achieved.

Yet as we observed while discussing ambiguity in the introduction, these two components are highly interactive. At least two kinds of ambiguity can be distinguished: lexical ambiguity and syntactic ambiguity. Lexical ambiguity arises when a particular word in a sentence can give rise to more than one interpretation, such as in

(16) He prefers to wear a light suit.

Here the lexical item light can be legitimately construed to mean either light in weight or light in color. In contrast

(17) He prefers to carry a light suit.

is a lexical ambiguity which can be resolved by the semantic component within the context of the sentence, because it is reasonable to assume that weight is highly correlated with carrying while color is not. Without lexical ambiguity there would be no possibility for puns or euphemisms. Syntactic ambiguity arises when a sentence has multiple parsings each of which can give rise to different interpretations, such as in the classic sentence

(18) They are flying planes.

Under one interpretation they are engaged in flying planes;
under another, one observes that those planes are planes which are flying. Another classic

(19) Look at the man on the hill with the telescope. illustrates a similar phenomenon. Under one interpretation we are exhorted to regard the man who carries a telescope; under another, to look with the aid of a telescope; under still another, our attention is drawn to the man on that hill distinguished by having a telescope mounted on it. Before we can hope to resolve such syntactic ambiguities with respect to a predetermined context, both the syntactic and semantic components must be required to interact with each other.

The syntactic ambiguity above illustrates that, even if the elements of a language description can be clearly partitioned into a terminal and a non-terminal vocabulary, there is no substance to the view that the terminal vocabulary alone is involved in semantic considerations. Such an assumption leads to hopelessly circular, inconclusive arguments as to the borderline between syntactical and semantical deviance. Though claiming the domains of syntax and semantics to be separate, linguists have repeatedly failed to locate any boundary between them. Furthermore, by insisting that semantics does not begin until syntax leaves off most grammarians have claimed too much for syntax. Indeed, one of the principal sources of difficulty for Katz and Postal is this same assumption that semantics begins where syntax ends.

The alternative approach to be presented in a later section makes no attempt to partition syntax and semantics
into mutually exclusive domains; on the contrary, we will argue for a deep association. To be sure, the basic parts of the syntactic and semantic components will remain recognizably distinct. At the same time, however, we will integrate the two through a sequential application of syntactic and semantic rules, and in particular, the generation of a semantic representation of the input sentence during the syntactic analysis.

To conclude this section, we have attempted to show
1) the principal domains of syntax and semantics must remain distinct;
2) an arbitrary delimitation—whether strongly in favor of syntax or semantics—leads to undesirable consequences;
3) in the general case syntax and semantics must interact in the analysis of sentences.

This interrelation between syntax and semantics is not intended as an apologetic compromise between contending proposals, but as a precise reflection of the facts of language.

2.4 The Pragmatic Component.

The role of the pragmatic component is to provide a variable, but well defined, universe of discourse to control the scope and content of the conversation with the system. The universe of discourse is taken to be a collection of objects, each of which is represented by a description list of attributes and values providing the information relevant to that object. We will show in the next chapter precisely how the system itself
may construct automatically such a collection of description lists from a pictorial input, but for the moment, the source of these lists need not concern us.

There are two reasons for restricting the domain of the conversation. First, the large base of knowledge about the world tacitly used by humans in their everyday inference making must remain extremely limited. This is essential in order to accommodate currently available computer memories. Second, if we are to avoid making spurious inferences, we must ensure a strict control over the meaning of the individual predicates recognized by the system. Those systems which do not appear to have such restrictions and claim to make non-trivial inferences about their subject matter are in reality mechanically generating relations without regard to the meaning of the individual lexical items being manipulated. For example, such a system given

(20) The bottle of ink is in the desk.

may legitimately infer that the ink is in the desk, taking advantage of the transitive properties of "of." Yet on the other hand given

(21) The King of France is in the castle.

the system could not avoid making the anomalous inference France is in the castle! This is because ink and France are merely place holders or dummies in the same dependency analysis. Of course, one could overcome this kind of fallacy with some postediting which takes into consideration the properties of ink and France. But this solution takes us back to description lists and a universe of discourse.

In addition, the universe of discourse provides a context within which the lexical and syntactic ambiguity of natural language input statements may be resolved.
Whenever an ambiguous statement is input at least two distinct well-formed expressions in the quantificational calculus will be constructed. Then by examination of the objects in the universe of discourse each expression will be evaluated, and the truth-value of each determined. If one of the evaluations is true, while the others are false, then we have a basis for resolving the ambiguity, assuming the speaker intended a true statement. Otherwise, the system must request clarification from the user.

In the event that the ambiguity is resolved, then we should not regard this ambiguity as a source of confusion and something to be avoided. On the contrary, in a sense it is desirable. It gives a new degree of freedom to the user in communicating with the computer by permitting him to concentrate on the problem at hand, rather than on its translation into a formal and unambiguous language. Resolvable ambiguity actually contributes to the expressive power of our language by permitting greater parsimony. What is lost in the input is recovered by relying on the inferential capabilities of the listener. Generally speaking, as long as the listener or the system comprehends the meaning intended, the speaker will seek as much freedom of expression as possible. Of course, inherent in such a scheme is the possibility of being misunderstood. But then, human beings occasionally are misunderstood. In fact, the possibility for misunderstanding increases as the common universe of discourse decreases. The language of treaties and legal documents is seemingly unnatural only because one cannot rely in some cases on the inferential capabilities of the parties involved to resolve ambiguities in the manner in which they were intended.
2.5 Productions.

As described, the syntactic component is not yet suitable for computer implementation. We have yet to obtain a recognizer from the syntactic component which provides convenient interaction with the semantic component within the framework of a computer model. A. Evans [23] and Feldman [25] have described production models for doing precisely this kind of task for compilers. Let us explore the possibility of using the already proven technique of syntax directed compiling in the processing of natural language. In our case, however, instead of using the source language to compile machine code which will be executed later on a particular machine, we will use natural language statements to construct well formed expressions in the predicate calculus which can be evaluated later with respect to a particular universe of discourse.

The essential ingredient of such recognition models is the production. Basic Floyd-Evans productions are labeled replacement rules for ordered pairs of strings similar to the primary statements in the COMIT and SNOBOL languages. Their crucial feature is the existence of an action field that permits a compile-time action to be associated with the execution of that rule. Generally the action field contains a call on a semantic routine that in turn calls on a collection of run-time routines which are pre-coded in the compiler and correspond to the primitive operations of the source language. In our application, which

1. Additional evidence for the value of this approach in natural language is provided by Thompson [79] in connection with his work on DEACON, even though DEACON does not use the predicate calculus as an internal representation.
is the translation of natural language statements into logical formulae, we propose to replace the action field of the production with an embedded production which we refer to as a semantic production. This innovation permits much greater flexibility in the construction of logical expressions, while still providing close interaction between the syntactic and semantic components.

The notion of using productions in semantic computations rather than in syntactic analysis or generation, however, is not a new concept in itself. Indeed, the notion of syntax directed computing is decendent from the canonical production systems of Post [57] which were not intentionally oriented to syntax analysis at all. Post [58] proved that any Turing machine calculation can be performed by a production system. The basic idea in the proof was to associate the infinite tape and the current state with the production stack. A one-to-one correspondence between the quintuplets of the Turing machine and the productions could then be established. Markov [48] and Davis [19] further extended the application of productions to logical systems. It was not until the work of Irons [31] and Floyd [26] that productions became an important tool in the parsing of compiler languages on a computer. Thus, as suggested by McCarthy [46], productions are both theoretically and practically a very powerful computing technique.

The translation of natural language statements using syntactic and semantic productions produces an expression in a logical language which is the interpretation of the source statement. Thus, this general method will be referred
to as syntax directed interpretation. This notion must be made more precise, however, if it is to be useful in the construction of a computer model. To this end we will attempt to formalize the ideas presented thus far within the framework of a natural inference system.

2.6 Natural Inference Systems.

In this section we will define a type of deductive question answering system based on natural language input called a Natural Inference System (NIS). It consists of a triple with the three components being essentially the syntactic, semantic, and pragmatic components discussed earlier. Our motivation for introducing this formal notion is so that we may prove important logical properties of these systems. One such property is completeness. Another is the complementary property of consistency. Both of these properties are shown to hold for natural inference systems.

2.6.1 Definitions.

A natural inference system is an ordered triple, \( \mathfrak{N} = (\mathfrak{S}, \Xi, \Omega) \) where \( \mathfrak{S} \) is a production system, \( \Xi \) is a dictionary, and \( \Omega \) is a universe of discourse.

2.6.1.1 A production system, \( \mathfrak{S} \) consists of two components: \( \mathfrak{S}_0 \) and \( \mathfrak{S}_\tau \) where \( \mathfrak{S}_0 \) is an analytic production system and \( \mathfrak{S}_\tau \) is a generative production system.

2.6.1.2 A string is a finite sequence of symbols, including the empty string, \( \Lambda \), which consists of no symbols. For any string \( s \), \( \Lambda s = s\Lambda = s \). A substring \( t \) of a string \( s \) (\( t \subseteq s \)) has

---

1. This is analogous to the notion of the environment, \( \mathfrak{T} \), in the development of Wirth and Weber [88].
the property that \( s = \gamma t_0 \) for strings \( \gamma, t_0 \) (possibly \( \gamma, t_0 = \Lambda \)).

A **proper substring** \( t \) of a string \( s \) \((t \subset s)\) is a substring such that \( t \neq s \neq \Lambda \).

2.6.1.3 A **vocabulary**, \( V = \{ a_i \mid a_i \text{ is a symbol } \wedge i = 1, 2, \ldots, n \} \).

2.6.1.4 A **production** is an ordered pair \((\varphi, \psi)\) of strings where \( \forall a \in \varphi \text{ and } \forall a \in \psi \ a \in V \). For some strings \( \chi_1 \text{ and } \chi_2 \) let

\[
\varphi = \chi_1 \omega \chi_2 \text{ and } \psi = \chi_1 \omega \chi_2 \quad \omega \neq \Lambda.
\]

A production is **context free** if \( \omega \in V \) and \( \chi_1 = \chi_2 = \Lambda \); otherwise **context sensitive**.

An **analytic production system** is an ordered pair, \( \mathcal{S}_\sigma = (V_\sigma, P_\sigma) \) where \( \sigma \), the syntactic stack, designates the stack associated with the analytic productions, \( P_\sigma = \{ (\varphi, \psi) \} \) is a finite set of productions, and \( V_\sigma = V_\sigma \cup V_\sigma \cup \mathcal{N}_\sigma \) where

\[
V_\sigma = \{ a \in V_\sigma \mid \forall (\varphi, \psi) \in P_\sigma \ a \notin \omega \subseteq \psi \} \text{ and } V_\sigma = V_\sigma - V_\sigma - \mathcal{N}_\sigma.
\]

\( \mathcal{N}_\sigma \) is called the **analytic terminal vocabulary** and \( V_\sigma \) is called the **analytic non-terminal vocabulary**.

A **generative production system** is an ordered pair, \( \mathcal{S}_\tau = (V_\tau, P_\tau) \) where \( \tau \), the semantic stack, designates the stack associated with the generative productions, \( P_\tau = \{ (\varphi, \psi) \} \) is a finite set of productions, and \( V_\tau = V_\tau \cup V_\tau \cup V_\tau \cup \mathcal{N}_\tau \) where

\[
V_\tau = \{ a \in V_\tau \mid \forall (\varphi, \psi) \in P_\tau \ a \notin \omega \subseteq \varphi \} \text{ and } V_\tau = V_\tau - V_\tau - \mathcal{N}_\tau - \mathcal{N}_\tau.
\]

\( \mathcal{N}_\tau \) is called the **generative terminal vocabulary** and \( V_\tau \) is called the **generative non-terminal vocabulary**.
2.6.1.5 Let \( P = \{(\varphi, \psi)\} \) where \( \varphi = x_1 \omega x_2 \) and \( \psi = x_1 \omega x_2 \).

A loop \( L \) is a subset of \( P, L = \{(\lambda_1, \mu_1)\} \), such that \( \lambda_1 = x_1 \omega_1 x_2 \) and \( \mu_1 = x_1 \omega_1 x_2 \), with \( \omega_1 \neq \Lambda \) for which the strings \( \xi_i \subseteq \omega_i \) and \( \eta_i \subseteq \omega_i \) satisfy

\[
\eta_i = \xi_{i+1} \quad \text{and} \quad \eta_n = \xi_1 \quad (1 \leq i \leq n).
\]

A set of productions is said to be loop-free if it contains no occurrence of a loop.

2.6.1.6 A production system is said to be monogenic if

\( \forall (\xi, \tau) \in P \) and \( \forall (\lambda, \mu) \in P, \xi = \lambda \) implies \( \tau = \mu \).

EXAMPLE: A typical labeled production in Formula Algol has the following format:

\[
\alpha: \text{IF } \forall = [c_1, c_2, \ldots, c_n] \text{ THEN BEGIN ALTER } \nu \text{ TO } [d_1, d_2, \ldots, d_n]; \quad \text{GO TO } \tau \text{ END};
\]

where \( \alpha \) and \( \tau \) are labels, \( c_i, d_j \in V \), \( i = 1, 2, \ldots, n \); \( j = 1, 2, \ldots, m \) are constituents such that \( [c_1, c_2, \ldots, c_n] = \varphi \) and \( [d_1, d_2, \ldots, d_n] = \psi \), and the double equal sign is a Formula Algol operator which determines if the object on the left hand side is an instance of the pattern on the right hand side.

2.6.1.7 A dictionary, \( \mathcal{D} = \{f_i(x_1, x_2, \ldots, x_n) \in V \mid \text{the } f_i \text{ are } m \text{-adic predicates (} m \geq 1 \text{) } i = 1, 2, \ldots, n \leq \infty \text{ and the } x_i, j = 1, 2, \ldots, m \leq \infty \text{ are free argument variables}\} \).

EXAMPLE: 1. TRIANGLE(X)
2. BLACK(X)
3. GREATER(X, \psi)
2.6.1.8 A **universe of discourse**, \( \Omega = \{ q_k \mid q_k \rightarrow [A_1 : V_{11}, V_{12}, \ldots, V_{1n_1}] \}
\begin{align*}
&[A_2 : V_{21}, V_{22}, \ldots, V_{2n_2}] \cdots [A_m : V_{m1}, V_{m2}, \ldots, V_{mn_m}],
\end{align*}
where the \( V_{ij} (i=1,2,\ldots,m \text{ and } j=1,2,\ldots,n_i) \) are values on the **description list** of the object \( q_k \) corresponding to the prescribed attributes, \( A_i (i=1,2,\ldots,m) \).

**EXAMPLE:**

1. **OBJECT[3] =**
\[
\begin{align*}
\text{TYPE: POLYGON, TRIANGLE} & \\
\text{SIZE: 10000} & \\
\text{COLOR: BLACK} & .
\end{align*}
\]

\[
\begin{align*}
\text{TYPE: POLYGON, SQUARE} & \\
\text{SIZE: 5000} & \\
\text{COLOR: WHITE} & .
\end{align*}
\]

2.6.1.9 Let the production system \( \Sigma = (V,P) \) be given. We may write \( \varphi \rightarrow \psi \) when \( (\varphi, \psi) \in P \). For each string \( v = v_a \varphi v_b \) (\( v_a, v_b \) possibly \( \lambda \)), \( v^1 = v_a \varphi v_b \) is called a **first transform** of \( v \), and is said to result from the **application** of \( (\varphi, \psi) \) to \( v \). Let \( v = v^1 \) denote this relation. A string \( v^k \), such that \( v = v^1 = \cdots = v^{k-1} \rightarrow v^k \) is called the **kth transform** of \( v \). Let \( v = v^k \) denote this relation. For analytic production systems the sequence \( v, v^1, v^2, \cdots \) where \( v^k \rightarrow v^{k+1} \) is called a **parse** of \( v \). For generative production systems this sequence is called the ** derivation** of \( v \). Note that for a given \( v \), there may be none, one, or many first transforms. If \( s \Rightarrow t \), we say that \( s \) **produces** \( t \), and conversely \( t \) is the **reduction** of \( s \).

2.6.1.10 A **sequencing function** is a method of passing control from one production to the next in an ordered sequence of productions. In a production \( \varphi \rightarrow \psi \), the elements of \( \varphi \) are said to form a **pattern**. These elements are also called **constituents**. A **match** is an attempt to recognize by means of the pattern \( \varphi \) an exact instance of a configuration of symbols in the stack.
associated with the production. By convention, the elements of the stack are scanned from right to left. If the contents of the stack, \( v \), is decomposable into \( v_a \varphi v_b \) in several ways, then we specify success only for that match which minimizes the length of \( \varphi v_b \) while maximizing the length of \( \varphi \). Thus, a match will be either uniquely successful or unsuccessful. If a match in a production is successful, then a serial sequencing function passes control to the labeled production named in the "go to" statement of that production; otherwise it passes control to the next production in the sequence. Using this definition, a Markov Algorithm (cf. Markov[48]) is a set of productions distinguished by a single entry point at the first production, a serial sequencing function, and the condition that all "go to" statements (which are implied in the set) refer to the entry point. A cluster of productions is a set of productions distinguished by one entry point, and an error production particular to that set indicating that a certain construction was anticipated in the parse and not found.

2.6.11 An analytic grammar, \( G = (\Sigma, S) \) where \( P = \{(\varphi, \psi)\} \) is a set of productions and \( S \in V \) is a distinguished symbol called the sentence symbol. The last production to be executed in an analytic grammar is called a success production and must contain an instance of \( S \).

A generative grammar, \( G = (\Sigma, T) \) where \( P = \{(\varphi, \psi)\} \) is a set of productions and \( T \in V \) is a distinguished symbol called the root symbol. The first production to be executed in a generative grammar must contain an instance of \( T \).
2.6.1.12 An analytic language, \( L = L(G) = \{ s = s_1 s_2 \cdots s_n \mid s_i \in V_{\Sigma} \text{ for } i = 1, 2, \cdots, n < \infty \text{ and } s \in S \text{ in a finite number of transforms} \} \).

A generative language, \( L = L(G) = \{ t = t_1 t_2 \cdots t_n \mid t_i \in V_T \text{ for } i = 1, 2, \cdots, n < \infty \text{ and } t \in T \text{ in a finite number of transforms} \} \).

2.6.1.13 \( G_1 \) and \( G_2 \) are said to be (weakly) equivalent if \( L(G_1) = L(G_2) \).

2.6.1.14 For a given analytic grammar a string \( v \) is said to be ambiguously derivable if there exist two parses of \( v \) to \( S \) whose associated phrase markers (cf. Chomsky [11, p. 65]) are different. For a given generative grammar a string \( v \) is said to be ambiguously derivable if there exist two derivations of \( v \) from \( S \) whose associated phrase-markers are different. A grammar is said to be unambiguous if \( \forall v \in L(G) \) \( v \) is not ambiguously derivable.

