Natural Language Parsing Systems

Edited by Leonard Bolc

With Contributions by
J. G. Carbonell K. W. Church W. Dilger T. W. Finin
P. J. Hayes W. A. Martin J. G. Neal R. S. Patil
J. Pitrat A. Sågvall Hein S. C. Shapiro
S. L. Small M. Stone Palmer M. Thiel

With 151 Figures

Volume Editor
Leonard Bolc
Institute of Informatics, Warsaw University,
PKiN, pok. 850, PL-00-901 Warsaw, Poland

ISBN 3-540-17537-7 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-17537-7 Springer-Verlag New York Berlin Heidelberg

Library of Congress Cataloging-in-Publication Data
Natural language parsing systems. (Symbolic computation. Artificial intelligence) Includes bibliographies and index. 1. Parsing (Computer grammar) I. Bolc, Leonard, 1934- .
II. Carbonell, Jaime G. (Jaime Guillermo) III. Series. P98.N34 1987 415 87-12856

Printed in Germany

Typesetting, printing, and bookbinding: Appl, Wemding

Springer-Verlag
Berlin Heidelberg New York
London Paris Tokyo

Knowledge-Based Parsing

J.G. Neal and S.C. Shapiro

Abstract. An extremely significant feature of any natural language (NL) is that it is its own metalanguage. One can use an NL to talk about the NL itself. One can use an NL to tutor a non-native speaker, or other poor language user, in the use of the same NL. We have been exploring methods of knowledge representation and NL understanding (NLU) which would allow an artificial intelligence (AI) system to play the role of poor language user in this setting. The AI system would have to understand NL utterances about how the NL is used, and improve its NLU abilities according to this instruction. It would be an NLU system for which the domain being discussed in NL is the NL itself.

Our NLU system is implemented in the form of a general rule-based inference system which reasons according to the rules of its knowledge base. These rules comprise the system's knowledge of language understanding in the same way that the rules of any rule-based system comprise that system's knowledge of its domain of application. Our system uses the same knowledge base for both linguistic and other knowledge since we feel that there is no clear boundary line separating syntactic, semantic, and world knowledge.

We are exploring the possibility of an NLU system's becoming more facile in its use of some language by being told how that language is used. We wish this explanation to be given in an increasingly sophisticated subset of the language being taught. Clearly, the system must start with some language facility, but we are interested in seeing how small and theory-independent we can make the initial, "kernel" language. This article reports the current state of our work.

1 Introduction

1.1 Overview

An extremely significant feature of any natural language (NL) is that it is its own metalanguage. One can use an NL to talk about the NL itself. One can use an NL to tutor a non-native speaker, or other poor language user, in the use of the same NL. We have been exploring methods of knowledge representation (KR) and NL understanding (NLU) which would allow an artificial intelligence (AI) system to play the role of poor language user in this setting. The AI system would have to understand NL utterances about how the NL is used, and improve its NLU abili-

.ties according to this instruction. It would be an NLU system for which the domain being discussed in NL is the NL itself.

It is essential to our approach to have the system's parsing and linguistic knowledge be an integral part of its domain knowledge. Acknowledging that what is meant by "meaning" is controversial (Quine, 1948), we take the meaning or significance of a word or phrase to include linguistic knowledge about the word or phrase. For example, we feel that how a word like "dog" is used in language is a part of its "meaning", along with other properties such as the fact that "dog" denotes a special kind of animal with typical characteristics. The implementation of our system is based upon the above stated view and therefore the rules and assertions comprising the system's knowledge of language understanding, including syntax, is integrated into the system's knowledge base along with its other task domain knowledge.

We are exploring the possibility of an NLU system's becoming more facile in its use of some language by being taught how that language is used. The teacher might be a conversation partner who happens to use some phrase the system is not yet familiar with, or a language theorist who wants to find out if she can explain her theory completely and clearly enough for the system to use language according to it. We wish this explanation to be given in an increasingly sophisticated subset of the language being taught. That is, why not test and make use of the system's language capability by using it to continue the system's "education"? Clearly, the system must start with some language facility, but we are interested in seeing how small and theory-independent we can make the initial, "kernel" language.

In this chapter, we will discuss our knowledge representation techniques, the system's kernel language (KL), and parsing strategy. We will demonstrate how our system can be instructed in the use of some language defined by the teacher and how the system's acquired language can itself be used as its own metalanguage. The kernel language only incorporates primitive relations such as one token being a predecessor of another in a string, membership in a lexical or string category, and constituency. As an example of using the system's language as its own metalanguage to enhance its language capability, we will demonstrate, starting with only the KL, how the system can be instructed with regard to the number (i.e., singular or plural) of some words and then be informed that "If the head-noun of a noun-phrase X has number Y, then X has number Y". This newly acquired knowledge can then be applied by the system to infer that since "glasses" is plural, so is "the old man's glasses" when it reads this phrase in a sentence such as "The old man's glasses were filled with water".