2.6.1.15 The generative grammar \( G_T = (\Sigma_T, T) \) will now be defined more explicitly. The semantic terminal vocabulary \( V_T \) consists of

(i) a finite set of \( n \)-adic predicates \( \{ F_i \} \) \( i = 1, 2, \cdots, m < \infty \) whose members comprise \( \Sigma \)

(ii) a finite set of argument variables \( \{ x_i \} \) \( i = 1, 2, \cdots, m < \infty \)

(iii) the logical operator "\( \& \)"

(iv) the logical connectives "\( \lor \)", "\( \land \)”, "\( \neg \)”, "\( \equiv \)"

(v) the comma ",”

(vi) the two parentheses "(" and ")"

(vii) the universal and existential quantifiers "\( \forall \)" and "\( \exists \)".

The semantic non-terminal vocabulary \( V_T \) consists of a finite set of \( n \)-adic predicates \( \{ F_i^T \} \) \( i = 1, 2, \cdots, N < \infty \).
A well formed formula (wff) in the quantificational calculus is defined recursively as follows:

(i) "F(x_1, x_2, \ldots, x_n)" is a wff where F \in V_T is an n-adic predicate (n \geq 1) and x_1, x_2, \ldots, x_n are argument variables;

(ii) If \varphi is a wff then "(\forall x)\varphi" and "(\exists x)\varphi" are also wff where x is an argument variable;

(iii) If \varphi is a wff then "-\varphi" is a wff;

(iv) If \varphi and \psi are wff then "(\varphi \lor \psi)", "(\varphi \land \psi)", "(\varphi \supset \psi)", and "(\varphi \equiv \psi)" are also wff.

G_T is said to be well formed in the QC if L(G_T) \subseteq \{wff\}.

2.1.6.16 To define the relation between \mathcal{D}_\sigma and \mathcal{D}_\tau in \mathcal{D}, clusters of productions from \mathcal{D}_\sigma and \mathcal{D}_\tau will be interleaved in the general case, although \mathcal{D} as a whole will be constrained to a serial sequencing function. Thus, associated with each parse in the analytic grammar is a derivation in the generative grammar. In this fashion the analytic productions guide the generation of expressions in the generative language. Even though productions from the analytic and generative grammar are interleaved, the identity of any particular production is always clear, since each production makes explicit reference to either the syntactic stack or the semantic stack. Depending on how the analytic and generative productions are interleaved, each sentence \sigma \in L(G_\sigma) will produce an expression \tau \in L(G_T) called the interpretation of \sigma. We will write \mathcal{D}(\sigma) to denote the interpretation of \sigma.

2.6.1.17 We shall say that a sentence \sigma \in L(G_\sigma) is valid if and only if all instances of the interpretation of \sigma, \mathcal{D}(\sigma), are true in the universe of discourse, \Omega. For any \Omega there
exists a mechanical procedure for determining whether \( s \in L(G) \) is valid. Let \( t = \mathcal{I}(s) \) be the interpretation of \( s \). The evaluation \( \mathcal{I} \) of \( t \) produces a truth-value in the context of the universe of discourse \( \Omega \). \( \mathcal{I}_\Omega(t) \) consists of two subprocesses:

1. Apply each of the following productions to \( t \) until it is no longer applicable:
   
   (i) \( t_1 = t_2 \rightarrow (t_1 \supset t_2) \land (t_2 \supset t_1) \)
   
   (ii) \( t_1 \supset t_2 \rightarrow (\neg t_1 \lor t_2) \)
   
   (iii) \( \neg \neg t \rightarrow t \)
   
   (iv) \( (\forall x)t(x) \rightarrow t(o_1) \land t(o_2) \land \cdots \land t(o_n) \)
   
   (v) \( (\exists x)t(x) \rightarrow t(o_1) \lor t(o_2) \lor \cdots \lor t(o_n) \)

   where \( n \) is the number of objects in \( \Omega \). These productions are said to be elimination productions, since they each eliminate one terminal symbol from the expression to which they are applicable. Let the resulting schema be \( t' \).

2. Subject \( t' \) to the truth table test.

Thus, \( s \) is valid if and only if \( t' \) is a tautology.

2.6.18 The negation of a sentence \( s \), \( \neg s \), is the addition of the prefix phrase "It is not the case that." In order to operate on the negation of sentences in \( L(G) \), we require that \( \exists \overline{p}_\sigma \in P_\sigma \) and \( \overline{p}_\tau \in P_\tau \) such that

\[
\overline{p}_\sigma: \quad [\text{It}, \text{is}, \text{not}, \text{the}, \text{case}, \text{that}, \psi] \to \psi \quad \text{go to} \quad \overline{p}_\tau;
\]

\[
\overline{p}_\tau: \quad \varphi \to \neg \varphi
\]

2.6.19 A production system \( \mathcal{I} \) is said to be complete if \( \forall s \in L(G) \), either \( s \) or the negation of \( s \) is valid. A production system is said to be consistent if \( \forall s \in L(G) \), it is not the case that both \( s \) and the negation of \( s \) are valid.

---

2.6.2 Properties of Natural Inference Systems.

**Lemma 1.** Let $G = (Q_\sigma, S)$ and $G = (Q_\tau, T)$ be given. If $P_\sigma$ and $P_\tau$ are loop free, then $\forall s \in L(G_\sigma) \exists (s)$ exists.

**Proof.** If no subset of $P$ is a loop, then for $i \neq j$ we must have $s^i \neq s^j$ and $s^i \neq s^n = S$ for $n < \infty$. Similarly, for $i \neq j$ we must have $t^i \neq t^j$ and $T^i \neq t^m = T \in L(G_\tau)$ for $m < \infty$. Thus, in the application of $\exists$ to $s$ we obtain an expression $t \in L(G_\tau)$ in a finite number of steps.

There are two principal sources of syntactic ambiguity. One arises if the sequencing function is not unique; another arises if some cluster of productions is not unique. In the latter case the first production is said to *preclude* the second, and the other possible parsing will not be found.

**Lemma 2.** Let $G = (Q_\sigma, S)$ and $G = (Q_\tau, T)$ be given. If each cluster of $Q_\sigma$ and $Q_\tau$ is monogenic and $\exists$ is endowed with a serial sequencing function, then $\exists (s)$ is unique.

**Proof.** Because we have a serial sequencing function, the next production to be executed in the parsing and derivation is unique. If each cluster of $Q_\sigma$ is monogenic, then there does not exist both $(\psi, \psi_1) \in P_\sigma$ and $(\phi, \phi_2) \in P_\sigma$ in the same cluster, $\psi_1 \neq \psi_2$; hence, in $s = s^1$, $s^1$ is unique. But if $s^{k-1}$ is unique, then $s^k$ is unique, and by induction on $k$, $s$ has a unique parse $s \models S$. Similarly, if each cluster in $Q_\tau$ is monogenic then there does not exist both $(\phi, \phi_1) \in P_\tau$ and $(\psi, \psi_2) \in P_\tau$ in the same cluster, $\psi_1 \neq \psi_2$; hence in $T = t^1$, $t^1$ is unique. But if $t^{k-1}$ is unique, then $t^k$ is unique, and by induction on $k$, $t$ has a unique derivation $T \models T$ as a function of $s$. Thus, $\exists (s) = t$ is a well defined mapping.
Although the serial sequencing function is given for \( \Sigma \), the sequencing through the productions of \( \Sigma _ { T } \) will in general be a function of \( s \). Thus the derivation for any particular \( t \in L( G _ { \Sigma } ) \) need not be unique, i.e., \( \phi ( s ) \) is not in general a one-to-one function. The set \( PP( t ) = \{ s \in L( G _ { \Sigma } ) \mid \phi ( s ) = t \} \) is called a paraphrastic set and consists of all the members of \( L( G _ { \Sigma } ) \) which give rise to the same interpretation.

For the moment we will restrict our attention to the case where \( \phi \) has a serial sequencing function and the production systems \( \Sigma _ { \Sigma } \) and \( \Sigma _ { T } \) are loop-free and have monogenic clusters.

**Lemma 3.** Let \( G _ { \Sigma } = ( \Sigma _ { \Sigma } , S ) \) and \( G _ { \Sigma } = ( \Sigma _ { \Sigma } , T ) \) be given. If \( G _ { \Sigma } \) is well formed in the QC then \( \phi ( s ) \in \{ \text{wff} \} \).

**Proof.** By Lemmas 1 and 2 \( \phi ( s ) \) is a well defined function. \( \phi ( s ) = t \in L( G _ { \Sigma } ) \). But, since \( G _ { T } \) is well formed in the QC, then \( t \in \{ \text{wff} \} \) and thus \( \phi ( s ) \in \{ \text{wff} \} \).

We will now turn our attention to \( G _ { T } \) well formed in the QC.

**Lemma 4.** \( \bigcup _ { \Omega } ( \phi ( s ) ) \in \{ 0, 1 \} \).

**Proof.** If we associate 0 with false and 1 with true, then it is sufficient to show that for any \( t \in \{ \text{wff} \} \) the elimination algorithm in \( \bigcup _ { \Omega } \) terminates, since once we have a quantifier free form of \( t \) with no occurrences of "\( = \)", "\( \neg \)", and bound variables, the application of truth tables for the logical connectives "\( \land \)", "\( \lor \)", and "\( \neg \)" to yield a single truth-value is an easy matter.
Each of the five elimination productions has the form

\[ \psi_1 \alpha \psi_2 \rightarrow \beta \]

where \( \alpha \not\in \beta \)

and \( \alpha \) takes as value "\( \& \)", "\( \neg \)", "\( \rightarrow \)" "\( \forall \)" and "\( \exists \)" respectively. Initially \( t \) contains a finite number of symbols, thus it contains finitely many occurrences of \( \alpha \). Since \( \alpha \not\in \beta \), the number of occurrences of \( \alpha \) cannot increase on application of the production. Label each occurrence of \( \alpha \) in \( t \). Thus we have \( \alpha_i \), \( i=1,2,\ldots,n \). By \( n \) applications of the production all occurrences of \( \alpha \) in \( t \) are eliminated, the production is no longer applicable, and control passes to the next production. Since no succeeding production affects the applicability of a preceding production, control will ultimately pass through the fifth production and the algorithm will terminate. The proof depends in an essential way on the finiteness of the universe of discourse, \( \Omega \), and it was because of this that it was unnecessary to obtain the Prenex and Skolem Normal Forms.

**Theorem 1.** \( \mathfrak{S} \) is complete.

**Proof.** From the definition of completeness and lemma 4 we must show \( \forall s \in L(\mathcal{G}_\varphi) \) that if \( \mathcal{U}_n(\mathfrak{S}(s)) = 0 \) then \( \mathcal{U}_n(\mathfrak{S}(\varphi)) = 1 \). It is sufficient to show \( \neg \mathfrak{S}(s) = \mathfrak{S}(\overline{s}) \), since \( \forall t \in \{ \text{wff} \} \mathcal{U}(t) = 0 \) implies \( \mathfrak{U}(t) = 1 \) implies \( \mathfrak{U}(-t) = 1 \). But the negation transforms in \( \mathfrak{S} \) ensure that \( \mathfrak{S}(\overline{s}) = \neg \mathfrak{S}(s) \).

**Theorem 2.** \( \mathfrak{S} \) is consistent.

**Proof.** From the definition of consistency and lemma 4 we must show \( \forall s \in L(\mathcal{G}_\varphi) \) that if \( \mathcal{U}_n(\mathfrak{S}(s)) = 1 \) then \( \mathcal{U}_n(\mathfrak{S}(\overline{s})) = 0 \). Again it is sufficient to show that \( \mathfrak{S}(s) = \neg \mathfrak{S}(\overline{s}) \), since \( \forall t \in \{ \text{wff} \} \mathcal{U}(t) = 1 \) implies \( \mathfrak{U}(t) = 0 \) implies \( \mathfrak{U}(-t) = 0 \). But again this is assured by the definition of the negation of \( s \).
2.6.3 Ambiguous Grammars.

In the event that all clusters are not monogenic, i.e., the recognizer accepts syntactically ambiguous sentences, then the sequencing function should be modified so that a distinct interpretation is constructed for each legitimate parse of the input sentence. In such a case $\mathcal{S}(s)$ is no longer a function in the strict sense, but a one-to-many mapping of the input sentence onto a collection of its different interpretations. Thus, the completeness and consistency of $\mathcal{S}$ no longer hold, since both preceding theorems depend in an essential way on the assumption of a unique interpretation for each input. It is not even clear how to assign a single truth-value to an ambiguous sentence. Although we will not adopt this approach, one method of attack is to use an infinitely many-valued logic. By defining the truth-value $\omega$ of a sentence $s$ as

$$\omega(s) = \frac{\text{The number of true interpretations of } s}{\text{The total number of interpretations of } s}$$

we have a method for identifying the truth-value of all sentences. We observe that $\omega \in [0,1]$ and for $\omega = 1$ we say that $s$ is true, while $\omega = 0$ implies that $s$ is false. $\omega = 1/2$ implies that $s$ is ambivalent; $\omega > 1/2$ that $s$ is more true than false; and $\omega < 1/2$ that $s$ is more false than true. $\omega(s_1) > \omega(s_2)$ suggests that $s_1$ is more true than $s_2$, and so on. Note that for the degenerate case of unique interpretations, the definition of $\omega$ reduces correctly. In this fashion one might generalize the notion of completeness to
say that \( \phi \) is complete if \( s \) or its negation has an \( w \geq 1/2 \). Similarly, consistency might be generalized as it is not the case that both \( s \) and its negation have an \( w > 1/2 \). Then, with a few additional changes, the theorems would again hold under these generalized definitions.

The difficulty with this approach is that it may violate our intuitive notion of truth and falsity for ambiguous sentences. In the vast majority of cases the speaker intends only one interpretation of his sentence, and frequently is oblivious to any others unless they are explicitly pointed out. Only in the realm of humor does the speaker systematically intend that the listener contrast the varying interpretations of his statement. Any proposal to determine the truth-value of a source statement by weighing the truth-values of the various interpretations and consolidating them all into one global truth-value would seem ludicrous to a serious speaker. But yet, because so much of our language is ambiguous, people must use some method of resolving ambiguity. Otherwise there would be much greater evidence of confusion and misunderstanding than is actually observed. There are at least two reasons why greater misunderstanding does not occur among humans.

First, people tend to order the possible interpretations of a syntactically ambiguous sentence in a natural way. A typical source of syntactic ambiguity is the potentially multiple dependencies that may arise between prepositional phrases or relative phrases and that which they modify. Thus, a heuristic which most people observe in finding the
preferred interpretation is to let the dependent word or phrase modify its nearest preceding referent. If this first interpretation makes sense, i.e., can be evaluated with respect to the universe of discourse, then the listener will assume that the speaker intended this natural interpretation and immediately proceed to the next statement.

A second reason why there is not more misunderstanding is that unintentional interpretations of ambiguous sentences are generally not false with respect to the universe of discourse, but vacuous. This provides the listener with another heuristic for making an inference about the likelihood of possible interpretations, since we assume that the serious speaker will tend to form statements that are either true or false, but never vacuous. Both of these heuristics— in direct opposition to the truth-functional computation described above—will be adopted for natural inference systems.

Therefore, our strategy for resolving ambiguity is to evaluate the most natural interpretation first. Only in the case where this natural interpretation is vacuous do we evaluate the next most likely interpretation, and so on until a non-vacuous interpretation is found. And only if each logically possible interpretation is neither true nor false with respect to the universe of discourse do we declare the sentence vacuous. The principal advantage of this strategy is that it minimizes processing time for ambiguous sentences, avoiding a complete evaluation of spurious interpretations.
The ordering of possible interpretations, rather than evaluating all logically possible interpretations immediately, eliminates the combinatorial interaction of ambiguities which, although harmless for simple sentences, can be severe for complex sentences. The main disadvantage of this approach, however, is that it will not explicitly test for genuine ambiguity so as to alert the speaker to possibly undesirable consequences. But this is a problem for humans as well.\footnote{In a recent psychological experiment in which humans were presented with the task of completing sentence fragments, MacKay \cite{45} observed that even though subjects consistently took longer to complete ambiguous sentences than unambiguous ones, none of them reported being aware of the ambiguities while completing the sentences.}
Chapter 3. GRANIS, A COMPUTER MODEL

A crucial test of any formal mathematical system, such as that outlined in section 2.6, is whether it can be used to guide the design of a running computer program. The importance of such a program is that it immediately provides a demonstration of the feasibility of the ideas upon which it is based. In this chapter we will see how a natural inference system, such as that presented in the preceding chapter, can be implemented.

Formula Algol [56] was selected as the programming language for the implementation. This choice was made because Formula Algol incorporates within the framework of one language a wide variety of formula manipulation and list processing capabilities. Both of these capabilities are essential in our application. Formula Algol extends ALGOL 60 by adding two new types of data structures together with appropriate primitive processes to operate on them. However, the control structure of ALGOL 60 is inherited without change. The two new declaration types are type "FORMULA" and type "SYMBOL."

Because Formula Algol is a compiler rather than an interpretive language it has the advantage of producing highly efficient machine code for later execution. The importance of efficient execution becomes apparent when one attempts to operate with the system in an interactive mode. Experience has shown that if the system takes longer than
thirty seconds to parse, interpret, and evaluate a simple English sentence, then conversation rapidly becomes degraded. The reason is that human attention span is exceeded, and conversation correspondingly degenerates. On any particular machine an interpretive programming language of necessity would only make this problem more severe. An undesirable consequence of the fact that Formula Algol is a compiler, however, is the high comparative cost of making even simple structural changes to the program. Recompiling may take as much as four minutes of computer time.

Another important advantage of Formula Algol is its easy readability. Its transparency contrasts sharply with most currently available programming languages. An original design requirement of ALGOL 60 was that it be a language suitable for the communication of algorithms within the programming community, and Formula Algol has inherited the relatively perspicuous notation of its predecessor.

3.1 Program Structure.

The three major components of GRANIS—syntactic, semantic, and pragmatic—are readily distinguished in the program. The syntactic component comprises the bulk of the program and uniformly consists of clusters of formula productions for the recognition and translation of well formed English sentences.
3.1.1 The Production System.

Given the initial phrase structure grammar, the method used to construct the production recognizer is based on the Earley Algorithm [20]. The algorithm produces a highly efficient one pass, one-push-down-stack, bounded context recognizer which involves no closed subroutines. Two kinds of production clusters called type 0 and type 1 clusters are constructed corresponding to each character in the phrase structure grammar. A cluster acts as a unit because only its first production has a label to which transfer can be made. The type 0 cluster for a character A (labeled A0) is constructed so as to appear at the point in the parse where an instance of A is expected in the sentence starting with the last symbol scanned. These are constructed only for non-terminals. The type 1 cluster for a character A (labeled A1) is constructed so as to appear at the point in the parse where A is second in the syntax stack, and a decision on the next cluster to be transferred to must be made as a function of context. There are three cases in which type 0 productions are constructed and five cases in which type 1 productions are constructed. These are illustrated in Table 1. A brief outline of the algorithm will be given.

**Type 0 Productions.** Let A be the non-terminal for which we are constructing type 0 productions. Further, let T(A) be the set of all terminal characters with which A can begin. We will construct a production corresponding to each member of T(A). For each alternative in the definition of A, if it begins with a terminal, a production is constructed from that string. For terminals T(a) is defined as T(a) = a.
BNF Rules to Formula Productions

**TYPE 0 PRODUCTIONS**

**Context 1.** \( G ::= a \) | ...  
IF \( S = [\$, a] \) THEN BEGIN ALTER LAST OF \( S \) TO \([G, \text{NEXT}]\); GO TO \( G \) END;

**Context 2.** \( G ::= ab \ldots \) | ...  
IF \( S = [\$, a] \) THEN BEGIN INSERT \([\text{NEXT}]\) AFTER LAST OF \( S \); GO TO \( a \) END;

**Context 3.** \( G ::= aH \ldots \) | ...  
IF \( S = [\$, a] \) THEN BEGIN INSERT \([\text{NEXT}]\) AFTER LAST OF \( S \); GO TO \( H0 \) END;

**TYPE 1 PRODUCTIONS**

**Context 1.** \( G ::= \alpha X \) | ...  
IF \( S = [\$, \alpha, X, \text{SIGMA}; \$] \) THEN BEGIN ALTER LAST 3 OF \( S \) TO \([G, \text{SIGMA}]\); GO TO \( G \) END;

**Context 2.** \( G ::= \alpha XH \) | ...  
IF \( S = [\$, \alpha, X, i] \) THEN GO TO \( H0 \);

**Context 3.** \( G ::= \alpha Xc \) | ...  
IF \( S = [\$, \alpha, X, c] \) THEN BEGIN ALTER LAST 3 OF \( S \) TO \([G, \text{NEXT}]\); GO TO \( G \) END;

**Context 4.** \( G ::= \alpha Xcd \) | ...  
IF \( S = [\$, \alpha, X, c] \) THEN BEGIN INSERT \([\text{NEXT}]\) AFTER LAST OF \( S \); GO TO \( c \) END;

**Context 5.** \( G ::= \alpha XcH \) | ...  
IF \( S = [\$, \alpha, X, c] \) THEN BEGIN INSERT \([\text{NEXT}]\) AFTER LAST OF \( S \); GO TO \( H0 \) END;

Notes:
1. As in COMIT, \$ represents an arbitrary number of constituents.
2. The "SIGMA" is a Formula Algol extractor.
3. "NEXT" is a parameterless procedure which places the next character from the input string into the stack.

**TABLE 1.** BNF RULES TO FORMULA PRODUCTIONS
If it begins with a non-terminal B, we determine \( T(B) \) from the definition of B and proceed as before with \( T(A) \) until we have only alternatives which begin with terminals. The production constructed depends on the context of the initial terminal. The three possible contexts are shown in Table 1. If two of the constructed productions have the same stack element then they should be replaced by one of the form of context 2, since any parsing decision must be postponed until another character has been scanned.

**Type 1 Productions.** Let \( A \) be the character for which we are constructing type 1 productions, and let \( P(A) \) be the set of all places in the BNF in which \( A \) appears. A type 1 production should be constructed for each member of \( P(A) \) according to the five cases illustrated in Table 1.

Unlike the type 0 productions, the order of the type 1 productions is important and care should be taken to ensure that no production precludes any other. In the exceptional case where two productions constructed in this fashion mutually preclude each other, we have what is called a "culprit". To delete such a culprit, the grammar must be augmented to increase the context of one of the two culprit alternatives. Since this can always be done without loss of generality, there is no theoretical problem. In any event for typical natural language grammars such culprit situations are extraordinarily rare. The introduction of metasyntactic class definitions is helpful in the consolidation of similar type 1 productions, and can realize a significant saving in the total number of productions. Since all of the resulting production
clusters are closed with respect to the rest of the productions, their order in the total set of clusters is irrelevant.

To better understand how to construct a recognizer using this algorithm, the BNF grammar given in Chapter 2 will be taken as an example. When applied to the given grammar of 10 BNF rules, the algorithm produces 46 productions in 15 separate clusters. By defining the following terminal meta-classes:

- **PROCLASS** = \{he, she, they\}
- **NOUNCLASS** = \{John, Mary, house\}
- **ADJCLASS** = \{big, small, black, white\}
- **VERBCLASS** = \{love, loves\}
- **DETCLASS** = \{the, a\}

and by appropriate relabeling, we may delete the redundant clusters for NOUN0 and DET0 to obtain the following 20 formula productions:

```plaintext
SENT0: IF S==[$,PROCLASS] THEN BEGIN ALTER LAST OF S TO [PRO,NEXT]; GO TO PRO1 END;
IF S==[$,DETCCLASS] THEN BEGIN ALTER LAST OF S TO [DETC,NEXT]; GO TO DET1 END;
IF S==[$,NOUNCLASS] THEN BEGIN ALTER LAST OF S TO [NOUN,NEXT]; GO TO NOUN1 END ELSE GO TO ERROR SENT0;

SENT1: IF S==[$,SENT,PERIOD] THEN GO TO SUCCESS ELSE GO TO ERROR SENT1;

SUBJ1: IF S==[$,SUBJ,VERBCLASS] THEN GO TO VERBO ELSE GO TO ERROR SUBJ1;

NOUNPHRZ1: IF S==[$,VERB,NOUNPHRZ,SIGMA:$] THEN BEGIN ALTER LAST 3 OF S TO [PRED,SIGMA]; GO TO PRED1 END;
IF S==[$,NOUNPHRZ,SIGMA:$] THEN BEGIN ALTER LAST 2 OF S TO [SUBJ, SIGMA]; GO TO SUBJ1 END ELSE GO TO ERROR NOUNPHRZ1;

ADJST1: IF S==[$,ADJST,ADJCLASS] THEN GO TO ADJ0;
IF S==[$,DETC,ADJST,NOUNCLASS] THEN GO TO SENT0 ELSE GO TO ERROR ADJST1;
```
PRED1: IF S==[$,SUBJ,PRED,SIGMA:$1] THEN BEGIN ALTER LAST 3 OF S TO [SENT,SIGMA]; GO TO SENT1 END ELSE GO TO ERROR PRED1;
PRO1: IF S==[$,PRO,SIGMA:$1] THEN BEGIN ALTER LAST 2 OF S TO [SUBJ,SIGMA]; GO TO SUBJ1 END ELSE GO TO ERROR PRO1;
NOUN1: IF S==[$,DET,ADJST,NOUN,SIGMA:$1] THEN BEGIN ALTER LAST 4 OF S TO [NOUNPHRZ,SIGMA]; GO TO NOUNPHRZ1 END;
If S==[$,NOUN,SIGMA:$1] THEN BEGIN ALTER LAST 2 OF S TO [NOUNPHRZ,SIGMA]; GO TO NOUNPHRZ1 END ELSE GO TO ERROR NOUN1;
ADJ0: IF S==[$,ADJCLASS] THEN BEGIN ALTER LAST OF S TO [ADJ,NEXT]; GO TO ADJ1 END ELSE GO TO ERROR ADJ0;
ADJ1: IF S==[$,ADJST,ADJ,SIGMA:$1] THEN BEGIN ALTER LAST 3 OF S TO [ADJST,SIGMA]; GO TO ADJST1 END;
If S==[$,ADJ,SIGMA:$1] THEN BEGIN ALTER LAST 2 OF S TO [ADJST,SIGMA]; GO TO ADJST1 END ELSE GO TO ERROR ADJ1;
VERB0: IF S==[$,VERBCLASS] THEN BEGIN ALTER LAST OF S TO [VERB,NEXT]; GO TO VERB1 END ELSE GO TO ERROR VERB0;
VERB1: IF S==[$,VERB,NOUNCLASS] THEN GO TO SENT0;
If S==[$,VERB,DETCLASS] THEN GO TO SENT0 ELSE GO TO ERROR VERB1;
DET1: IF S==[$,DET,ADJCLASS] THEN GO TO ADJ0 ELSE GO TO ERROR DET1;

It should be emphasized that except for declarations and a few steps for initialization, the above production grammar could be entered directly into the computer and run as a Formula Algol program to parse sentences (4) through (6) in Chapter 2. Note that sentence (7), "Mary loves the house" would not terminate at the "SUCCESS" exit, but at the "ERROR DET1" exit, since the "DET1" production is attempting to scan for an instance of an adjective while the current entry in the S stack is the terminal "house" and not an adjective.
To illustrate the order sensitivity of certain type 1 productions, observe that interchanging the two productions in the NOUNPHRZ cluster would cause the SUBJECT production to preclude the PREDICATE production, i.e., the grammar would always incorrectly scan predicates as subjects. As seen in this example, the introduction of class definitions for terminals considerably decreases the number of productions required. For example, by introducing the ADJCLASS definition the ADJ0 cluster is reduced from four productions, one for each adjective, to one production for the entire class. In addition, a similar saving is realized for the DET1 and ADJST1 clusters. Some of these savings for type 0 productions, however, will be lost with the later introduction of semantic productions. When the BNF grammar is completely partitioned, as in the example given, i.e., there are no occurrences of adjacent terminals and non-terminals in any alternative, the algorithm simplifies, since context 3 for type 0 and contexts 3, 4, and 5 for type 1 productions never occur.

Once the recognizer has been constructed, we must turn our attention to the introduction of semantic productions to cause the source string to be translated into an expression in the predicate calculus. Our policy is to postpone as long as possible the introduction of semantic productions in the parsing, so as to capitalize on as much context as possible. On the other hand whenever an element of the syntax stack is deleted or replaced in the parsing, some potentially important semantic information may be lost, and a semantic production must appear. Thus, our strategy is to introduce semantic productions at those places and only those places where semantic information may be lost. In terms of
the recognition algorithm, provision for semantic productions must always be made for context 1 type 0 and for contexts 1 and 3 type 1 productions. None of the other contexts require semantic productions, since they do not alter the syntax stack. Taken by themselves the semantic productions form a generative grammar for expressions in the quantificational calculus. Their form is structurally the same as the syntactic productions. The basic difference is that they operate on an independent semantic stack T. Due to technical constraints in Formula Algol there are slight differences in appearance. For example, "+" is used as a delimiter for constituents in the stack rather than a comma, and the pattern "ANY" plays a role analogous to the pattern "$".

The combined syntactic and semantic productions are used for the recognition and translation of source statements respectively. For example, upon recognition of the first statement in Figure 1,

(1) Each resistor in parallel with a capacitor which is ten ohms is an input resistor.

the following wff are constructed:

(2) \( (\forall x)\{R(x) \land (\exists y)\{C(y) \land P(x,y) \land \Omega(y,10)\} \supset I(x) \land R(x)\} \)

(3) \( (\forall x)\{R(x) \land (\exists y)\{C(y) \land P(x,y)\} \land \Omega(x,10) \supset I(x) \land R(x)\} \).

The first wff (2) is the natural interpretation of the sentence (1), but it incorrectly relates the relative pronoun "which" to the noun resistor; the second (3) correctly associates the relative pronoun with the noun capacitor. At this point, however, the correctness of either interpretation has not yet been established.
3.1.2 The Dictionary.

The terminal vocabulary of the source language is partitioned into two subsets: referent terminals and function terminals. The referent terminals are those words which characterize objects or events. In English, referent words are typically nouns, pronouns, adjectives, adverbs, and most verbs. Function terminals on the other hand are those words which serve to delimit or relate the referent words. In English, function words are typically conjunctions, articles, prepositions, and auxiliary verbs. The semantic meaning of each of the lexical items in the referent terminal vocabulary is provided by the dictionary. The dictionary consists of a collection of boolean procedures for the evaluation of predicates corresponding to the referent terminals. The arguments of these predicates are variables which take as value objects in the universe of discourse. Nouns and adjectives generally give rise to one place predicates. Relations such as "greater than" and "to the left of" give rise to two place predicates. Some typical procedures for the predicates triangle(x), black(x), and greater(x,y) are programmed in Formula Algol as follows:

```algol
BOOLEAN PROCEDURE TRIANGLEF(X); VALUE X; FORM X;
BEGIN IF AMONG(TRIANGLE,TYPE(X)) THEN TRIANGLEF← TRUE
   ELSE TRIANGLEF← FALSE END;

BOOLEAN PROCEDURE BLACKF(X); VALUE X; FORM X;
BEGIN IF COLOR(X) = BLACK THEN BLACKF← TRUE
   ELSE BLACKF← FALSE END;
```

1. The essential part of these procedures has been shown, and the slight variation with those same procedures in the program listing in Appendix 2 is only a matter of program-ment technicality.
These procedures of course assume the existence of other numerical and symbolic procedures as well as a set of description lists for the parameters when they are executed. These description lists form the universe of discourse. How they are constructed from a pictorial input will be discussed in the next section.

3.1.3 The Universe of Discourse.

The universe of discourse for the user of GRANIS is defined to be a picture which may be altered dynamically during the course of conversation. It is input directly to the computer on a graphic display console by means of a light pen or cursor. Before the pictorial information can be used in the processing of natural language statements, it must be analysed and represented in a suitable fashion for later processing.

The fundamental data structure used in the internal representation of pictorial information is the description list. The ring structure described by Sutherland in Sketchpad [78] and by Roberts in CORAL [66] can be simulated as a special case of this general format.

To illustrate the use of description lists, the format of the universe of discourse for the NOR circuit given in Figure 1 is as follows:
In general, the objects OB can possess an arbitrary number of attributes each of which possesses an associated value list.

But how does one automatically abstract from the two dimensional information contained in the picture that part which is relevant to the construction of these description lists? The basic problem of two dimensional analysis has not been solved in any general fashion. Thus, there is little theoretical knowledge to draw on. On the other hand the display hardware described by Quatse [60] must itself perform a linearization of the pictorial data, since the regeneration display file for the picture in core memory is in fact a linear array. This is the same file that the hardware uses to refresh the screen on a continuous basis, and thereby maintain a flicker-free display. The file is composed of vector, header, and character string commands. The complete set of display commands and their formats is shown in Table 2. The task of processing a picture then reduces to processing a logic array of display commands and producing as output a set of description lists. This
<table>
<thead>
<tr>
<th>Command</th>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELIMIT</td>
<td>010000</td>
<td>Address field</td>
</tr>
<tr>
<td>HEADER</td>
<td>010000</td>
<td>X and Y coordinates</td>
</tr>
<tr>
<td>VECTOR STRING</td>
<td>010000</td>
<td>X and Y coordinates</td>
</tr>
<tr>
<td>CHARACTER STRING</td>
<td>10000</td>
<td>X and Y coordinates</td>
</tr>
<tr>
<td>COMPARE</td>
<td>10000</td>
<td>Character</td>
</tr>
<tr>
<td>LEFT MARGIN</td>
<td>110000</td>
<td>X and Y coordinates</td>
</tr>
<tr>
<td>RIGHT MARGIN</td>
<td>111000</td>
<td>X and Y coordinates</td>
</tr>
<tr>
<td>CYCLE</td>
<td>10000</td>
<td>Address field</td>
</tr>
<tr>
<td>STORE</td>
<td>00</td>
<td>Address field</td>
</tr>
</tbody>
</table>

Notes: 1. A 13 bit address field is required to locate addresses in the eight thousand word core memory used to hold display files for all three consoles.
2. Ten bit "X" and "Y" coordinates are used to discriminate points in a 1024x1024 matrix, which is the resolving power of the screen.

TABLE 2. DISPLAY COMMANDS
task has considerable structural similarity to the task of parsing a source statement in natural language and producing as output an expression in the predicate calculus. Let us explore the possibility of using the same syntax directed techniques in this new application, i.e., recognizing and translating pictures rather than text into a suitable internal representation.

First, we observe that because the display file representation is constructed dynamically by the hardware as the user draws the picture, the file contains not only information about what is displayed on the screen at any moment, but also the complete history of the order in which the total picture was drawn. At this point we have a choice. We may choose either to disregard this order information inherent in the hardware representation of the display file or to capitalize on this preprocessing in some fashion, thereby considerably simplifying the recognition process itself. The disadvantage of this latter choice, however, is to require the user to observe certain conventions which specify how certain parts of the picture are to be drawn in addition to what the picture is to look like when he is done. If these conventions are sufficiently natural, however, so that a user would probably not violate them anyway, then it would inefficient not to take advantage of this hardware preprocessing. Hence, we have adopted this second choice for GRANIS.

Just how restrictive are these conventions? By way of illustration, the conventions for drawing geometric figures are as follows:
1. A user must draw figures one at a time, i.e., once having started a figure, he must complete it before going on to the next one, and having completed it, he must never return to it.

2. Each polygon must be drawn as a sequence of vectors forming a polygonal path.

3. Upon hitting the last "mark" for the endpoint of the polygon, the user must press the vector function button off, and then on again, before starting a new figure.

The purpose of the first rule is that it guarantees that each figure is distinct and occupies a set of contiguous locations in the display file; the second rule guarantees that there is only one header command for each figure; the third rule has the effect of inserting a blanked vector of zero length after the vector string sequence. This operation would be necessary in any event if the user expected to capitalize on the off-line translation features inherent in the hardware (cf. Quatse [60, p. 42]). Note that there are no restrictions on the order in which figures are to be drawn, whether figures may be overlaid, or on the position, size, or orientation or any particular figure. Similar conventions such as those above hold for chemistry molecules and electrical circuits.

A consequence of inadvertently violating one of the above rules while drawing a figure is the possibility that the figure may not be correctly recognized even though to outward appearances it is identical to one which has already been scanned correctly. It should be clearly pointed out, however, that the introduction of these conventions in drawing figures is merely an expedient to gain more efficiency in the utilization of memory space and processing time in the picture scanner portion of GRANIS which
has no immediate theoretical interest in itself. There is no reason in principle why these conventions could not be dispensed with upon introduction of a set of separate preprocessing routines to evaluate absolute coordinates for each vector, and determine the connectivity of the total picture, irrespective of how it was drawn, by means of a set of $\varepsilon$-neighborhood computations.

A somewhat related problem is the possibility that the picture grammar as constructed may recognize a peculiar object drawn on the screen as a legal figure or symbol even though the peculiar object is only related to the legal figure by means of an extreme topological distortion. In certain situations such recognition is undesirable. For example, GRANIS may identify some object as a "resistor" even though that object bears little outward resemblance to the standard resistor symbol from the human point of view. What this means is that the class of objects which potentially can be generated by the grammar is larger than the intended class of well formed objects which the grammar is used to recognize. However, in practice this problem has never arisen, since users are inclined to draw only those objects they expect to make statements about later. In any event as the grammar is extended, this problem becomes less severe, since distinguishing between similar equivalence classes requires finer distinctions, and violations of intuition are less likely.