Our system is able to understand when strings are *mentioned* in input utterances as well as when they are *used* to communicate with the system. This capability is demonstrated frequently in this chapter, but particularly in Sect. 4 with the classic sentence from Tarski (1944): "'Snow is white' is true if and only if snow is white".

The use of inference and world knowledge is essential for a system to parse sentences such as "John saw the bird without binoculars" and "John saw the bird without tailfeathers" from Schubert and Pelletier (1982) or "John saw the man on the hill with a telescope". Our research is based upon the concept of having parsing performed by a general reasoning system which has the capability of

world knowledge inferences during parsing, since the "parser" is not a separate isolated component with special sublanguage, representations, or knowledge base.

1.2 Fundamental Assumptions

Our system incorporates the *use-mention distinction* (Quine, 1951) for language. Our representations reflect the fact that the meaning of a token or surface string is distinct from the token or string itself. Our system's knowledge base maintains a representation for a token or surface string that is distinct from the representation of the interpretation of the input token or string. This distinction is the same as between a numeral and a number in mathematics. To refer to a word or string rather than its meaning, the user must use the usual English convention of prefacing the word by a single-quote mark or enclosing the string in quotation marks. (See Sections 2.2.1 and 2.4.2 for more information.)

A second principle upon which our work is based is that each occurrence of a given surface string in the input stream is assumed to have a different interpretation, unless the teacher has entered rules into the system to dictate otherwise. For example, if a name such as "John" has been entered into the lexicon and is used twice, either in successive utterances or within the same utterance, then the system interprets each occurrence of the name as referring to a different entity unless the teacher has instructed the system otherwise. Since an NLU system must be capable of handling ambiguities, and, in a situation in which no explicit rules are known to the system to guide it in determining whether a word or phrase is ambiguous, it must have a default procedure to follow, we have chosen to implement the above principle. Although our approach would seem to overly complicate the network, it is a reasonable default principle since there is some evidence that merging of nodes is easier than splitting nodes (Maida and Shapiro, 1982).

A third principle which is fundamental to our theory is that all possible parses and interpretations of a surface string are to be determined according to the language definition used by the system. We feel that multiple interpretations, when justified by the language definition, are warrented since agile human minds frequently perceive alternative interpretations and, in fact, a great deal of humor is dependent upon this.

Our system does not currently do morphological analysis. One of the areas in which we plan to do future research is knowledge-based morphological analysis. We plan to develop a system component that would perform morphological analysis and function as a preprocessor or coprocessor with the system discussed in this article.

1.3 Declarative Knowledge Representation in an Integrated Knowledge Base

Our approach is to represent knowledge in declarative form, to the greatest extent possible, in the semantic network formalism. This applies to all knowledge including linguistic knowledge and the rules which are applied by the inference machine

to guide the system's reasoning, the parsing process being one manifestation of the system's reasoning according to the rules of its network knowledge base. It is our intent that the system's knowledge, including its linguistic knowledge, be available to the teacher in the same way that domain knowledge is in other AI systems.

Furthermore, the declarative form is a more suitable form for linguistic knowledge in theoretical studies of language. A language definition or description is inherently declarative, and as Pereira and Warren have pointed out: "The theorists have concentrated on describing *what* natural language is, in a clear and elegant way. In this context, details of *how* natural language is actually recognized or generated need not be relevant, and indeed should probably not be allowed to obscure the language definition" (1980, p.269, italics in the original). In this regard, a declarative representation is preferable to a formalism such as an ATN, in that the ATN is a description of a *process* for recognizing a language.

Our system uses an integrated knowledge base for both linguistic and other knowledge as advocated by Pollack and Waltz (1982) and by Dahl (1981). As indicated in Section 1.1, we take the meaning of a word or phrase to include linguistic knowledge about the word or phrase and its use. Furthermore, we feel that there is no clear boundary line separating syntactic, semantic, and world knowledge. For example, it is not clear to what extent the classification of words into lexical categories depends on meaning, function, or form. Should certain words be classified as mass nouns because they fit certain distributional frames or have a certain form (e.g., I used {sand, the sand, a bag of sand, *a sand, *two sands.}) or are the frames and forms simply a reflection of the property we think of as characterizing the substances named by mass nouns, namely that the substance is not naturally physically bounded and that when two amounts of the substance are "put together" they become one amount? Perhaps certain aspects of syntax cannot or should not be separated from semantics. Furthermore, the terms "semantic knowledge" and "world knowledge" seem only to be used to informally express a measure of the sophistication or complexity of knowledge.

1.4 System Overview

Our Natural Language System is being developed and implemented using the SNePS semantic network processing system (Shapiro, 1979a; Shapiro and the SNePS Group, 1981). The terminology and representations for some of the basic categories, objects, and relations of this work evolved from a preliminary study reported in Shapiro and Neal (1982). Figure 1 illustrates an overview of the system.