By means of a simple example let us now see how syntax directed techniques are actually used in the recognition and translation of figures into description lists. For our example let us take the four geometric figures illustrated in Figure 2.
The pentagon, triangle, and both squares are represented internally in the computer regeneration memory as a display file of the following format:

1. 30001770407 2. 12000000222 3. 12000134053
4. 12020256157 5. 12060242157 6. 12040154036
7. 12004000000 8. 30001114660 9. 12000570123
10. 12040562107 11. 12060000231 12. 12040000000
13. 30002077137 14. 12000214000 15. 12000000177
16. 12040214000 17. 12060000116 18. 12004000000
19. 30002116154 20. 12000334000 21. 12000000233
22. 12040334000 23. 12060000232 24. 1204000000

Each of the 24 words in the display file is an octal number representing a 32 bit word formatted according to the commands given in Table 2. Thus, words 1, 8, 13, and 19 are header commands which give the initial position on the screen of the vector strings following. A command of the form 12004000000 is a vector string with a $\Delta X$ and $\Delta Y$ of zero length. Such a command is generated automatically by the hardware whenever a vector string is terminated, and a new figure initiated on the screen. It is used by the scanner to signal the termination of a given figure. Words 7, 12, 18, and 24 are of this form, and are the terminations corresponding to the header words 1, 8, 13, and 19. Thus, by inspection we immediately observe that there are four vector strings in the display file; the first consists of five vectors, the second of three, and the third and forth of four.

The actual description lists constructed by the picture scanner for Figure 2 are as follows:
How does the program take as input the 24 word logic array and produce the four description lists above? First, some preliminary analysis is performed by appropriately masking each word, determining the number of objects in the picture, and the number of sides in each object. Then in the geometry package, a routine is executed for any particular object to determine whether it is a polygon, as distinct from a mere polygonal path. A polygonal path is a polygon if and only if the end point and the initial point coincide. In terms of the primitive ΔX and ΔY information available to us from the logic array, this condition is satisfied if

\[ \sum_{i=1}^{n} \Delta X_i = \sum_{i=1}^{n} \Delta Y_i = 0 \]

where \( n \) is the number of sides. To allow for human error in drawing, this condition is relaxed to

\[ \left| \sum_{i=1}^{n} \Delta X_i \right| < \varepsilon \quad \text{and} \quad \left| \sum_{i=1}^{n} \Delta Y_i \right| < \varepsilon \]

where the tolerance \( \varepsilon \) defines a rectangular neighborhood around the initial point. Coincidence now requires that the end point merely lie in this neighborhood. In practice it has been found that an \( \varepsilon = 8 \) points is adequate. Although the new definition is more relaxed, the concept of polygon is still precisely defined.

---

1. "BLACK" is indicated by cross hatching the figure.
If the object is indeed a polygon, then a general area computation is evoked using the cross product rule for vectors. The area of a triangle with one vertex at the origin and the other two vertices at \((x_1, y_1), (x_2, y_2)\) is given by \(\frac{|\overrightarrow{a} \times \overrightarrow{b}|}{2}\) where \(\overrightarrow{a} = x_1 \hat{i} + y_1 \hat{j}\) and \(\overrightarrow{b} = x_2 \hat{i} + y_2 \hat{j}\). That is to say
\[ A = \frac{1}{2} |x_1 y_2 - y_1 x_2| \]

Then for a polygon of any shape and coordinates \((x_i, y_i)\) given in a counterclockwise direction, if we sum the signed areas of the triangles formed by each side of the polygon and the origin, we obtain for the area of the polygon
\[ A = \frac{1}{2} \sum_{i=1}^{n} (x_i y_{i+1} - x_{i+1} y_i) \]

where \(n\) is the number of sides.

For actual construction of description lists, we use Formula Algol assignment statements. For example,

```algol
FOR I-1 STEP 1 UNTIL OBNO DO BEGIN
  IF POLYGONP(OB[I]) THEN BEGIN
    THE TYPE OF OB[I] IS POLYGON;
    THE SIZE OF OB[I] IS AREA;
    IF BLK THEN THE COLOR OF OB[I] IS BLACK
    ELSE THE COLOR OF OB[I] IS WHITE;
    IF SIDES = 3 THEN THE TYPE OF OB[I] IS ALSO TRIANGLE;
    IF SIDES = 4 THEN THE TYPE OF OB[I] IS ALSO QUADRILATERAL;
    IF SIDES = 5 THEN THE TYPE OF OB[I] IS ALSO PENTAGON END;
    ...
  END;
```

is taken directly from the program, illustrating the readability of the language. OBNO is the number of objects in the universe.

1. This result is due to Roberts [65, p.17]. For a short proof using Stokes' Theorem see T. Evans [24, p. 47].
of discourse, and \textsc{POLYCONP} is a boolean procedure which determines whether \( \text{OB}[1] \) is a polygon, and if so, computes the boolean variable \( \text{BLK} \), the integer variable \( \text{SIDES} \), and the real variable \( \text{AREA} \) as described above.

3.1.4 Evaluation.

Once a well formed picture has been drawn, the corresponding description lists constructed, a well formed sentence entered, and the corresponding interpretations built up by the semantic component, we are in a position to determine the truth-value of the input with respect to the picture. As defined by the system, the truth-value of the input sentence can take on one of three values: valid, invalid, or vacuous. The vacuous case is explicitly distinguished to facilitate the resolution of syntactic ambiguity. The evaluation process takes place in three steps: quantifier elimination, predicate evaluation, and truth-value determination.

First, all occurrences of universal and existential quantifiers are eliminated by appropriately generating the objects in the universe of discourse. If \( \varphi(x) \) is a wff to be evaluated, the repeated application of the productions

\[
(\forall x)\varphi(x) \rightarrow \bigwedge_{i=1}^{n} \varphi(\text{OB}_i) = \varphi(\text{OB}_1) \land \varphi(\text{OB}_2) \land \cdots \land \varphi(\text{OB}_n)
\]

\[
(\exists x)\varphi(x) \rightarrow \bigvee_{i=1}^{n} \varphi(\text{OB}_i) = \varphi(\text{OB}_1) \lor \varphi(\text{OB}_2) \lor \cdots \lor \varphi(\text{OB}_n)
\]

will eliminate all occurrences of quantifiers where \( n \) is the number of objects in the domain. This process must always
terminate, since the universe of discourse contains at most a finite number of distinct objects. In Formula Algol each of the productions is implemented in a formula procedure as follows--

For universal quantifiers:

\[
\text{FORM PROCEDURE } \text{A}(X, \text{SCOPE}); \text{VALUE } X, \text{SCOPE}; \text{FORM } X, \text{SCOPE}; \\
\text{BEGIN INTEGER } I; \text{FORM TEMP}; \text{TEMP} \leftarrow \text{EVAL}(X)\text{SCOPE}(0,[1]); \\
\text{FOR } I \leftarrow 2 \text{ STEP } 1 \text{ UNTIL OBNO DO} \\
\text{TEMP} \leftarrow \text{TEMP} \land \text{EVAL}(X)\text{SCOPE}(0,[I]); \\
\text{A} \leftarrow \text{TEMP} \text{ END;}
\]

For existential quantifiers:

\[
\text{FORM PROCEDURE } \text{E}(X, \text{SCOPE}); \text{VALUE } X, \text{SCOPE}; \text{FORM } X, \text{SCOPE}; \\
\text{BEGIN INTEGER } I; \text{FORM TEMP}; \text{TEMP} \leftarrow \text{EVAL}(X)\text{SCOPE}(0,[1]); \\
\text{FOR } I \leftarrow 2 \text{ STEP } 1 \text{ UNTIL OBNO DO} \\
\text{TEMP} \leftarrow \text{TEMP} \lor \text{EVAL}(X)\text{SCOPE}(0,[I]); \\
\text{E} \leftarrow \text{TEMP} \text{ END;}
\]

As a specific example of the quantifier elimination process, let us take the statement

(4) Each polygon smaller than a black triangle is a square.

The interpretation of (4) is the wff

(5) \( (\forall x)(P(x) \land (\exists y)(B(y) \land T(y) \land Sm(x,y)) \supset Sq(x)) \)

Using the description lists on page 71 derived from Figure 2 as our universe of discourse, if we label the pentagon "a", the triangle "b", the upper square "c", and the lower square "d", then the result of quantifier elimination yields

---

1. (5) is the only interpretation of (4), since (4) is unambiguous with respect to the grammar.
\[
\{P(a) \land [B(a) \land T(a) \land Sm(a,a) \lor B(b) \land T(b) \land Sm(a,b) \lor B(c) \land T(c) \land Sm(a,c) \\
\lor B(d) \land T(d) \land Sm(a,d)] \Rightarrow Sq(a) \} \land \\
\{P(b) \land [B(a) \land T(a) \land Sm(b,a) \lor B(b) \land T(b) \land Sm(b,b) \lor B(c) \land T(c) \land Sm(b,c) \\
\lor B(d) \land T(d) \land Sm(b,d)] \Rightarrow Sq(b) \} \land \\
\{P(c) \land [B(a) \land T(a) \land Sm(c,a) \lor B(b) \land T(b) \land Sm(c,b) \lor B(c) \land T(c) \land Sm(c,c) \\
\lor B(d) \land T(d) \land Sm(c,d)] \Rightarrow Sq(c) \} \land \\
\{P(d) \land [B(a) \land T(a) \land Sm(d,a) \lor B(b) \land T(b) \land Sm(d,b) \lor B(c) \land T(c) \land Sm(d,c) \\
\lor B(d) \land T(d) \land Sm(d,d)] \Rightarrow Sq(d) \}
\]

which is the quantifier free form of (5) in the given domain.

The second evaluation process evaluates each predicate by determining the truth-value of each of the boolean procedures in the quantifier free form. Such procedures were illustrated on pages 62 and 63. In the example above we obtain

\[
[T \land [TAFAF \lor TAFAF \lor FAFAF \lor FAFAF] \Rightarrow F] \land \\
[T \land [TAFAF \lor TAFAF \lor FAFAF \lor FAFAF] \Rightarrow F] \land \\
[T \land [TFAAT \lor TATAT \lor FAFAT \lor FAFAT] \Rightarrow T] \land \\
[T \land [TFAAT \lor TATAT \lor FAFAT \lor FAFAT] \Rightarrow T].
\]

Finally, knowing the rules of combination for each logical connective, the truth-value of the resulting expression is determined. If the expression is false, then the final truth-value is assigned the value invalid. If true, however, then it must be further verified that the expression is genuinely true rather than vacuously true. For example, sentence (4) would be true at this point even if there were no black polygons in the universe of discourse. This follows from the logic of what is called "material implication," and corresponds in a sense to the notion that one can legitimately attribute anything at all to something which doesn't exist. That is, assuming there were no black polygons on the screen as say in Figure 7, then one might argue that all black polygons were squares, pentagons, circles, or whatever with impunity.

\footnote{In practice the second and third steps are not distinct internally.}
To avoid this sort of anomaly, all occurrences of universal quantification, if any, are replaced by existential quantification and the expression is reevaluated. If the expression is still true upon reevaluation, then the final truth-value is assigned the value valid, otherwise, vacuous. In the case of our example above we obtain a "true" for both evaluation and reevaluation, and therefore it follows that the input statement is valid for Figure 2.

In terms of operation, the time to process sentence (4) with respect to Figure 2 on the G-21 computer is about eighteen seconds. In general one observes that processing time increases linearly with the complexity of the picture or the length of the input sentence, but exponentially with the number of nested relative phrases. Processing time at execution, however, has actually proved less of a problem than exhausting available space. This problem is sometimes referred to as "intermediate expression swell" because even though the initial input data may be quite modest, and the final output a mere yes or no, the storage requirements for intermediate results may be extremely large. A Formula Algol garbage collection routine which returns unnecessary intermediate results to a linked list of available space has reduced this problem to manageable proportions.

3.2 Results.

To illustrate the ability of GRANIS to translate simple English sentences into the predicate calculus Table 3 shows seven representative sentence forms among the many possible forms specified by the grammar. In the domain of geometry
<table>
<thead>
<tr>
<th>SIMPLE ENGLISH SENTENCE</th>
<th>LOGICAL TRANSLATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Positive Universal Every polygon is a triangle</td>
<td>$(\forall x){\text{Poly}(x) \Rightarrow \text{Tri}(x)}$</td>
</tr>
<tr>
<td>2. Negative Universal No resistor is ten ohms.</td>
<td>$(\forall x){\text{Res}(x) \Rightarrow \neg \text{Ohms}(10, x)}$</td>
</tr>
<tr>
<td>3. Positive Existential Some atom is carbon.</td>
<td>$(\exists x){\text{A}(x) \land \text{C}(x)}$</td>
</tr>
<tr>
<td>4. Negative Existential Some square is not black.</td>
<td>$(\exists x){\text{Sq}(x) \land \neg \text{Bl}(x)}$</td>
</tr>
<tr>
<td>5. Relative Phrase Each white polygon bigger than a hexagon is a rectangle.</td>
<td>$(\forall x){\text{Wt}(x) \land \text{Poly}(x) \land (\exists y){\text{Hex}(y) \land \text{Bg}(x, y)} \Rightarrow \text{Rec}(x)}$</td>
</tr>
<tr>
<td>6. Compound Phrase Every atom is either hydrogen, oxygen, or carbon.</td>
<td>$(\forall x){\text{A}(x) \Rightarrow {\text{H}(x) \lor \text{O}(x) \lor \text{C}(x)}}$</td>
</tr>
<tr>
<td>7. Compound Relative Phrase Every PNP transistor connected to an output resistor and connected to a bias resistor is part of a flip-flop.</td>
<td>$(\forall x){\text{PNP}(x) \land \text{Trn}(x) \land (\exists y){\text{Out}(y) \land \text{Res}(y) \land \text{Con}(x, y)} \land (\exists z){\text{Bi}(z) \land \text{Res}(z) \land \text{Con}(x, z)} \Rightarrow (\exists w){\text{FF}(w) \land \text{Pt}(x, w)}} $</td>
</tr>
</tbody>
</table>
Figures 3 through 7 illustrate typical pictures of geometric figures which can be recognized by GRANIS. The system has proved sufficiently general in conception to permit a fairly easy conversion from one subject matter to another. To convert from one domain to another the user need only recompile GRANIS to incorporate the new dictionary and picture recognizer. The dictionary redefines the relevant terms in the referent terminal vocabulary, while the picture scanner permits the recognition of new objects in the new universe of discourse. In the domain of electric circuits for example, Figures 8 through 12 illustrate typical circuits which can be recognized. In the domain of organic chemistry molecules, Figures 13 through 18 illustrate typical molecules which can be recognized.

To illustrate the power of GRANIS in resolving ambiguity, consider the following three sentences:

(4) Each polygon smaller than a black triangle is a square.
(6) Each black polygon smaller than a triangle is a square.
(7) Each polygon smaller than a triangle which is black is a square.

together with the nine possible displays in Figure 19. The first two sentences are unambiguous, while the third has two possible interpretations. The natural interpretation of sentence (7) is equivalent to sentence (4); the other is equivalent to sentence (6). The following 3x9 matrix gives the truth-value of each of the sentences for each of the nine pictures:
FIGURE 6
DIODE LOGIC

FIGURE 8
FIGURE 9

NOR CIRCUIT
FIGURE 10

NOR CIRCUIT
LOW LEVEL LOGIC

FIGURE 11
CURRENT MODE LOGIC

FIGURE 12
FIGURE 13
FIGURE 14
FIGURE 17
FIGURE 18
FIGURE 19
Sentence (4) is vacuous in contexts 7, 8, and 9, since there is no occurrence of a black triangle in any of these pictures. Similarly, sentence (6) is vacuous in contexts 3, 6, and 9, since there is no occurrence of a black polygon smaller than a triangle in any of these pictures. In evaluating the truth-value of sentence (7) by means of the conventions adopted in section 2.6.3, if the natural interpretation, sentence (4), is non-vacuous, then the second interpretation, sentence (6), will not even be evaluated, and the truth-value of sentence (7) is defined to be the same as sentence (4). If the natural interpretation is vacuous, however, the second interpretation will be evaluated, and the truth-value of sentence (7) is defined to be the same as sentence (6).

Appendix 2 gives the complete Formula Algol program listing for these examples. With the help of the comments and some knowledge of ALGOL, the reader at this point should have little difficulty in understanding how GRANIS analyzes and evaluates sentences.
There are a large number of dimensions along which the work described thus far could be expanded. The following four directions will be explored in some detail:

1. the habitability of the natural language grammar
2. inferential power
3. knowledge acquisition
4. adaptive properties

4.1 Habitability.

An extremely difficult problem for question answering systems is the stability of restricted subsets of natural language. As Oettinger [53] pointed out, attempts to define easily manageable fragments of natural language generally fail because the subsets become unstable for one of two reasons-- either due to a drift in the direction of a formal mathematical notation or to a drift in the opposite direction toward a full use of an unrestricted vernacular. In the former case the advantages of natural language are lost, and successful use of the system requires considerable prior indoctrination. In the latter case, the enigmatic problems of unrestricted natural language contantly require extensions to the grammar.

GRANIS- the computer model actually programmed- tends to fall in the latter category. To be sure there are several factors which tend to constrain the user from casual discourse:

1. He is seated in front of a computer display console;
2. He must enter input on a typewriter keyboard;
3. He is warned in advance that the system comprehends very little English;
4. He is there to discuss a particular class of pictures;
5. His referent vocabulary is generally limited to descriptions of objects currently displayed.

Nevertheless, there is an inadvertent tendency to compose phrases which are simply not acceptable to the grammar as defined by GRANIS. Although error recovery from these situations is adequate, the error messages are generally symptomatic of a global difficulty, and the user still lacks specific information on how to change his input sentence so that it will be acceptable to the system. This often results in a trial and error kind of interaction which has little of the desirable qualities of a genuine conversation.

How might one investigate the stability of a micro-grammar for a fragment of natural language? In working on a related problem Watt [83] defined a concept which he called habitability. The habitability of a given micro-grammar, according to Watt, is a measure of the ability of the grammar to accept elementary lexical and syntactic extrapolations of phrases which are already acceptable to the grammar. To illustrate this definition let us assume that

(1) Is a capacitor here?
is acceptable to the system. Then an elementary syntactic extrapolation might be

(2) Is there a capacitor here?
That is to say, (2) is a strict paraphrase of (1) under identical contextual conditions. Some non-elementary syntactic extrapolations might be
(3) Are there any capacitors here?
(4) Do there happen to be any capacitors here?
(5) Can you find any capacitors here?
(6) Do you see anything in the way of a capacitor here?

Each of the sentences (1) through (6) form an equivalence class which Watt refers to as a paraphrastic set. An elementary lexical extrapolation might be

(7) Is a resistor here?

or

(8) Is a diode here?

Non-elementary lexical extrapolations might be

(9) Is a molecule here?
(10) Is a rectangle here?
(11) Is a computer here?

Watt avoids the problem of providing an operational definition of the term "elementary" in the kinds of extrapolations with which he is concerned. Although one might set some arbitrary limits for syntactic extrapolations, saying that elementary syntactic extrapolations are those which require at most the application of two transformations from a given grammar, this sort of distinction is ad hoc and may still violate our intuitive notion of what it means for a micro-grammar to be syntactically habitable. Even though the universe of discourse may be exhibited in the form of a display, a similar criticism may be levied at any attempt to define operationally an elementary lexical extrapolation. In spite of these difficulties in defining an absolute notion of habitability, one can easily compare the relative habitability of two grammars. Thus, the notion of habitability should be useful in approaching the problem of stability in a grammar as it expands or contracts through use.
We have observed that the grammar in our computer model is unstable in the direction of an ever increasing number of productions due to constraints on the flexibility of the input language and a desire to exceed the restrictions on the universe of discourse. Moreover, no absolute boundary can be foreseen such that expansion beyond that point is not required for augmenting the usefulness of the system. Indeed, English itself has no absolute boundaries, and is constantly undergoing expansion, albeit at a slow rate. One of the important properties of natural language is the ability to coin a new phrase or term as it becomes useful to do so. Thus, one concludes that the system must incorporate within itself provision for expansion of the grammar, if it is ever to satisfy the requirement of habitability.

How might such expansion take place? Yershov [89] has suggested that the system, distinguishing those text fragments which are unclear to it, assign to the user a series of questions on the distinguished fragments. The form and character of the questions is defined by the fragments and the manner in which they are embedded in the overall text. The user's replies to these questions may be regarded as paraphrases of the unclear fragments. The system substitutes them into the initial text and analyses it again. If necessary, the system again assigns questions to the user, and in this way establishes a dialog between the man and the machine. By means of this dialog the user continually simplifies the formulation of his input statements until they are completely understood by the system.

Once the system and the user have converged on a mutually acceptable paraphrase for a new phrase, this paraphrase will be incorporated into the transformational component of the grammar, and a mandatory transformation will be automatically
evoked whenever the new phrase is encountered in all similar future contexts.

And how might one implement such a proposal within the framework of GRANIS? Such a dialog would require in essence changing the binding time of certain system parameters from compile-time to run-time. The productions themselves would have to be treated as formula input data rather than compiled statically into the program. Then for production data structures we would need appropriate primitive operators such as

1. constructors for new productions,
2. alteration and deletion operators for existing productions,
3. extractors for parts of productions,
4. predicates for determining if a production is an instance of some production schema.

It is not clear how one might successfully drive such primitive operators with commands given in English. The construction of mandatory paraphrastic transformational productions, however, involves only the substitution of terminals for terminals and they can be executed before the parsing process begins. If the phrase \( \alpha \) in the context \( \psi_1, \psi_2 \) is not recognized by the system, then an error recovery procedure initiates a dialog to determine ultimately a paraphrase \( \beta \), such that \( \psi_1 \beta \psi_2 \) can be recognized. If the user approves of the paraphrase, then the system constructs a syntactic production of the form

\[
\psi_1 \alpha \psi_2 \rightarrow \psi_1 \beta \psi_2
\]

and places it at the top of the sequence of productions to be executed (if appropriate) before any others. If \( \varphi_1 \alpha \varphi_2 \rightarrow \varphi_1 \beta \varphi_2 \)
already exists, then the two productions should be consolidated as $X_1 \alpha X_2 \rightarrow X_3 \beta X_2$ where $X_1 = \varphi_1 \cup \psi_1$ and $X_2 = \varphi_2 \cup \psi_2$. If the paraphrase $\beta$ can be thought of as a definition for $\alpha$, then of course the production $\alpha \rightarrow \beta$ should be constructed, making the transformation context free.

On the other hand one must pay a price for this systematic escalation of variability in the grammar. The price of generality is efficiency. If the productions are no longer compiled into the program, they must be executed interpretively, and the parsing and interpretation of input sentences will take longer as a function of the administrative overhead in passing parameters during procedure calls. Each production corresponds to a procedure call and already 90% of the total interpretation time is spent in production processing. (The remainder is taken up mostly in I/O operations with the displays.) Perhaps a compromise can be achieved whereby the base component of the grammar is compiled and the transformational component is executed interpretively. More empirical evidence must be obtained, however, before the proper trade-off between a compiled and an interpretive grammar can be decided upon.

Another interesting problem in regard to the habitability of GRANIS is how to handle the increasing complexity of sentence forms without continually expanding the base component. As suggested on page 26, the basic approach to this problem is to require that the transformational component map the more complex sentences into simpler kernel sentences which are acceptable to the base component. The next question which arises is what is the proper trade-off between the base component and the transformational component?
One extreme position suggested by Simmons [73] is that the base component be restricted to kernel sentences of the form "subject-verb-nominal". For example, given the input sentences

(12) Jack and Jill went up the hill to fetch a pail of water.

(13) The quick brown fox jumped over the lazy dog.

the kernel sentences for (12) would be

1. Jack went up the hill.
2. Jill went up the hill.
4. Jill fetched a pail.
5. Pail (is of) water.

and for (13) would be

1. The fox is quick.
2. The fox is brown.
3. The fox jumped.
4. The jumping was over.
5. The dog is lazy.

The difficulty with this approach is that it places too much of the burden of syntactic analysis on the transformational component of the grammar with the undesirable consequence of further isolating syntax from semantics. Another difficulty is that additional ambiguity is introduced in the process of derivation, and therefore reference terms must be added to the kernel sentences which permit one to reconstruct the original input sentence.

In GRANIS the approach taken is to derive kernel sentences only when the sentence consists of genuinely independent clauses delimited by a conjunction or other punctuation. In such a case the scope of quantification will not extend across clause boundaries, thereby introducing
ambiguity when kernels are derived. Thus, both sentences (12) and (13) are already considered to be kernel sentences, and would be translated directly into the following wff:

\[(\exists x_1)[\text{Jack}(x_1) \land (\exists x_2)[\text{Jill}(x_2) \land (\exists x_3)[\text{Hill}(x_3) \\
\land \text{Go}(x_4 \land x_5, x_3)] \land (\exists x_4)[\text{Pail}(x_4) \land \text{Fetch}(x_1 \land x_2, x_3)] \\
\land (\exists x_5)[\text{Water}(x_5) \land \text{Of}(x_4, x_5)])]

\[(\exists x_1)[\text{Quick}(x_1) \land \text{Brown}(x_1) \land \text{Fox}(x_1) \land (\exists x_2)[\text{Lazy}(x_2) \\
\land \text{Dog}(x_2) \land (\exists x_3)[\text{Jump}(x_1, x_2, x_3) \land \text{Over}(x_3)])].

4.2 Inferential Power.

Another important avenue for further exploration is the inferential power of GRANIS. There are several aspects to expanding inferential power. First we will discuss some extensions to the predicate calculus notation to facilitate operations with factored predicates and additional quantifiers. Second, an extension to enhance the ambiguity resolving power of GRANIS will be described. Finally, a brief exploration of the problems of handling large blocks of input text will be undertaken.

4.2.1 Extending the Predicate Calculus.

A major source of abbreviation in natural language derives from the factorization of common predicates governing compound arguments. For example, we say

\[(16) \text{Draw a rectangle and a triangle.}
\]
rather than

\[(17) \text{Draw a rectangle and draw a triangle.}
\]

---

1. We will discuss parameters which are themselves boolean expressions in the next section.
Now from the predicate calculus point of view, the compound object "a rectangle and a triangle" in (16) may be thought of as the result of factoring out the repeated predicate "draw" in (17). Just as the logical translation of (17) is $D(r) \land D(t)$, we can imagine a logical notation as suggested by Bohnert [9, p. 16] which permits the representation of the more compact form of the sentence (16) as $D(r \land t)$.

That is, we can construct a "factoring law" which establishes the equivalence

$$ (18) \quad D(r \land t) \equiv D(r) \land D(t). $$

Compound subjects or indirect objects which also occur in English can be handled in a similar manner. For example, if the immediate logical interpretation of

$$ (19) \quad \text{a goes from b or c to d or e}. $$

is $G(a,(b\lor c),(d\lor e))$ where "G" stands for "goes", then successive distributions would yield the underlying logical translation

$$ (20) \quad G(a,b,d) \lor G(a,b,e) \lor G(a,c,d) \lor G(a,c,e). $$

To construct a logical notation of this sort in which factoring and distribution would be possible, we must augment the definition of well formed formula (wff) so as to make compound individual expressions acceptable arguments for predicates. Thus for example, $P(a \land (b \lor c) \land d, \neg e \land f)$ would be a 2-place augmented well formed formula (awff). In the actual processing of such predicates it may be more desirable to handle them directly rather than first expanding them before processing.

Another extension of the predicate calculus notation might be to incorporate new quantifier types such as iota quantifiers which correspond to the English definite article
"the", or imperative and interrogative quantifiers which correspond to imperative and interrogative statements respectively. In English, when we say "the object" with a certain property, we mean that there exists one and only one object having that property. Thus, it would be incorrect to say "the black square in Figure 6 is above the pentagon," since there are two black squares. Russell introduced the notation \((\exists x)F(x)\) to mean "the particular thing \(x\) having the property \(F\)" (cf. Reichenbach [64, p. 258]). The operator in this expression is an inverted Greek "\(\epsilon\)", and is therefore called the iota operator; \(x\) in this expression is a bound variable. Imperative and interrogative operators can be introduced in an obvious way. For example, the sentences

\begin{align*}
(21) & \text{ Draw a triangle.} \\
(22) & \text{ Is there a triangle?}
\end{align*}

might be translated as

\begin{align*}
(23) & \neg x \neg (\text{Draw}(x) \land \text{Triangle}(x)) \\
(24) & \exists x \neg (\text{Triangle}(x))
\end{align*}

What are the rules of inference for these proposed quantifiers? We can write a general definition for the iota quantifier as

\begin{align*}
(25) & \neg x F(x) = \text{def} \ (\exists x)(F(x) \land (\forall y)(F(y) \supset (y=x)))
\end{align*}

although this is probably not the way it should be implemented. The imperative and interrogative quantifiers are not as simple. For elementary imperative statements, i.e., those where the command corresponds to a primitive operation already programmed as a Formula Algol procedure, the procedure should be executed, and the expression evaluated first replacing the imperative operator with an existential operator. If the evaluation is
true, then the procedure was executed correctly. Otherwise, clarification from the user is necessary. On the other hand if the operation is not primitive, then some additional inferences must be made by first mapping the imperative statement into its corresponding declarative form, and then decomposing this declarative statement into more elementary kernels each of which can then be mapped back into its corresponding imperative form (cf. Simon [74]).

For interrogative quantifiers, if the question is a simple "yes-no" question, then it can be answered by merely evaluating the truth-value of the expression obtained by replacing the interrogative quantifier with an existential quantifier. Otherwise, the situation is much more complex, and the issue of "entailment" arises. Most questions make only implicit reference to the class of possible answers which are permissible, and it is a difficult task merely to determine what assumptions are made about the class of legal replies (cf. Belnap [4]).

4.2.2 Ambiguity Resolution.

At the end of Chapter 2 we proposed a strategy for resolving syntactic ambiguity which required ordering all logically possible interpretations in a natural way, evaluating the most natural one first, and evaluating the next most likely only in the event this first interpretation was vacuous with respect to the universe of discourse. In GRANIS this was one of the important ways that feedback of semantic information occurred in the syntactic analysis of input sentences. If we assume that the interpretation can
be ordered in a natural way, are there any other reasons for entertaining a second interpretation, aside from the case where the natural interpretation is vacuous? A more useful and general criterion might be that the natural interpretation was not "meaningful" in some fashion, where vacuousness is merely a special case. Without attempting an exhaustive investigation of the concept of "meaning", some brief consideration of the relevant issues should be given.

Both Katz [34] and Chomsky [11] have developed a taxonomy of anomalous sentences. Some of these categories are clearly syntactic, some clearly semantic, and others not easily identifiable along a syntax-semantics continuum. Chomsky specifies two kinds of rules in his syntax to avoid deviant sentences: subcategorization rules and selectional rules.

1. The subcategorization rules prevent improper verb categories as a function of the immediate structure of the phrase marker around the verb. A violation of this type of rule, for example, might result in an intransitive verb being placed in a sentence with a direct object, or a pre-adjectival verb being placed in an intransitive sentence (e.g., "John runs a book" or "John seems quickly").

2. The selectional rules establish agreement requirements among such parts of speech as

   (i) adjectives and the nouns they modify,
   (ii) subjects and their verbs, and
   (iii) verbs and their objects.

These rules would prevent such utterances as "ten ohm capacitors" and "sour bottle" referred to in the introduction,
or "drink concrete" referred to in section 2.3, or such innocent questions as

(26) What is the temperature of an atom?

It is not clear whether these rules belong in syntax, semantics, or both, but in the realm of syntax, Chomsky [11, p. 83] defines a set of binary selectional features which define subclasses of nouns which demand certain patterns of agreement in corresponding verbs and adjectives. These features are ±common, ±count, ±abstract, ±animate, ±human, and are hierarchically organized as follows:

```
+    +    -
|    |    |
+----+----+
|    |    |
+----+----+
|    |    |
+----+----+
|    |    |
+----+----+
```

(27) The rock admires the scientist.

would be ruled out on syntactic grounds by Chomsky, since "admires" demands a human subject whereas "rock" is inanimate. On the other hand there is a close correlation between selectional rule violations and contradictions, and a contradiction is certainly a semantic concept. A selectional rule violation is not properly a contradiction in itself, since by definition a contradiction is a statement which is false under all interpretations for any universe of discourse, while a selectional rule violation cannot even be properly said to have a truth-value.
However, most selectional rule violations do entail a contradiction. For example,  

(28) The married spinster is beautiful.  
a selectional rule violation, entails the derived assertion  
(29) There exists a spinster who is married.  
a contradiction. Chomsky goes on to discuss whether selectional rules might be more appropriately included in the semantic component [11, pp. 153-160], and Katz accomplishes much the same effect with what he calls "semantic markers" in the semantic component. Katz [34, p. 26] then proceeds to define some clearly semantic categories such as the distinction between contradictory, synthetic, and analytic (tautologous) statements. For example,  

(30) Each rectangle is a triangle.  
(31) Each rectangle is a square.  
(32) Each rectangle is a quadrilateral.  
are contradictory, synthetic, and analytic respectively. Later, he even makes a further distinction between statements which are analytic versus statements which are "blatantly true", and correspondingly, statements which are contradictory versus statements which are "blatantly false". For example, an analytic statement might be  

(33) My father was a man.  
whereas a blatantly true statement might be  

(34) Rattlesnakes are poisonous.  

If we are to assume that the user of GRANIS tends to make statements which are not only non-vacuous, but also non-contradictory and non-analytic, and furthermore do not violate selectional rules, and hence entail contradictions, i.e., we assume he desires to explore synthetic relations
among objects which really exist in the universe of discourse, then we have additional criteria at our disposal for resolving ambiguity. How might GRANIS be extended to take advantage of these new criteria?

What is called for is a greater hierarchical structuring in the universe of discourse. At present the description lists are all on one level, i.e., a uniform set of description lists of tokens of objects which appear on the screen of the display. The ability to detect contradictions, as distinct from merely false synthetic statements, or to detect analytic statements, as distinct from merely true statements, or to incorporate selectional features may be easily achieved through description lists of "types" as well as "tokens".

To introduce description lists for types, each lexical item would have an associated description list. In particular, for selectional rules each list would contain the selectional features as attributes with the plus or minus as value. Note that in the general case more complex features would be needed, but these are easily included, since Formula Algol provides for a list of values associated with each attribute as an integral part of the language. For example, a part of the description list corresponding to the dictionary entry "triangle" might be

\[ \text{TRIANGLE} \right\{ [\text{SHAPE: EQUILATERAL, ISOSELES, SCALENE, EQUIANGULAR, ACUTE, RIGHT, OBTUSE}] \]

indicating seven possible shapes of triangles. Now an interpretation which gave rise to the selectional rule violation "circular triangle" such as in the ambiguous sentence

(35) Point to every object next to the triangle which is circular.

would be immediately rejected as anomalous, even though it was the natural interpretation in the sense described
earlier, and also "circular" was determined to be a conceptually possible value for the attribute shape. The selectional rule violation in this case can be easily determined from the description list of triangle by generating the value list of possible shapes and noting that "circular" is not one of them. The second interpretation of "circular object" would then be evaluated. This process is considerably more efficient than the current mode of operation which would exhaustively generate the universe of discourse testing each triangle on the screen for possible "circularity" before concluding that the interpretation was vacuous in the current universe of discourse, and only then going on to look for additional interpretations, if any.

What is proposed then is that first a set of global description lists of "types" be added to GRANIS, independent of any local description lists of "tokens" derived from the pictorial universe of discourse on the screen. These global lists should provide information on the intrinsic properties of each type, as well as on the range of possible values over each contingent property. Second, our strategy for resolving syntactic ambiguity then should be refined to first use the global lists to determine whether an interpretation is anomalous (not synthetic), and if not, then to go on to determine whether it is vacuous using the local lists, and proceeding as before. Otherwise, we reject this interpretation as out of hand and go on to the next, if any.

To speculate on the distinction between truth-value and meaning in terms of this extension, one may say that the truth-value of a sentence lies in the local lists, while the
meaning of a sentence resides in the global lists. These
global lists are highly structured, since most values of
attributes are themselves types with pointers to other
description lists in the set. To recast this list structure
in a slightly different perspective, if one were to represent
each type as a node in a directed graph where the edges of
the graph correspond to the pointers, one begins to approach
the kind of model of semantic memory organization described
by Quillian [61]. (To carry the analogy still further, the
graph may even be said to have "colored" edges to facilitate
the construction of legal transitive inferences.) The main
distinction with Quillian's work, however, is that this under-
lying semantic structure would be closely supervised by the
syntactic component.

Another interesting possibility for extending the
ambiguity resolving power of GRANIS is to incorporate the
basic heuristic of "means-ends" analysis. This is the same
technique which has proved successful over a wide class of
problems domains in the General Problem Solver (GPS) designed
by Newell, Shaw, and Simon (cf. [52]). The "given" is the ambiguous
input sentence; the "goal" is a meaningful (synthetic, non-
vacuous) interpretation within the universe of discourse;
and the "relevant operators" for reducing differences are the
productions (syntactic and semantic) applicable to the input
sentence. Considerable reprogramming would be needed, however,
before GPS techniques could be applied to advantage in the
current version of GRANIS.
4.2.3 Multiple Sentences.

Two major limitations on the present version of GRANIS are (i) it accepts sentences which concern only the current display, and thus it is impossible to refer to several different pictures in one statement, and more important (ii) all record of prior conversation with the system is destroyed once an input statement has been correctly processed. Of course, it is recognized that no great sophistication in language processing can be achieved by analysis of single sentences. To extend the system along this dimension, as a first approximation, a sequence of sentences could be analysed by taking the logical conjunction of their interpretations, but this solution overlooks some obvious difficulties. Chief among them is the discovery of antecedents of pronomial expressions which cross sentence boundaries. Such expressions have been called "anaphoric" expressions by Olney [55]. Generally, third-person pronouns and possessive adjectives, such as "he" and "his", are anaphoric expressions, as are many other kinds of pronouns. However, in English scientific and technical writing the most prevalent type of anaphoric expression is exemplified by the following sentences:

(36) Several experiments with college students were performed. The results were encouraging.