The semantic network formalism has been used by many researchers for knowledge representation (Quillian, 1968, 1969; Rumelhart and Norman, 1973; Simmons, 1973; Woods, 1975; P. Hayes 1977; Schubert, 1976; Hendrix, 1978, 1979; Schubert et al., 1979; Brachman, 1979). In contrast to other semantic network implementations, the SNePS system provides a uniform declarative representation for both rules and assertions in the network (Shapiro, 1971, 1979b). Furthermore, our system comprises an effort to utilize a common representation for problem-solving and language-comprehension information as advocated by Charniak

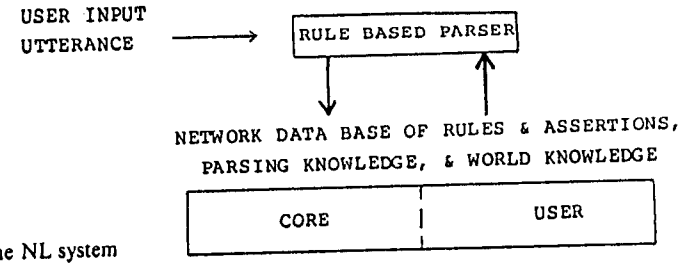


Fig. 1. Overview of the NL system

(1981). Our system is similar to the Prolog-based (Roussel, 1975) systems of Warren and Pereira (1982), Dahl (1979, 1981), and McCord (1982) in that it is implemented in a logic-based system in which processing is a form of inference. The SNePS inference package (Shapiro et al., 1982), however, is not based on the resolution principle (Robinson, 1965) as is Prolog, but on a multi-processing approach (Kaplan, 1973; McKay and Shapiro, 1980) incorporating a producer-consumer model. SNePS also provides a facility for "procedural attachment" in rules to handle processing knowledge for which the declarative network representation is unnatural.

The PSI-KLONE system (Bobrow and Webber, 1980) uses linguistic knowledge represented in a KL-ONE network (Brachman, 1978a, 1978b, 1979) to function as semantic interpreter for parsed surface strings. The PSI-KLONE interpreter, however, functions in cooperation with an ATN parser in the RUS framework (Bobrow, 1978). In contrast, we are implementing an integrated system for syntactic and semantic processing which uses a uniform representation for syntactic and semantic knowledge.

The rule-based parser of Figure 1 is essentially the SNePS inference package which reasons according to the rules of the knowledge base.

The knowledge base consists of CORE knowledge and USER knowledge. The CORE knowledge is provided by the designers of the system and defines a kernel language initially acceptable to the system. USER knowledge results from the processing of user input utterances.

The function of our NL parser is twofold:

1. derivation of zero or more annotated parse trees for the input surface string;
2. construction of a network representation for the interpretation of the input utterance from the annotated parse tree and from other relevant knowledge from the network data base.

The above two functions are not handled by separate processors, but, instead, are both accomplished by the SNePS inference package as a result of the application of CORE and USER rules. The processes of accomplishing the two functions are interrelated and can cooperate. The interpretation of a surface string will depend on how it is syntactically parsed and, conversely, the syntactic parse of a surface string can depend on the meanings of related, constitutent, or neighboring strings.

The two processes are not carried out in a purely sequential fashion for a given input utterance, since interpretations can be constructed for parsed constituent strings before the parsing of the entire utterance is complete.

1.5 Knowledge Representation Techniques

A SNePS semantic network is a directed graph with labeled arcs in which nodes represent concepts and the arcs represent nonconceptual binary relations between concepts. It is generally agreed that the nodes of a semantic network represent intensional concepts (Woods, 1975; Brachman, 1977; Maida and Shapiro, 1982). A "concept" is something in our domain of interest about which we may want to store information and which may be the subject of "thought" and inference. Since each concept is represented by a node, the relations represented by the arcs of our system are not conceptual, but structural (Shapiro, 1979a).

The primary type of arc in a SNePS network is the *descending* arc and if there is a path of descending arcs from node N to node M, N is said to *dominate* M. Two important types of nodes are *molecular* and *atomic* nodes. Molecular nodes are nodes that dominate other nodes. Atomic nodes are simply not molecular. Atomic nodes can be *constant* (representing a unique semantic concept) or *variable*. Variable nodes are used in SNePS as variables are used in normal predicate logic notations. Network nodes can also be categorized as in the table in Figure 2.

A propositional molecular node N together with the arcs incident from the node and the nodes $M_1,\ldots,M_k$ immediately dominated by N correspond to a case frame (Fillmore, 1968; Bruce, 1975) where the arc names correspond to the slot names, and the nodes $M_1,\ldots,M_k$ represent the slot fillers. Undominated molecular nodes in a SNePS network represent propositions believed by the system. Concepts such as the following are propositional and are represented by molecular nodes: Lexeme L is a member of category C; S1 is a constituent string of S2; lexeme L has number N (i.e. singular or plural). Simple examples of propositional nodes are M1 and M2 of Figure 3.