"The results" is anaphoric; it stands for an expression that does not actually occur in the text but can be constructed from one that does, namely, "several experiments with college students". Anaphoric expressions play a crucial role in promoting information transfer across sentence boundaries and between parts of complex sentences. In particular, they serve to abbreviate preceding passages, enabling writers
and speakers to add to what they have just said without repeating all of it. In English scientific and technical articles about 90 percent of the sentences contain at least one anaphoric expression [55].

Thus, it becomes immediately clear that no understanding of a conversation can be achieved on the basis of an independent analysis of constituent sentences. Indeed, the expressive power of natural language is achieved largely through the abbreviation of complex expressions by means of a small number of pronouns. English permits such abbreviation without loss of information by capitalizing on the inferential capability of the listener to resolve any ambiguity arising from potentially multiple antecedents associated with anaphoric expressions. The greater the inferential capacity of the listener, the greater parsimony achievable by the speaker without fear of being misunderstood.

The task of finding the correct antecedents for anaphoric expressions such as "The results" in (36) requires more sophisticated machinery than mechanically generating all possible antecedents in the universe of discourse to determine whether the truth-value of any particular substitution instance is true while others are false. The combinatorial explosion of possible substitutions becomes prohibitive even for small numbers of objects. Clearly, some heuristics are required both to (i) restrict the potential number of antecedents and (ii) order those remaining. For example, heuristics concerning case, gender, and number are obviously important to restrict the potential number of antecedents, whereas an ordering heuristic might be "give first consideration to nearest potential antecedents."
With the introduction of multiple sentence analysis more general kinds of inference making such as syllogisms, class inclusion, and transitivity can be considered. For example, given a triangle, a square, and a pentagon, the knowledge that the relation "to the left of" is transitive, and the sentences

(37) The triangle is to the left of the square.
(38) The square is to the left of the pentagon.
(39) The triangle is to the left of the pentagon.

then the third sentence is redundant, since it is implied by the first two. Such an inference is not possible in the current version of GRANIS. Based on Figure 7 GRANIS currently would mechanically evaluate the truth-value of each of the three sentences independently. In a sense the display serves as a direct model which can be used to test the consistency of all statements made about it. This observation, however, is not especially interesting by itself, since the model is fixed before hand by the user who draws the display. If on the other hand the system could accept imperative statements and modify the display during questioning by the user, much greater inferential capability would be possible. Such a capability entails the construction of a large number of primitive constraint operations for arbitrary displays and would require considerable programming.

4.3 Knowledge Acquisition.

In examining the natural language text which accompanied the various pictures of electrical circuits (Figures 8-12) and chemical molecules (Figures 13-18) as possible inputs to GRANIS, it was discovered that invariably the text
was inappropriate not merely because the constructions were too complicated for the grammar, but basically because the text served the purpose of supplementing the picture by describing that part of the subject matter which was not easily representable in two dimensions, rather than discussing relations of the various sub-parts of the picture. To illustrate this phenomenon, consider the following two passages, the first from a transistor manual [44, p. 134-135] which accompanied Figures 8 through 12, and the second from a college chemistry text [27. p. 504-505] which accompanied Figures 13 through 18.

In the first basic NOR circuit logic is performed by resistors. Any positive input produces an inverted output irrespective of the other inputs. The bias resistor gives temperature stability. The circuit design is straightforward. All logical operations can be performed with only this circuit. Many transistors readily meet the steady state requirements. The second NOR circuit is similar to the first except that capacitors are used to enhance switching speed. The capacitors increase the base current for fast collector current turn-on and minimize storage time by supplying a charge equal to the stored base charge. Thus, it is faster than the first NOR circuit at the expense of additional components and stringent stored charge requirements.

Phenol, benzaldehyde, benzoic acid, and aniline are important industrial derivatives of the benzene series of hydrocarbons. Phenol, commonly called carbolic acid, is a product of the distillation of coal tar. It is a colorless crystalline solid that slowly turns brown when it is exposed to the air. The compound is extremely poisonous. In dilute solution, it is used as a powerful antiseptic. Industrially it is used in the manufacture of explosives, synthetic resins such as bakelite, drugs, and dyes. At the present time, the demand for phenol is so great that synthetic methods of preparation account for better than 75 per cent of the phenol that is used.
Benzaldehyde, another coal-tar product, is used as the raw material for the preparation of many compounds used in industry. It is a liquid that boils at 180°C. It is sold as a perfume and as flavoring material under the name of oil of bitter almonds. Before the development of the coal-tar industry, the compound was obtained from bitter almonds.

Benzoic acid is a white crystalline solid that melts at 122°C. Most of the benzoic acid used in industry is prepared by synthetic methods. Aniline is a colorless liquid that boils at 184°C and also turns brown on continued exposure to the air. Most of the aniline used in industry is prepared from substances that are found in coal tar. Aniline is the starting material for aniline dyes, sometimes called coal-tar dyes. They are made in almost every conceivable shade of color.

As can be seen from the above passages, before GRANIS can be truly useful as a question-answering or information retrieval system, some provision must be made for acquiring knowledge about the universe of discourse other than by inspection of pictures. The main implication of this observation is that natural language text in the form of assertions about the universe of discourse must be used to guide either the construction of new description lists or the modification of existing description lists of objects displayed on the screen.

One method of performing this operation without disturbing either the natural language grammar or the expressions in the quantificational calculus output by the semantic productions is to generalize our conception of the evaluation process to include the construction and modification of description lists as well as computations to determine truth-values. That is to say, if the declarative input sentence is an assertion about the universe of discourse rather than a statement to be evaluated with respect to the
universe of discourse, then the predicates in the logical expressions should not be construed as boolean procedures, but rather as constructive procedures. In fact, it would be advisable to avoid any a priori assumption that an input statement is an assertion rather than an expression to be evaluated. Thus, each expression should be first evaluated in the conventional manner. To see precisely what this means, let us assume that the input is not anomalous. Then if upon evaluation it is true, the input is redundant with respect to the universe of discourse, and processing should terminate. If it is false then it is contradictory with respect to the universe of discourse, and the user should be informed of an inconsistency. Finally, if under all interpretations it is still vacuous, then it can be construed as an assertion about the universe of discourse, and this should signal the construction of description lists. This construction, as pointed out by Simon [75] in his work with the Heuristic Compiler, takes place in two stages. First, there is a selection stage which locates the place in memory where the new information is to be annexed. Second, there is an annexing stage which adds the new descriptive information to the memory structure. The virtue of this entire approach is that it prevents inconsistencies from arising in the data base, even though it may be used by many different people.

To see how this process might operate in practice, consider the third sentence in the passage on transistor logic.

(40) The bias resistor gives temperature stability.

1. This is not always clear even in human conversation, and sometimes gives rise to the "Are you asking me or telling me?" phenomenon.
The logical translation of (40) might be

\[(41) \ (\forall x)(\exists y) ([\text{Bias}(x) \land \text{Resistor}(x) \land \text{TemperatureStability}(y) \land \text{Gives}(x,y)]).\]

Upon evaluation with respect to Figure 1, (41) could then be determined to be a meaningful, synthetic, vacuous interpretation (there don't exist any objects on the screen which are temperatures!). Since (40) is unambiguous, the next step of constructive evaluation is evoked, and by selection and annexing the fifth description list of the set shown on page 64 would be augmented to

\[
\text{OB}[5] = [\text{TYPE: RESISTOR, BIAS}][\text{OHMS: 0}][\text{LINK: 4, 3}]
\]

[\text{GIVES: TEMPERATURE STABILITY}];

In this manner, the universe of discourse could be systematically expanded to contain information not easily represented in two dimensions.

Description lists might also be used to identify the input assertion for future reference. Williams [87] has suggested that the documentation of an input sentence include

\[
\langle \text{Sentence No.}\rangle = \langle \text{STATEMENT: input string}\rangle \langle \text{SPEAKER: name}\rangle \langle \text{PLACE: console no.}\rangle \langle \text{TIME: date}, \langle \text{hour}, \langle \text{minute}\rangle\rangle \rangle \langle \text{AMBIGUITY: no. of parsings}\rangle \langle \text{UNIVERSE OF DISCOURSE: display class}, \langle \text{display no.}\rangle\rangle;
\]

In this manner cross-referencing between sentences becomes possible.
4.4 Adaptive Properties.

What about the ability of the system to learn from its experience? In the first section of this chapter on habit-ability we saw how a limited kind of language learning might take place by means of paraphrastic transformational productions. No mention, however, was made of concept learning or modifying the semantic structure of the model by means of English sentences. This involves much more than the deductive kinds of inference making discussed in section 4.2. It involves "inductive" rather than "deductive" inference, and such problems of hypothesis formation lie far beyond the scope of the current system. For example, a typical inductive task might be to infer the concept of "square" from several instances of squares displayed on the screen, assuming that the concepts of polygonal figure, side, parallel, and equal in length were either primitive or already inferred from more primitive concepts. The work of T. Evans [24] with geometric analogy problems demonstrates some of the problems and progress in this connection.

A more ambitious goal is to attempt to anticipate what the user will say next by extrapolation of previous conversation together with a knowledge of the subject matter and the properties of the display. A continuous check can then be established between the system's hypothesis of what was expected and what was actually said by the user. In this fashion a measure of the credibility of the conversation is obtained which can be used by GRANIS as a basis for requesting clarification from the user on points of serious disparity.
In an interactive conversation where the user and the system are jointly forming hypotheses and making comparisons, a restriction to the present tense rapidly becomes impractical. The ability to use future and past tenses is therefore imperative. The semantics for various natural language tense constructions is highly complex, and complete analysis of English tense shows the logical existence of tenses that are not even distinguished by special names in the English grammar. For a full analysis of tense it is obviously necessary to distinguish between the time of the event (t_e) and the time of speaking or writing (t_s); but as Reichenbach [64, p. 287-98] has shown it is also necessary to recognize a third time, the time of reference (t_r). By positioning these times along a temporal continuum we can characterize standard English tenses as in Table 4.

It will be noted that more than one tense pattern is possible for future tenses giving us nine fundamental patterns in all. The terms "past", "present", and "future" are distinguished by the position of t_r relative to t_s. The terms "simple" and "perfect" are distinguished by the position of t_r relative to t_e. Reichenbach distinguishes additional tenses called posterior past and posterior future for which traditional names are nonexistent. The posterior past occurs when t_r precedes both t_e and t_s. The posterior future occurs when t_s < t_r < t_e. These tenses for which our language has no established forms are expressed by transcriptions. We say, for instance,

(42) I shall be going to see him.

and thus express the posterior future by speaking, not directly of the event at t_e, but of the act of preparation for it. In this fashion we can at least express the time order of events which closely succeed the point of reference.
<table>
<thead>
<tr>
<th>Traditional Tense Name</th>
<th>Tense Pattern</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Past</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a. Simple</td>
<td>( t_e = t_r &lt; t_s )</td>
<td>went</td>
</tr>
<tr>
<td>b. Perfect</td>
<td>( t_e &lt; t_r &lt; t_s )</td>
<td>had gone</td>
</tr>
<tr>
<td>2. Present</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a. Simple</td>
<td>( t_e = t_r = t_s )</td>
<td>goes (go)</td>
</tr>
<tr>
<td>b. Perfect</td>
<td>( t_e &lt; t_r = t_s )</td>
<td>has (have) gone</td>
</tr>
<tr>
<td>3. Future</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a. Simple</td>
<td>( t_s &lt; t_e = t_r )</td>
<td>will go</td>
</tr>
<tr>
<td></td>
<td>( t_r = t_s &lt; t_e )</td>
<td></td>
</tr>
<tr>
<td>b. Perfect</td>
<td>( t_s &lt; t_e &lt; t_r )</td>
<td>will have gone</td>
</tr>
<tr>
<td></td>
<td>( t_s = t_e &lt; t_r )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( t_e &lt; t_s &lt; t_r )</td>
<td></td>
</tr>
</tbody>
</table>

**TABLE 4 STRUCTURE OF ENGLISH TENSES**
Once the tense pattern of the sentence has been established, it may be added to the documentation of the sentence. The sentence may then be transformed into a canonical form which deletes all auxiliary verbs and replaces all non-auxiliary verbs with their infinitive form for later parsing. In this fashion no actual tense information is lost, while the kernel grammar may still be restricted to the present tense.

 Aside from the four dimensions along which GRANIS might be expanded which we have examined thus far, several other avenues for further development present themselves. For the sake of completeness some of these will be mentioned--

 1. A demonstration that the linguistic theory proposed applies to other natural languages aside from English.

 2. An exploration of voice input together with the accompanying pattern recognition problems. Such an extension would require the inclusion of a phonological component in our linguistic description.

 3. The development of a theory of deviation from well-formedness for each of the components in our linguistic description. Such a theory would account for why some deviations are understood by most listeners while others are not. In the syntactic component we would have to explain why a child who says "gooder" is really understood to mean "better"; in the semantic component why some sentences appear to be anomalous, but in reality are meaningful, and ultimately, why some metaphors are considered poetic while others are merely nonsensical; in the pragmatic component we would be required to investigate the attitudes and beliefs of the speaker together
with his memory and perceptual limitations. See Weinreich [91, p. 466] for an approach to a quantitative measure of deviance or Stolz [80] for current work on a psychological approach to deviance.

4. An exploration of natural language generation by GRANIS using an English generative grammar to drive answers to questions. Although the system at present can be said to understand certain English statements, it is itself rather inarticulate, having an extremely limited repertoire of rigidly formatted responses.

5. An extension to permit GRANIS to modify or augment the universe of discourse, i.e., to alter or add to the display which the user has drawn.

6. An exploration of the effect of providing the system with a goal structure which may or may not be in harmony with that of the user.

7. An exploration of the psychological implications of natural inference models. Note that no claim has been made in this thesis regarding the manner in which humans process natural language.

Although all of these directions just mentioned represent interesting and challenging avenues for further development, they are largely beyond the scope of the present investigation.

Of the original four directions, the most important direction is increasing the habitability of GRANIS, and thus it has the highest priority. Increasing the inferential power, knowledge acquisition, and adaptive properties of GRANIS depend to a large extent on future applications and the needs of individual users.
Chapter 5. CONCLUSION

The purpose of the research reported here was to investigate the processing of English and pictorial input on a computer. Particular emphasis was placed on deductive question answering using syntax directed techniques and the resolution of ambiguity in natural language. A theoretical framework for linguistic description was proposed as the basis for the design of a computer model. Its three components consisted of a production system, a dictionary, and a universe of discourse. Under certain conditions, as described, these components form a natural inference system which is both complete and consistent. The theory was only outlined, and much additional work remains to be done. However, in its present rough form, the theory served as a guide for the implementation of a computer model, GRANIS, which can interact with the user in a limited subset of English concerning restricted pictorial domains.

In this final chapter we shall summarize the accomplishments of this research, review some of the insights gained, the difficulties encountered, and comment on the practical and theoretical significance of GRANIS as a logic-based question-answering system.

Speaking first in a general way, GRANIS has demonstrated that the same syntax directed techniques which have proved successful in the construction of compilers can be
applied to advantage in the efficient processing of both natural language and pictures. Also, it has been demonstrated that a fairly modest extension of the natural language grammar to incorporate some elementary semantic processing is sufficient to resolve many kinds of lexical and syntactic ambiguity. Lexical ambiguity was largely avoided by restricting GRANIS to a specific universe of discourse at any particular time. Thus, each word in the terminal vocabulary was immediately provided with a unique meaning. Syntactic ambiguity was resolved by inspection of the individual objects described in the universe of discourse to determine which interpretation, if any, was meaningful. Thus, the concept of a universe of discourse as a set of description lists was an essential ingredient in the resolution of ambiguity, and a similar device is probably necessary in the construction of a semantic model for any domain. It was in this fashion that the fundamental problem with the Harvard Syntactic Analyser was avoided, and from which followed the conclusion that sentences containing resolvable ambiguity need not be ruled out as inadmissible inputs to a question-answering system. Indeed as was noted, resolvable ambiguity has the virtue of permitting greater parsimony by the speaker, since he is able to capitalize on the inferential capability of the system to recover correctly any information which was lost due to ambiguity in the manner which the speaker intended. Hence, contrary to a commonly held view, some ambiguity is desirable, i.e., that ambiguity which can be resolved by means of simple semantic processing. This is probably the major contribution of incorporating syntax and semantics within the framework of an integrated computer model for the purpose of processing conversational language.
In the introduction we adopted five criteria in the evaluation of other pictorial, question-answering systems. Let us now examine GRANIS in the light of these criteria.

1. **Syntactic**—although more sophisticated than the grammar in PLM, the GRANIS syntax is still far too restricted for comfortable use. The underlying theoretical issues were discussed in the previous chapter in the section on habitability. The principal practical constraint on the size of the grammar, and thus on the range of acceptable input formats, is the amount of computer memory available. The present version of GRANIS has already reached the maximum allowable size for a Formula Algol program on the G-21 computer. Thus, without a significant reorganization of the Formula Algol language to take advantage of additional core memory on the G-21, no extensions can be added directly to the current system. The programming extensions suggested in the preceding chapter, of course, are predicated on a machine with much larger memory facilities embedded within the framework of a genuine time-sharing system.

2. **Semantic**—the method of syntax directed interpretation in the construction of logical expressions has proved to be extremely general, and has even been applied to the construction of description lists based on the syntactic analysis of pictures.

3. **Deductive**—the predicate calculus has served as a useful tool not only in representing the deep structure of source statements, but also in making inferences about the universe of discourse. It has been learned, however, that attempting to adhere too rigidly to the notation of the
predicate calculus leads to undesirable restrictions on the semantic productions as well as the class of English sentences which can be easily translated into this notation. Thus, generalizing the notion of well formed formula, expanding the number of legitimate quantifier types, or increasing the kinds of logical connectives may lead to greater flexibility in making certain kinds of inferences which occur in natural language without sacrificing the significant inferential power of the original notation.

4. Interactive Ability-- the interactive ability of GRANIS depends strongly on the properties of the G-21 computer and the various underlying systems and monitors with which it must communicate. At present the response time of GRANIS ranges from about five seconds for simple sentences to about twenty seconds for the more complex or ambiguous sentences. The time to change pictures is in the order of ten seconds. This level of performance is barely adequate, and genuine conversation is frequently strained by the sluggishness of the system. Nevertheless, much insight into the problems of establishing a continuing dialog between man and machine was revealed in the course of developing a conversational system, and the additional effort in systems programming was judged worth while. One such insight was the crucial importance of recoverability in error conditions. Indeed, in retrospect the embodiment of GRANIS as a running computer program seems almost indispensable. Not only have program runs repeatedly revealed subtle errors in algorithms which appeared only in examples too complex to have been simulated by hand, but also, on occasion, they have provided valuable suggestions for improving the theoretical framework of GRANIS. One such suggestion was the use of vacuous interpretations as a basis for the resolution of syntactic ambiguity.
It has been observed empirically that the response time of GRANIS depends linearly on the number of objects displayed on the screen, linearly on the length of the source string, and exponentially on the depth of nesting of relative phrases. Although these kinds of dependencies are intrinsic to the problem domain, greater efficiency in the evaluation process could be achieved if the algorithms were modified to take greater advantage of the results of successive partial evaluations, since in some circumstances a partial evaluation is sufficient to guarantee a yes or no result regardless of the length of the remaining computation, which may then be dispensed with.