Node M1 represents the proposition that B1 represents the concept expressed by the word "NOUN" and M2 represents the proposition that the lexeme "SNOW" is in the category called "NOUN".

The syntactic objects represented in our network knowledge base include morphemes, surface strings, and nodes of annotated parse trees. Individual morphemes are represented as nodes whose identifiers or print names are the morphemes themselves. The representation of a surface string utilized in this study consists of a network version of the list structure used by Pereira and Warren (1980). This representation is also similar to Kay's charts (1973) in that whenever alternative analyses are made for a given substring of a sentence, the sentence structure is enhanced by multiple structures representing these alternative analyses. Retention of the alternatives avoids the reanalyses of previously processed substrings which occurs in a backtracking system. Our basic representation of a surface string is illustrated in Figure 4.

Nodes identified by the atoms B0, SNOW, IS, and WHITE are atomic nodes and represent objects: the empty string, and tokens "SNOW", "IS", and

Node Category	Type of Concept
Non-dominated (asserted) molecular node	Asserted proposition which is "believed" by the System
Dominated molecular node	Proposition or structured object which is a participant in a proposition
Atomic node	Object

Fig. 2. Table of node categories

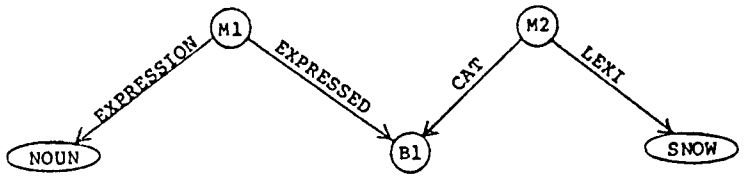


Fig. 3. M1 and M2 are simple examples of propositional nodes

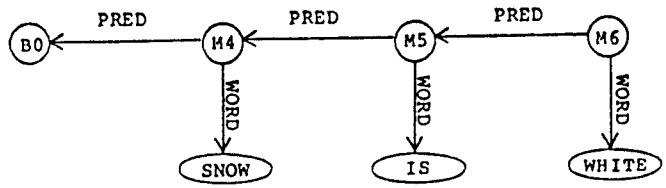


Fig. 4. Basic network representation of a surface string

"WHITE", respectively. Node M4 is molecular and represents the initial string "SNOW". M5 is also molecular and represents the initial string "SNOW IS", and similarly for node M6. A node such as M6 that represents an object would typically be dominated in our system by some node representing a proposition about it.

As each word of an input string is read by the system, the network representation of the string is extended and relevant rules stored in the SNePS network are triggered.

Interpretations of surface strings are also represented as nodes of our network knowledge base. The kernel language of the system enables the user to define case frame structures and to define rules to guide the system in interpreting input utterances.

1.6 Core Knowledge and the Kernel Language

Our approach is to provide the teacher (user) with a kernel language in which she can begin to "explain" the syntax and semantics of some natural or invented language to the system. The present version of our kernel language includes:

a) predefined terms such as L-CAT, the set of the names of lexical classes, and S-CAT, the set of the names of string classes; S-CAT contains the important category names ANT-CLAUSE, CQ-CLAUSE, and RULE-STMT, which are used to bootstrap into a more sophisticated rule input capability;
b) predefined objects such as (i) initial strings and (ii) bounded strings with beginning and ending token;
c) predefined relations such as (i) lexeme L is a member of category C; (ii) bounded string B is a member of category C and this structure is represented by S; (iii) structure S expresses concept C; (iv) structure S1 is a constituent of structure S2;
d) predefined functions such as (i) a test to determine whether two network nodes are identical, (ii) a test to determine whether two bounded surface strings match, and (iii) a test to determine whether one bounded string precedes another bounded string.

The KL provides the teacher with a basic language of rewrite rules for the purpose of defining *syntactic* lexical insertions, context free phrase structure rules, and context sensitive rules as well as *semantic* mappings from string categories to case frame structures and mappings from string categories to case frame participant or component slots.

1.7 Metalanguage Conventions and Symbols

In this chapter, we use the notational convention that words written in upper case letters denote words of KL and we use the metasymbols:

⟨⟩ denote a non-terminal; if the angle brackets enclose the name of a category of the core or a user-defined category, then such usage denotes a variable whose domain is the category named within the enclosing brackets.
() for grouping.
* Kleene star: when used as a superscript on an item, denotes zero or more of the items in a finite sequence.
+ when used as a superscript on an item, denotes one or more of the items in a finite sequence.
... ellipsis.

2 Core Knowledge and Representations

2.1 Uniform Representation and Intensional Constructs

We use the semantic network formalism to represent both syntactic and semantic knowledge in the form of assertions and rules to be applied in inference formation. We include linguistic knowledge in the network knowledge base and use the network formalism as a uniform "language" with which to represent all types of knowledge. Thus we model surface strings and syntactic properties and categories as intensions (Woods, 1975; Brachman, 1977; Maida and Shapiro, 1982), concepts, or objects of thought.

2.2 Predefined Categories, Objects, Relations, Functions

2.2.1 Predefined Categories

We are investigating the capability of an NLU system becoming more adept in the use of some language by being instructed in the use of the language. The system must start with some language facility, but we are striving to make the core knowledge base as small and theory-independent as possible.

Included among the core primitives are certain predefined categories. Since we are designing a language *capability* that is as theory-independent as possible and not a robust parser for a predetermined language such as English, some of these categories are initially empty, while others have very few members. All the categories are to be utilized by the teacher, either directly or indirectly, and the membership of the categories expanded by the teacher as the definition of her target language takes shape.

The most basic of these categories are L-CAT, S-CAT, and VARIABLE. L-CAT consists of the names of lexical classes or classes of terms. L-CAT initially contains the predefined terms L-CAT, S-CAT, VARIABLE, PUNCTUATION, and FUNCTION-NAMES. Names that the teacher would add to L-CAT might include, for example, NOUN, VERB, and PREPOSITION.

The purpose of VARIABLE is to contain all the identifiers that the teacher will use as variables in her processing rules when stated as input to the system. The VARIABLE category is initially empty.

The category PUNCTUATION initially contains the punctuation marks period, single-quote, and double-quote.

The class FUNCTION-NAMES contains the names of the functions that the teacher has available to be used in a form of procedural attachment to the declarative rules of the network knowledge base. FUNCTION-NAMES initially contains the names of the tests discussed in Section 2.2.4: IDENTITY-TEST, STRING-MATCH-TEST, and PRECEDES-TEST.

S-CAT is defined to be the set of all the names of string categories. S-CAT initially contains the names of the predefined string categories UTTERANCE, P-RULE, CASE-FRAME-DEFINITION, CASE-SLOT-DEFINITION, LITERAL, LITERAL-STRING, UNIQUE-MEANING-CAT, VAR-APPOSITION-PHR,

MAIN-APPOS-PHR, VAR-NAME, ANT-CLAUSE, CQ-CLAUSE, and RULE-STMT. The string categories P-RULE, CASE-FRAME-DEFINITION, CASE-SLOT-DEFINITION, LITERAL, LITERAL-STRING, and VAR-NAME each have predefined syntax. For the remaining string categories except UTTERANCE, the definition of the syntax is left to the teacher, VAR-APPOSITION-PHR and RULE-STMT having restrictions discussed later in this article. The class UTTERANCE contains all input surface strings.

The predefined string category P-RULE includes all strings that qualify as production or syntactic rewrite rules as discussed in Section 2.5.2. These rewrite rules are part of the kernel language understood by the system.

The kernel language includes semantic rewrite rules to enable the teacher to define case frames and associations between case frames and particular string categories for use in the interpretation of input utterances. CASE-FRAME-DEFINITION and CASE-SLOT-DEFINITION are string categories that contain the two types of semantic rewrite rules. The capability associated with the CASE-FRAME-DEFINITION and CASE-SLOT-DEFINITION classes is discussed in Section 2.5.3.

LITERAL is the category of strings consisting of a single-quote mark followed by a lexeme. LITERAL-STRING is the category of strings that consist of a pair of double-quote marks enclosing a surface string.

VAR-APPOSITION-PHR, MAIN-APPOS-PHR, and VAR-NAME are string categories that enable a variable to be used as an appositive so as to establish the variable as the identifier for the MAIN-APPOS-PHRase which it is adjacent to. For example, after input of appropriate user-defined rules to the system, the string "a noun-phrase X" (from the sentence "If the head-noun of a noun-phrase X has number Y then X has number Y") could be parsed as a VAR-APPOSITION-PHR with "a noun-phrase" as the MAIN-APPOS-PHR and "X" the VAR-NAME so that in parsing the stated rule, X is remembered by the system as an identifier for the unknown noun-phrase refered to in the phrase and is thus capable of being referred to again later as in the given rule example. Since no referencing mechanisms are built into our system to enable the teacher to refer to previously mentioned concepts, the above string categories assist the teacher in establishing rules to determine the referencing process according to her own theory. The use of this capability is illustrated by example in Section 3.

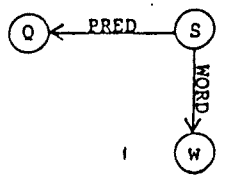
As the teacher proceeds to instruct the system in her language definition, she will need to enter rules that cannot be expressed in the language of rewrite rules. Such rules would include rules concerning the semantics of utterances. Therefore, the core primitives include three initially empty string categories, RULE-STMT, ANT-CLAUSE, and CQ-CLAUSE to enable the teacher to define the syntax of general conditional rules. These categories are discussed in subsequent sections.