Of course it is realized that natural language communication with computers will come into its own only when supported by a system which permits voice communication with the computer. So long as a keyboard stands between man and machine, unpremeditated and informal programming is blocked, and the potential advantage of natural language cannot be wholly realized. Fortunately, projects designed to achieve man-machine voice communication are under way at a number of laboratories, and are sufficiently promising to encourage the parallel pursuit of natural language question-answering systems such as GRANIS.

5. Growth Potential-- the approach taken in GRANIS of implementing the grammar in terms of a set of syntactic and semantic productions has considerable growth potential for two reasons. First, the productions provide a uniform notational framework for extending both the syntactic and semantic components of the system, while simultaneously
maintaining a well defined interface between them. Second, the productions inherit a general property of productions in that either an alteration or an increment to the productions will generally affect only the local cluster in which the change or addition was made. That is, a single alteration does not have the unfortunate effect of propagating through the entire system, requiring an unknown number of global changes to preserve consistency as do some systems which do not enjoy this property. As mentioned before, the principal constraint which inhibits the growth and also conditions the performance of GRANIS more than any other single factor is the limited core memory available on the G-21. This factor is far more important at present than the slow processing speed of the G-21, and would probably continue as the dominant constraint if GRANIS were to be implemented on one of the larger and faster third generation computers now becoming available.

To conclude our evaluation, we have seen that GRANIS represents progress beyond both PLM and NAMER along each of the five dimensions adopted for evaluation. But it should be reemphasized that the fundamental, functional improvement of GRANIS over its predecessors is its unique ability to resolve syntactic ambiguity in a conversational mode while the man and the machine jointly converge on a mutually acceptable form of an input question.

Concerning the practical significance of GRANIS, certain applications in the field of electrical circuits and chemistry can be derived by means of an extension of the results described in section 3.2. However, in the introduction
two possible applications of pictorial, natural language question-answering systems were proposed, viz- pictorial information retrieval and automated teaching. What can be said concerning the use of GRANIS as it now stands in either of these applications? Speaking in the area of information retrieval over a large pictorial data base, the current system would require some conceptual modifications in the structuring of the description lists which form the universe of discourse. A hierarchical structuring would be necessary to permit more efficient indexing through the description lists. In automated teaching, several high school plane geometry texts were examined as possible inputs to GRANIS. This hope was abandoned when it became apparent that, although the material was coherently organized and fairly self-contained, the required grammatical sophistication was well beyond the limitations of the current system. In searching for more elementary treatments of plane geometry as a programmed text in the hope that at some point the grammatical constructions would be sufficiently simple to be acceptable to GRANIS, it was observed that increasing appeal is made to the extraordinary "anthropomorphic" data base available to students at that level. Requirements for general knowledge about the world such as "libraries are collections of books" or "screw drivers are used to screw in screws" abound in such presentations (for example, see Ranucci [62], "Points, Lines, and Planes"), and preclude a large percentage of sentences from semantic analysis simply because the lexical definitions are not primitive to the system. Any particular sentence, however, can in principle be understood once the implicit information has been made explicit. Thus, even without extensive reprogramming, the present system could be made to handle most statements about simple geometric pictures, providing there was not too much information tacitly assumed about the real world.
From the linguistic point of view an important theoretical contribution of GRANIS is to the notion of "understanding" in computer systems. Generalizing the operational definition of understanding given by Bobrow [7, p. 7] we say that a computer understands a fragment of natural language if it accepts input sentences which are members of this fragment, and answers questions based on information contained in a common universe of discourse. GRANIS can be said to understand in this sense. The universe of discourse in the case of GRANIS is a picture which exhibits the relations between the various objects displayed, while the objects in the universe of discourse provide an immediate interpretation of each of the individual lexical items in the referent terminal vocabulary. Although Raphael has said that a system like SIR can be said to "understand" for the class of relations with which it deals [63, p. 9], its "understanding" is in fact restricted to the function words in the terminal vocabulary, i.e., it "understands" the relationships between the referent terms without having any knowledge of the meaning of the referent words themselves. This policy can lead to the generation of spurious inferences as was pointed out in section 2.3. In GRANIS when one says "The triangle is to the left of the square" not only is the relation "to the left of" understood, but also the terms "triangle" and "square". That is, GRANIS really "knows" which objects are triangles and which objects are not. This is because the basic concepts of "polygon" and "number of sides", and so on are primitive to the system, and ultimately, it is this elementary notion together with the universe of discourse which permit resolution of syntactic ambiguity. Systems like SIR are in principle not capable of such disambiguation without
requiring some fundamental semantic reorganization.

Another theoretical contribution of GRANIS might be to the field of psychology. In some sense the computer model constructed could be considered a simulation of human processing of linguistic and pictorial information. Of course, psychological experiments would have to be devised to test precisely the similarity between the behavior of the model and human behavior, but in this manner we might obtain useful suggestions for improving GRANIS and at the same time gain valuable insight into human cognitive processes.

To conclude, in 1959 Edmund C. Berkeley said,

Of all the territories of application of computers to human affairs, one that is very likely to have most far-reaching effects is the territory in which computers converse and discuss with human beings, using ordinary language, and handling ideas appropriately. This is a future development which clearly casts its shadow before.  

Although most would agree with this statement in spirit, it is recognized that we are still far from writing a program which can understand all, or even a very large segment of English. However, within its narrow field of competence, GRANIS does serve as a demonstration that computer understanding of limited pictorial and text information is possible. Furthermore, it shows that using syntax directed interpretation, one could build a system which would communicate well with people both in English and pictures. Indeed, it is my conviction that ultimate progress in the field of artificial intelligence will depend strongly on the use of on-line graphic display devices and natural language communication. Natural language provides the convenience which permits the user to concentrate on the

problem at hand, rather than on a translation into a formal and unambiguous language, while the displays quite literally provide a new dimension in man-machine interaction. Finally, their on-line property provides the necessary responsiveness for genuine conversation.
APPENDIX I. SPECIFICATION OF FORMULA ALGOL DISPLAY PROCEDURES

The interface between Formula Algol and the graphic display units consists of a set of machine-language dependent procedures, based on primitive scope monitor routines specified in Coles [15]. Such procedures appear at various points within the program of Appendix 2, but are not defined within the program itself. They are briefly specified as follows:

1. **B(BNO,B1,B2,B3,B4,B5)** is a fundamental interface procedure used to pass parameters to scope monitor routines and perform administrative tasks such as memory protection and external interrupt control.

2. **READSCOPE** accepts text input data from the display and passes it to the program for immediate processing. It first enables the keyboard, appropriately positions the cursor on the screen for input, and then returns control to the scope monitor while the user is typing. An interrupt is generated upon receipt of special characters, such as period, question mark, or exclamation point, and all preceding text is then passed for processing.

3. **MOVEPIX** accepts a drawing on the display and passes it as a logic array to the picture scanner.

4. **DISPLAYON** sets a display switch to permit display of all text and identifiers with the scope of any **PRINT** statements which follow.

5. **DISPLAYOFF** returns the display switch to its normal setting.
6. **DISPLAY(TEXT)** turns display on, prints TEXT, a formula variable, and turns display off.

7. **CLEAR(PAGE)** clears one of the four pages on the scope.
APPENDIX 2. PROGRAM LISTING

THE ABOVE IS A TYPICAL BACKUS NAUR FORM (BNF) GRAMMAR FOR SIMPLE ENGLISH SENTENCES PERTAINING TO PLANE GEOMETRY.

LINE 1 SYNTAX TERMINALS DECLARED
SYMBOL EACH, A, POLYGON, PENTAGON, SQUARE, TRIANGLE, THAN, SMALLER, GREATER, BLACK, WHITE, ISA, ISNOTA, PERIOD, CIRCLE, BIG, WHICH, ISNOTTHE, LEFT;

LINE 1 INTERNAL IDENTIFIERS DECLARED
COMMENT S IS THE SYNTACTIC STACK WHILE T IS THE SEMANTIC STACK;
SYMBOL INSTRING, TYPE, COLOR, SIZE, XCO, YCO, EXSTRING, S, VOCAB, SIGMA, FAULT;
INTEGER I, J, K, N, ORN, WDNO, SIDES; EPSIL;
REAL SR24, SL24, SR16, SL16, SR10, SR8, SL8, AREA;
FORM T, TP, LALPHA, I, BETA, VB, V1, L1, ALPHA, L1BETA;
BOOLEAN KEY, QKEY, NONE, BLK;
LOGIC LWD, MASK;

LINE 1 SEMANTIC TERMINALS DECLARED
FORM ARRAY 0(0:10); SYMBOL ARRAY 0B(0:10);
FORM PROCEDURE PENTAGONF(X); VALUE X; FORM X;
BEGIN IF KEY THEN PENTAGONF AMONG(PENTAGON, TYPE(COSYMB(X))); ELSE PENTAGONF PENTAGONF.(X) END;
FORM PROCEDURE SQUAREF(X); VALUE X; FORM X;
BEGIN IF KEY THEN SQUAREF AMONG(SQUARE, TYPE(COSYMB(X))); ELSE SQUAREF SQUAREF.(X) END;
FORM PROCEDURE TRIANGLEF(X); VALUE X; FORM X;
BEGIN IF KEY THEN TRIANGLEF AMONG(TRIANGLE, TYPE(COSYMB(X))); ELSE TRIANGLEF TRIANGLEF.(X) END;
FORM PROCEDURE BLACKF(X); VALUE X; FORM X;
BEGIN IF KEY THEN BLACKF (COLOR(COSYMB(X))=BLACK); ELSE BLACKF BLACKF.(X) END;
FORM PROCEDURE WHITEF(X); VALUE X; FORM X;
BEGIN IF KEY THEN WHITEF (COLOR(COSYMB(X))=WHITE); ELSE WHITEF WHITEF.(X) END;
FORM PROCEDURE BIGF(X); VALUE X; FORM X;
BEGIN IF KEY THEN BIGF SIZE(COSYMB(X))>50000 ELSE BIGF BIGF.(X) END;
FORM PROCEDURE SMALLERF(X,Y); VALUE X,Y; FORM X,Y;
BEGIN IF KEY THEN BEGIN REAL A,B;
A= SIZE(COSYMB(X)); B= SIZE(COSYMB(Y)); SMALLER= A>B END ELSE SMALLER= SMALLERF.(X,Y) END;
FORM PROCEDURE GREATERF(X,Y); VALUE X,Y; FORM X,Y;
BEGIN IF KEY THEN BEGIN REAL A,B;
A= XCO(COSYMB(X)); B= XCO(COSYMB(Y)); GREATER= (A>B) END ELSE GREATER= GREATERF.(X,Y) END;
FORM PROCEDURE LEFTF(X,Y); VALUE X,Y; FORM X,Y;
BEGIN IF KEY THEN BEGIN INTEGER A,B;
A= XCO(COSYMB(X)); B= XCO(COSYMB(Y)); LEFTF= -(A>B) END ELSE LEFTF= LEFTF.(X,Y) END;
FORM PROCEDURE ALLF(X,SCOPE); VALUE X,SCOPE; FORM X,SCOPE;
BEGIN IF QKEY THEN BEGIN INTEGER I; FORM INTER;
INTER= EVAL(X)SCOPE(0,11)); FOR I= 1 STEP 1 UNTIL OBNO DO ALLF= ALLF INTER= INTER= EVAL(X)SCOPE(0,11)); END ELSE BEGIN SCOPE= -(LALPHA)-LBETA; ALLF= ALLF.(X,SCOPE) END END;
FORM PROCEDURE EXIF(X,SCOPE); VALUE X,SCOPE; FORM X,SCOPE;
BEGIN IF QKEY THEN BEGIN INTEGER I; FORM INTER;
INTER= EVAL(X)SCOPE(0,11)); FOR I= 1 STEP 1 UNTIL OBNO DO EXIF= EXIF INTER= INTER= EVAL(X)SCOPE(0,11)); END ELSE BEGIN SCOPE= LALPHA-LBETA; EXIF= EXIF.(X,SCOPE) END END;
LOGIC PROCEDURE P(L1,L2,L3,L4); VALUE L1,L2,L3,L4; LOGIC L1X,L2,L3,L4;
P= (L1*SL8+L2)*SL8+L3+SLA+L4;
INTEGER PROCEDURE SUM(I,J,N,F); VALUE J,N; INTEGER I,J,N,F;
BEGIN INTEGER TEMP,X; TEMP= 0;
FOR K= J STEP 1 UNTIL N DO BEGIN I= K; SUM= TEMP= TEMP + F END END;
INTEGER PROCEDURE DX(V); VALUE V; LOGIC V;
BEGIN INTEGER II; II= DX= V*AR1777; IF (V=BL002)=BL002 THEN DX= -1 END;
INTEGER PROCEDURE DY(V); VALUE V; LOGIC V;
BEGIN INTEGER II; II= DY= (V*SR10)+AR1777; IF (V=BL004)=AL004 THEN DY= -1 END;
REAL PROCEDURE LNGTH(V); VALUE V; LOGIC V;
LUVGTH = SORT(DX(V) * DX(V) + DY(V) * DY(V));  
BOOLEAN PROCEDURE EQUAL(S1, S2); VALUE S1, S2;  
   PROPERTY LNUGTH(S1) = LNUGTH(S2);  
LOGICAL PROCEDURE SIDE(I, OBJ); VALUE I, OBJ; INTEGER I; FORM OBJ;  
BEGIN INTEGER K, I;  
IF OBJ <= 0 THEN BEGIN K = -1;  
   FOR J = 0 STEP 1 WHILE K = 0 DO IF ((HIJ - BL377) = BL377) THEN BEGIN  
      IF J = 1 THEN BEGIN PRINT(, NOTASIDE);  
      ENDGO TO OUTSI END;  
ELSE K = K+1;  
END  
ELSE BEGIN  
OUTSI; PRINT(, NOTASIDE); S1 = BL12004 END;  
BOOLEAN PROCEDURE POLYGONP(OBJ); VALUE OBJ; FORM OBJ;  
BEGIN INTEGER I, J; BOOLEAN POL; INTEGER S; OPA = 0; POL = BLK = FALSE;  
IF (SIDE(I, OBJ) = BL12004) THEN BEGIN I = I+1 GO TO Q END;  
ELSE S = 1;  
POLYGONP = POL; S7IDES = 0A(ABS(SUM(J, S, DX(SIDE(J, OBJ)))) * EPSIL1) =  
   (ABS(SUM(J, S, DY(SIDE(J, OBJ)))) * EPSIL1);  
IF S7IDES > 10 THEN BEGIN S7IDES = 10; BLK = TRUE; END;  
IF POL THEN AREA = 5A(ABS(SUM(J, S, S2IDES - 2,  
   DX(SIDE(I, OBJ)) * SUM(J, S, DX(SIDE(J, OBJ))))) +  
   DX(SIDE(I, OBJ)) * SUM(J, S, DX(SIDE(J, OBJ)))));  
END;

LINE 1  
SYNTACTIC NON-TERMINALS DECLARED  
SYMBOL SENT, SUBJ, PRED, NOUNPHR, RELPHR, ART, NOUN, ADJ, ADJS, INTRO,  
COMPARADJ, REL, LOC, RELPHR,  
LINE 1  
SYNTACTIC CLASS IDENTIFIERS DECLARED  
SYMBOL ART, ADJ, NOUN, COMPARADJ,  
LINE 1  
SEMANTIC NON-TERMINALS DECLARED  
SYMBOL O, 01, INTRO, NOUNF, ADJF, ADJSF, LOCF, RELPHRZF, NOUNPHRZ;  
FORM WFF, X, Y, VAR, I, VAR, RVAR, QUANT, SNOUN, SADJ, SNOUNPHRZ, SINTR,  
SRELPHRZ, SSRJ, SREP, LANY, SADJS, SLOC, SRELPHRZ;  
INTEGER RPAG, BOOLEAN INTRP?1 SYMBOL Z, MSJ;  
LINE 2  
CLASS DEFINITIONS FOR SYNTAX ANALYSER  
LET {ARTC} = [ZA, (A, B, C)];  
LET {ADJC} = [ZB, (B, C, D)];  
LET {NOUNC} = [ZC, (POLYGON, PENTAGON, SQUARE, TRIANGLE)];  
LET {COMPARADJC} = [ZD, (SMALLER, GREATER)];  
SN  
INITIALIZATION OF SEMANTIC PATTERNS  
MS-ALLF, (LVARFORM, RVARFORM); XEF, (LVAR, RVAR);  
INTRO- (SMALLERF, LVARFORM, RVARFORM), GREATERF, (LVARFORM, RVARFORM);  
NOUNF- (POLYGON, (VARFORM), PENTAagonF, (VARFORM), SQUAREF, (VARFORM),  
   TRIANGLEF, (VARFORM));  
NOUNPHRZF- OF(OF(NOUNF), OF(ADJSF), OF(NOUNF));  
ADJF- BLACKF, (FORM), WHITEF, (FORM), HIGF, (FORM));  
ADJSF- [OF(ADJF), OF(ADJF), OF(ADJSF)];  
RELPHRZF- [OF(O1), OF(RELPHRZF), OF(O1)];  
SN  
SN  
SN
LOC= (LEFT,

Q= [ALL,

Q1= [ALL,

LINE 1

MASK= 8F377S8F4= 8F649;8F61--10;8F62--10;8F64--10;8F64--10;

VOCAB= [TRIANGLE, X, SQUARE, X, X, POLYGON, X, X, EACH, X, BLACK, X, X,

N= 501

FOR LWD= P(8R23, 8R24, 8R17, 8R20), 0,

LINE 4 START UNIVERSE OF DISCOURSE... OMEGA

PIX= MOVEPIX= CLEAR(3), CLEAR(4),

FOR I= 50 STEP -1 UNTIL 45 DO PRINT(H(I)); N= 0BNO= 9;

FOR I= 0 STEP 1 UNTIL OBN0 DO BEGIN

IF POLYGON(O(I)) THEN BEGIN THE TYPE OF O(I) IS POLYGON;

IF BLK THEN THE COLOR OF O(I) IS BLACK

ELSE THE COLOR OF O(I) IS WHITE;

THE SIZE OF O(I) IS AREA;

IF SIDES=3 THEN THE TYPE OF O(I) IS ALSO TRIANGLE;

IF SIDES=4 THEN THE TYPE OF O(I) IS ALSO SQUARE;

IF SIDES=5 THEN THE TYPE OF O(I) IS ALSO PENTAGON END;

IF SIDES=0 THEN OBN0= 1-1 END;

YPT= 270; XPT= 0; DISPLAYON;

FOR I= 0 STEP 1 UNTIL OBN0 DO PRINT(OBI(I));

DISPLAYOFF; INTRP= FALSE.