UNIQUE-MEANING-CAT is defined to be the class of all the strings that have a unique meaning. That is, if a string is in UNIQUE-MEANING-CAT, it must express the same intension each time it is encountered in an input utterance. As stated in Section 1.2, a premise of our theory and NL system is that each time a given word or string is "read" by the system, it has a new or different meaning unless this meaning is determined by rules and/or assertions input by the teacher.

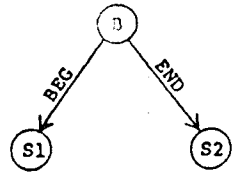
2.2.2 Predefined Objects

The predefined objects essential to our theory and implementation are the concepts of the Initial String and the Bounded String. These objects and their network representations are described below.

a) Initial string S consists of the word or symbol W concatenated to the initial string Q. Q may be the null string represented by node B0.



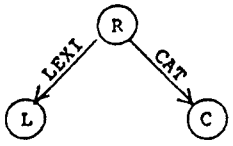
b) Bounded string B represents the surface string beginning with the last word of initial string S1 and ending with the last word of initial string S2 where S1 precedes S2.



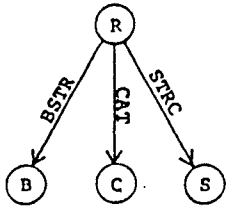
2.2.3 Predefined Relations

It is necessary for the NL system to have a set of predefined relations for knowledge representation. The current set of these relations and their corresponding semantic network structures are listed below.

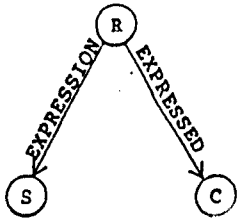
a) Lexeme L is a member of category C; e.g., node M21 of Figure 5 represents the concept that 'STUDENT is a NOUN.



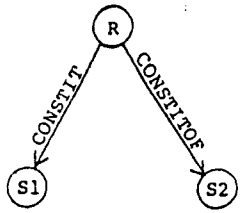
b) The bounded string B is in category C and this structure or parse of the string B is represented by node S (analogous to a node of an annotated parse tree); e.g., node M43 of Figure 5 represents the concept that the structure represented by B21 represents a parsing of the bounded string represented by M42 as an INDEF-NOUN-PHRASE.



c) Structure or parsed string S expresses concept C; e.g., node M20 of Figure 5 represents the concept that the string "NOUN" expresses the category of nouns represented by node B10.



d) The structure S1 is a constituent of structure S2; e.g., node M44 of Figure 5 represents the concept that the literal 'STUDENT is a constituent of the structure represented by node B21.



e) The rule structures of SNePS (Shapiro, 1979a).

Figure 5 shows a surface string enhanced by additional structure that would result from the system's reading and parsing the input string "A STUDENT" after some syntactic rules had been input by the teacher (e.g., 'A is an INDEF-DET, 'STUDENT is a NOUN, a string consisting of an INDEF-DET followed by a NOUN is an NOUN-PHRASE).

2.2.4 Predefined Functions

The following functions are essential for the NL system and could not be efficently implemented in the declarative SNePS language.

a) *Identity test* takes two network nodes as arguments, and returns true if the two nodes are identical and returns false otherweise.
b) *String-match test* takes two bounded strings in network representation as arguments, and returns true if the sequence of words or symbols in the two strings are identical, and returns false otherwise.
c) *Precedes test* takes two bounded strings in network representation as arguments, and returns true if the first string precedes the second string in the input stream, and returns false otherweise.

2.3 The Reading Function

The system's reading function "reads" one token (lexeme or punctuation mark) at a time from the input stream. For each input token, the structure of Figure 6 is added to the network, where node S represents the previously added initial string, C represents the lexical category of the token, I represents the newly established initial string, and B represents the newly added bounded string.

If the token belongs to any lexical categories, this membership would already be represented in the network in the form of relation (a) of Section 2.2.3 (how such relations are established in explained in Section 2.4). The lexical categories to which the input token belongs are found in the network by the reading function and, for each such category C, a node such as M of Figure 6 is added. If no such categories exist, then only the initial string and the bounded string are added.

Forward inference may be triggered by the addition of the network of Figure 6 for each token, depending on what rules are already in the system. For example, in

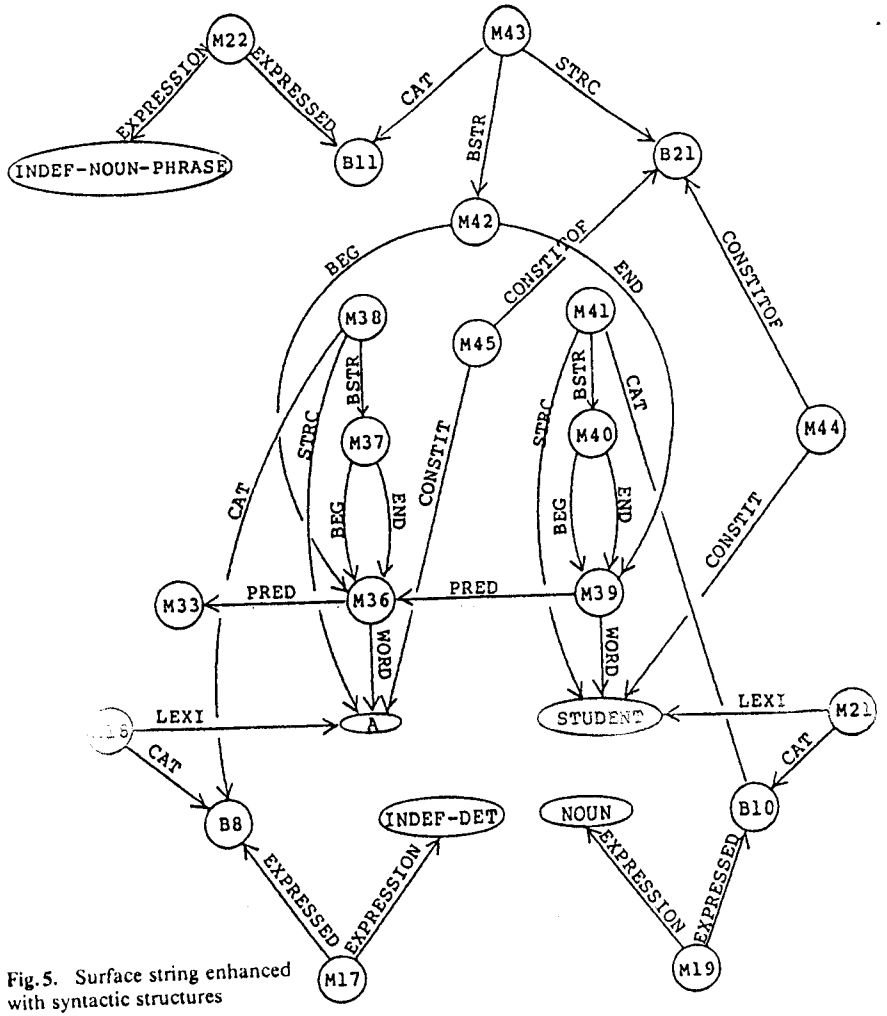


Fig. 5. Surface string enhanced with syntactic structures

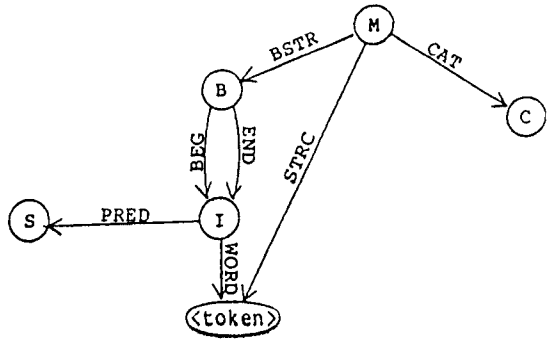


Fig. 6. The structure added to the network for each input token

Figure 5, nodes M38 und M41 are added by the reading function and nodes M42, M43, M44, and M45 are built only if there is a rule in the system that asserts that an INDEF-DET followed by a NOUN is an INDEF-NOUN-PHRASE.

2.4 The Representational Mapping

2.4.1 Introduction

Not all strings of a language form meaningful "chunks". For example, the substring "a large" from the sentence "A large aggressive dog frightened the girl" is not a conceptually coherent constituent of the sentence. Many researchers, e.g., Fodor and Garrett (1967), Bever (1970, 1973), and Levelt (1970, 1974), have investigated the relationship between surface constituents and the conceptually coherent components of an utterance. There seems to be good evidence for surface constituents being the coherent units for comprehension of discourse. How sentential constituents or discourse constituents (moving up to a higher level in the organization of text) are utilized in the comprehension process is an active field of research Brown and Yule, 1983).

We let R designate the representational mapping (Allen, 1978) from surface strings to their interpretations. The domain of R contains the categories of strings that form conceptually coherent units, possibly depending on linguistic or other contexts. The domain of R initially contains predefined categories L-CAT, S-CAT, VARIABLE, LITERAL, LITERAL-STRING, VAR-APPOSITION-PHR, RULE-TMT, P-RULE, CASE-FRAME-DEFINITION, and CASE-SLOT-DEFINITION. They are discussed in the following sections.

We provide the teacher of the system with the facility for determining what the conceptually coherent constituents will be, in addition to the core, and for instructing the system in their use.