DRAWING TOLERANCE AND SCOPE NUMBER IDENTIFICATION

EPSIL = 10; IF B(-1, 0, 0, 0) THEN BEGIN PRINT(ERROR, 3.6) GO TO OUT END;
IN
LINE 1  INITIAlIZATION OF INSTRING
MORE T= WFFJ KEY= DKEY= FALSE) READCOLOR 0; 0414
IF INTPR2 THEN GO TO Q41
YPT= 1020; XPT= 0; WDN0= 1; INSTRING= (PERIOD);
DISPLAY(. INPUT= . SENTENCE= READSCOPE;
COMMENT: UNPACK R TO ONE CHARACTER PER WORD; 0417
FOR I= 50 STEP -1 UNTIL 1 DO BEGIN
D[4*I+1]= (R[I]*SR24)*MASK; D[4*I+1-1]= (R[I]*SR16)*MASK;
D[4*I-2]= (R[I]*SR8)*MASK; D[4*I-3]= R[I]*MASK END;
DONE= FALSE; I= 200;
Q1: COMMENT: PUT A WORD FRM ARRAY D INTO ARRAY L;
FOR J= 16 STEP -1 WHILE -(D[I+J-16]=0) DO BEGIN
M[IJ]= D[I+J-16]; IF M[IJ]=9R5 THEN BEGIN
M[IJ]= 0; DONE= TRUE END END;
I= I+J-17; IF I<1 THEN BEGIN DISPLAY(. SENTENCE= . INT=. INN); GO TO MORE END;
IF J=16 THEN GO TO Q1;
FOR J= J STEP -1 UNTIL 1 DO M[IJ]= 0;
FOR K= 4 STEP -1 UNTIL 1 DO
L[K]= P(M[I+K], M[I+K-1], M[I+K-2], M[I+K-3]);
N= 50;
Q2: COMMENT: SEE IF WORD IN ARRAY L IS AMONG WORDS ON VOCAB;
FOR K= 4 STEP -1 UNTIL 1 DO BEGIN
IF U[N]=K-1=0 THEN BEGIN
IF (K>1 A= (L[K-1]=0)) THEN GO TO Q3;
IF N=50 THEN BEGIN CLEAR(1); CLEAR(4); DISPLAY(. BYE); 0440
30 TO OUT END;
IF N=9 THEN GO TO PIV;
IF N=7 THEN BEGIN PAUSE TRUE; CLEAR(4); CLEAR(1); 0443
DISPLAY(. PROGRAM= . PAUSED); GO TO TOP END;
INSTRING= (INSTRING, (49-N)TH OF VOCAB;
IF DONE THEN BEGIN INSTRING= (INSTRING, PERIOD); GO TO Q4 END
ELSE BEGIN WDNO= WDN0+1; GO TO Q1 END END END;
ELSE BEGIN
Q3: N= N-1; IF N<1 THEN BEGIN XPT= 200;
DISPLAY(. WORD= NUMBER= WDN0= . INT=. INN=. VOCABULARY); GO TO MORE END;
IF U[N]=0 THEN BEGIN N= N-1; GO TO Q2 END
ELSE GO TO Q3 END END;
Q4: EXSTRING= (INSTRING); DELETE 1ST OF INSTRING; CLEAR(3); YPT= 270;
DISPLAYON; PRINT(INSTRING); DISPLAYOFF; S= NEXT; 0453
START SYNTAX ANALYSER

SENT0: IF $=={S,EACH}$ THEN BEGIN ALTER LAST OF S TO [ART,NEXT];
    T = T + ALP.(X,-ALPHA,(X)\*BETA,(X)) GO TO ART1 END;
    IF $=={S,ISA,A}$ THEN BEGIN ALTER LAST OF S TO [ART,NEXT];
    GO TO ART1 END;
    IF $=={S,A1}$ THEN BEGIN ALTER LAST OF S TO [ART,NEXT];
    T = T + EXP.(X, ALPHA.(X)\*BETA,(X)) GO TO ART1 END;
    ELSE PRINT(ERROR,SENT,0); GO TO SORRY;

SENT1: IF $=={S,SENT,PERIOD}$ THEN BEGIN DELETE LAST 2 OF S;
    GO TO SUCCESS END;
    ELSE PRINT(ERROR,SENT,1); GO TO SORRY;

SUBJ1: SIGMA\* ,SIGMA;
    IF $=={S,INTRO, SUBJECT,SIGMA,S1}$ THEN BEGIN ALTER LAST 3 OF S TO
    [RELPHRZ,SIGMA];
    IF T == (LANY\*ANY + INTRO\*OF(INTRO) + SSUBJ1OF(S1)) THEN
    BEGIN SSUBJ1 = SUBS(X)\*SSUBJ(RVAR);
    LALPHA = SUBS(X)\*LALPHA(RVAR);
    L\*BETA = SIN(1);
    T = LANY\* EVAL SSUBJ END;
    ELSE PRINT(FAULT,TT,SUBJ,1); GO TO RELPHRZ1 END;
    IF $=={S,SUBJ,ISA}$ THEN BEGIN INSERT [NEXT] AFTER
    LAST OF SJ GO TO SENT0 END;
    IF $=={S,SUBJ,ISNOTA}$ THEN BEGIN INSERT [NEXT] AFTER
    LAST OF SJ GO TO SENT0 END;
    ELSE PRINT(ERROR,SUBJ,1); GO TO SORRY;

RELPHRZ1: SIGMA\* ,SIGMA;
    IF INTR2 THEN GO TO NP2;
    COMMENT: THE INTERNAL LABELS IN THIS CLUSTER EFFECTIVELY INTERCHANGE
    THE ORDER OF PRODUCTIONS DURING THE SECOND INTERPRETATION IF ANY;
    IF $=={S,NO NPHRZ, (COMPARADJC1)}$ THEN GO TO RELPHRZ0;
    IF $=={S,NO NPHRZ, WHICH}$ THEN GO TO RELPHRZ0;
    IF INTR2 THEN GO TO NP4;

RELPHRZ0: IF $=={S,INTRO, NO NPHRZ,SIGMA,S1}$ THEN BEGIN ALTER
    LAST OF S TO [SUBJ,SIGMA]; GO TO SUBJ1 END;
    IF INTR2 THEN GO TO NP3;
    IF $=={S,NO NPHRZ,SIGMA,S1}$ THEN BEGIN ALTER LAST OF S TO
    [SUBJ,SIGMA]; GO TO SUBJ1 END;
    ELSE PRINT(ERROR,NO NPHRZ,1); GO TO SORRY;

ART1: IF $=={S,ART,(INU NCI)}$ THEN GO TO NOUN0;
    IF $=={S,ART,(IADJC1)}$ THEN GO TO ADJC;
    ELSE PRINT(ERROR,ART,1); GO TO SORRY;

NOUN0: IF $=={S,POLYGON}$ THEN BEGIN ALTER LAST OF S TO [NO NUB, NEXT];
    T = T + POLYGON(X); GO TO NOUN1 END;
    IF $=={S,PENTAGON}$ THEN BEGIN ALTER LAST OF S TO [NOUN, NEXT];
    T = T + PENTAGON(X); GO TO NOUN1 END;
    IF $=={S,SQUARE}$ THEN BEGIN ALTER LAST OF S TO [NOUN, NEXT];
    T = T + SQUAREF(X); GO TO NOUN1 END;
    IF $=={S,TRIANGLE}$ THEN BEGIN ALTER LAST OF S TO [NOUN, NEXT];
    T = T + TRIPLEF(X); GO TO NOUN1 END;
    ELSE PRINT(ERROR,NO NUB,0); GO TO SORRY;

NOUN1: SIGMA\* ,SIGMA;
    IF $=={S,ART,NOUN,SIGMA,S1}$ THEN BEGIN ALTER LAST 4 OF S TO
    [PRED,SIGMA]; G3 TO PRED1 END;
    IF $=={S,ISA,ART,ADJS,NO NUB,SIGMA,S1}$ THEN BEGIN ALTER
    LAST 5 OF S TO [PRED,SIGMA];
    IF T == (LANY\*ANY + SADJS1OF(ADJSF) + SNO NUBOF(NO NUB)) THEN

143.
T←LANY+(SADJS\*NOUN) ELSE
PRINT(FAULT,NOUN1,ISA,ART,ADJS,NOUN) GO TO PRED1 END;

IF S==($) AND NOUN;SIGMA\*S1) THEN BEGIN ALTER LAST 3 OF S TO
[NOUNPHRZ,SIGMA]
ELSE PRINT(FAULT,NOUN1,ART,NOUN) GO TO NOUNPHRZ1 END;

IF T==($LANY;ANY + QUANTOF(Q) + S NOUNOF(NOUN)) THEN BEGIN
LALPHA\*SNOUN; T←LANY + EVAL(X)QUANT(VAR) END
ELSE PRINT(FAULT,NOUN1,ART,NOUN) GO TO NOUNPHRZ1 END;

IF S==($) AND ADJS,NOUN,SIGMA\*S1) THEN BEGIN ALTER LAST 4 OF S TO
[NOUNPHRZ,SIGMA]
ELSE PRINT(FAULT,NOUN1,ART,NOUN) GO TO NOUNPHRZ1 END;

IF T==($LANY;ANY + QUANTOF(Q) + SADJS;OF(ADJSF) + S NOUNOF(NOUN))
THEN BEGIN LALPHA\*SADJS;NOUN
T←LANY + EVAL(X)QUANT(VAR) END
ELSE PRINT(FAULT,NOUN1,ART,NOUN) GO TO NOUNPHRZ1 END;

IF S==($) AND NOUN;SIGMA\*S1) THEN BEGIN ALTER LAST 3 OF S TO
PRED2;SIGMA]
ELSE PRINT(FAULT,NOUN1,ART,NOUN) GO TO PRED1 END
ELSE PRINT(ERROR,NOUN1) GO TO SORRY1

ELPHRZ0: RPCODE←RPCODE + 1
IF S==($) AND SMALLER) THEN BEGIN ALTER LAST OF S TO
[COMPARADJ,next];
T←T + SMALLER.(X,Y) GO TO COMPARADJ1 END

IF S==($) AND GREATER) THEN BEGIN ALTER LAST OF S TO
[COMPARADJ,next];
T←T + GREATER.(X,Y) GO TO COMPARADJ1 END

IF S==($) AND WHICH) THEN BEGIN ALTER LAST OF S TO
[REL1;NEXT];
GO TO REL1 END
ELSE PRINT(ERROR,RELPHRZ,0) GO TO SORRY1

ELPHRZ1: SIGMA←SIGMA1
IF S==($) AND RELPHRZ.RELPHRZ;SIGMA\*S1) THEN BEGIN ALTER LAST
3 OF S TO [RELPHRZ,SIGMA]
IF T==($LANY;ANY + SRELPHRZ;OF(RELPHRZ;SIGMA) + SRELPHRZ;OF(Q))
THEN T←LANY + (SRELPHRZ;SRELPHRZ)
ELSE PRINT(FAULT,RELPHRZ1,RELPHRZ,SRELPHRZ) GO TO RELPHRZ1 END

IF S==($) AND RELPHRZ;SIGMA\*S1) THEN BEGIN ALTER LAST 2 OF S TO
(RELPHRZ;SIGMA1) GO TO RELPHRZ1 END
ELSE PRINT(ERROR,RELPHRZ,1) GO TO SORRY1

RELPHRZ1: SIGMA←SIGMA1
IF S==($) AND RELPHRZ;ICOMPARADJ1) THEN GO TO RELPHRZ01
IF S==($) AND RELPHRZ;WHICH) THEN GO TO RELPHRZ01
IF S==($) AND NOUNPHRZ,RELPHRZ,SIGMA\*S1) THEN BEGIN ALTER LAST
3 OF S TO [SUBJ,SIGMA]
IF T==($LANY;ANY + SADJS;OF(Q) + SRELPHRZ;OF(RELPHRZ;SIGMA)
THEN BEGIN _BETA=BETA.(X)
LALPHA; LALPHA;SRELPHRZ1
T←LANY + EVAL SADJS;NOUNPHRZ END
ELSE PRINT(FAULT,RELPHRZ1,RELPHRZ,1) GO TO SORRY1

REL1: IF S==($) AND (ISONTHE) THEN BEGIN INSERT [NEXT] AFTER LAST OF S
GO TO LOC0 END

IF S==($) AND ISA) THEN BEGIN INSERT [NEXT] AFTER LAST
OF S; GO TO ADJ0 END
ELSE PRINT(ERROR,REL,1) GO TO SORRY1

LOC0: IF S==($) AND (LEFT) THEN BEGIN ALTER LAST OF S TO [LOC,NEXT];
T←T + LEFTF.(X,Y) GO TO LOC1 END
ELSE PRINT(ERROR,LOC,0) GO TO SORRY1
LDC1: SIGMA = .SIGMA1
    IF S = [S, REL, SONTHE, LOC, SIGMA$1] THEN BEGIN ALTER LAST 4 OF S TO [REL, PHRZ, SIGMA1]
    IF T = (ANY + SLOC + (LOCF)) THEN T = LANY + ALLF.(RVAR, SLOC)
    ELSE PRINT(FAULT, LOC, 1); GO TO RELPHRZ1 END
    ELSE PRINT(ERROR, LOC, 1); GO TO SORRY
INTRO1: IF S = [S, INTRO, (INTACT)] THEN GO TO SENT1
    ELSE PRINT(ERROR, INTRO, 1); GO TO SORRY
COMPARADJ1: IF S = [S, COMPARADJ, THEN] THEN BEGIN ALTER LAST 2 OF S TO
    [INTRO, NEXT]; GO TO INTRO1 END
    ELSE PRINT(ERROR, COMPARADJ, 1); GO TO SORRY
ADJ1: IF S = [S, BLACK] THEN BEGIN ALTER LAST OF S TO [ADJ, NEXT];
    T = T + BLACKF.(X); GO TO ADJ1 END
    IF S = [S, BIG] THEN BEGIN ALTER LAST OF S TO [ADJ, NEXT];
    T = T + BIGF.(X); GO TO ADJ1 END
    ELSE PRINT(ERROR, ADJ, 0); GO TO SORRY
ADJS1: IF S = [S, ADJS, ADJ, SIGMA$1] THEN BEGIN ALTER LAST 3 OF S TO
    [ADJS, SIGMA1]
    IF T = (ANY + SADJS + OF(ADJSF) + SADJS + OF(ADJF)) THEN
    T = LANY + (SADJS + ADJ) ELSE
    PRINT(FAULT, ADJ, 1, ADJS, ADJS1) GO TO ADJ91 END
    IF S = [S, REL, ISA, ADJ, SIGMA$1] THEN BEGIN ALTER LAST 4 OF S TO [REL, PHRZ, SIGMA1]
    GO TO RELPHRZ1 END
    IF S = [S, ADJ, SIGMA$1] THEN BEGIN ALTER LAST 2 OF S TO [ADJS, SIGMA1] GO TO ADJ1 END
    ELSE PRINT(ERROR, ADJ, 1); GO TO SORRY
ADJS1: IF S = [S, ADJS, (ADJCI)] THEN GO TO ADJ1
    IF S = [S, ART, ADJS, (INOJNCI)] THEN GO TO NOUNO
    ELSE PRINT(ERROR, ADJS, 1); GO TO SORRY
PRED1: SIGMA = .SIGMA1
    IF S = [S, SUBJ, PRED, SIGMA$1] THEN BEGIN ALTER LAST 3 OF S TO [SENT, SIGMA1]
    IF T = (ANY + SSUBJ + OF(Q) + SPRED + OF(NOUNPHRZF)) THEN
    BEGIN LSUBT = SPRED; T = LANY + EVAL(X) SSUBJ(VAR) END
    ELSE PRINT(FAULT, PRED, 1); GO TO SENT1 END
    ELSE PRINT(ERROR, PRED, 1); GO TO SORRY

LINE 3 START EVALUATION OF LOGICAL TRANSLATION
SORRY! DISPLAYON; PRINT(S, T, SENTENCE-., ISNT-., INN-., GRAMMAR); DISPLAYOFF;
CLEAR(4); GO TO MORE1
SJACCESS IF T = (WFF + LANY: ANY) THEN T = TP= LANY ELSE GO TO SORRY1
QKEY = TRUE; T = EVAL T; J COLLECT;
KEY = TRUE; T = EVAL T; J COLLECT;
DISPLAYON;
    IF EVAL T THEN BEGIN IF TP = ALLF.(FORM, FORM) THEN BEGIN
    KEY = QKEY; FALSE; T = TP = .MSI
    QKEY = TRUE; T = EVAL T; J COLLECT;
    KEY = TRUE; T = EVAL T; J COLLECT;
    IF EVAL T THEN PRINT(.VALID) ELSE BEGIN PRINT(.VAQUOUS);
    IF -INTRP2<APCONT>1 THEN BEGIN INTRP2 = TRUE; INSTRING = EXSTRING;
    30 TO MORE END END
    INTRP2 = FALSE END
ELSE PRINT(.VALID) END ELSE PRINT(.INVALID))
DISPLAYOFF; CLEAR(4));
GO TO MORE;
OJT; END OF PROGRAM;
N
REMO 1
END;
BIBLIOGRAPHY


2. Aristotle, "De Sophisticis Elenchis" (On Sophistical Refutations).


43. Lindsay, R. K., "Toward the Development of a Machine which Comprehends," University of Texas, Austin, Texas, May 1961.


This dissertation presents a method called syntax-directed interpretation which permits the use of semantic information in a syntactic analysis of sentences taken from a restricted domain of natural language. This method is used in the resolution of syntactic ambiguity of English sentences with respect to a variable but well defined universe of discourse.

The first chapter defines the problem and surveys previous work related to natural language and question-answering systems.

In chapter two we describe an integrated approach to natural language processing which combines both syntax and semantics within the framework of a natural inference system. The logical properties of completeness and consistency are demonstrated for such systems.

In chapter three a computer program (GRANIS) written in FORMULA ALGOL and using the method of syntax directed interpretation is described. A picture, which can be input directly via a graphic display console, serves as a context within which syntactic ambiguity of input sentences can be resolved.

Chapter four considers various extensions of GRANIS in the direction of greater habitability, inferential power, knowledge acquisition, and adaptability. Chapter five concludes with a summary of results and their
Implication for future work in this field. The program listing in Appendix 2 forms an integral part of this thesis, and some effort was taken to ensure that it would be readable.
<table>
<thead>
<tr>
<th>KEY WORDS</th>
<th>LINK A</th>
<th>LINK B</th>
<th>LINK C</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROLE</td>
<td>WT</td>
<td>ROLE</td>
<td>WT</td>
</tr>
<tr>
<td>ROLE</td>
<td>WT</td>
<td>ROLE</td>
<td>WT</td>
</tr>
<tr>
<td>ROLE</td>
<td>WT</td>
<td>ROLE</td>
<td>WT</td>
</tr>
</tbody>
</table>