2.4.2 Base Cases

The categories L-CAT, S-CAT, VARIABLE, LITERAL, and LITERAL-STRING are the base cases for the representational mapping. The most basic subclass of the domain of R is L-CAT, the class of identifiers for the lexical categories of the system, including both system identifiers and user-defined identifiers. The class L-CAT contains the predefined identifiers L-CAT, S-CAT, and VARIABLE. The representational mapping applied to any identifier in L-CAT maps to a constant base node. In Figure 6, the interpretations of the identifiers L-CAT and S-CAT are represented by nodes B1 and B2 respectively. Similarly, if the system is informed that 'NOUN is an L-CAT, then its interpretation is represented by a base node (B4 of Fig. 7). The information that 'GOOSE is a NOUN and that 'NOUN-PHRASE is an S-CAT is also represented in Figure 7.

A member of VARIABLE maps to a corresponding network variable node. Since the interpretation of a user variable must be local to the rule in which it is used, the representational mapping applied to the class VARIABLE is handled in a special manner as explained in Section 3.

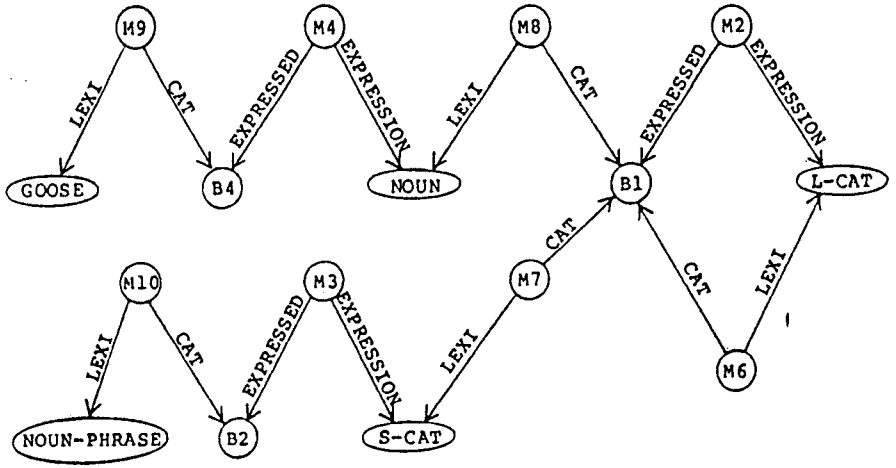


Fig. 7. Representation of some basic lexical knowledge

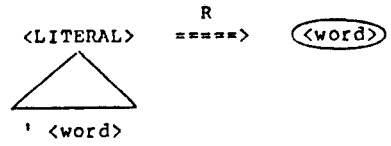


Fig. 8. The representational mapping applied to a LITERAL

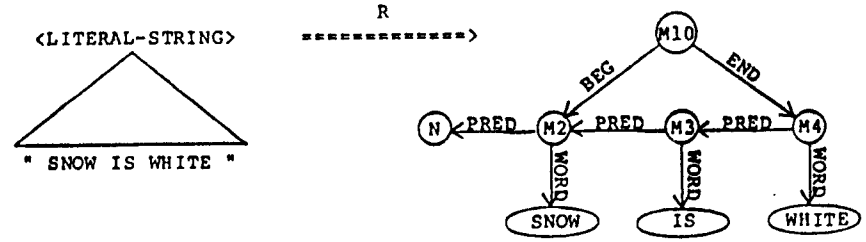


Fig. 9. Representational mapping applied to a LITERAL-STRING

The representational mapping applied to a LITERAL (defined in Section 2.2.1 as the single-quote mark followed by a word) maps to the node whose identifier is the word itself, as illustrated in Figure 8.

The representational mapping applied to a LITERAL-STRING (a string enclosed by double-quote marks) maps to the bounded string representing the string enclosed by the quote marks. Figure 9 illustrates the representational mapping applied to the literal string "SNOW IS WHITE".

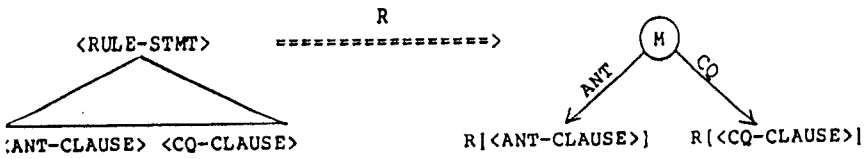


Fig. 10. Representational mapping applied to a RULE-STMT

2.4.3 Propositions and Structured Objects

Some string categories contained in the domain of R are mapped to non-atomic network (case frame) structures representing propositions or structured objects by the representational mapping. The system has just two predefined string categories, namely P-RULE and RULE-STMT, whose members' interpretations are represented as non-atomic structures by the representational mapping. P-RULEs have a predetermined syntax and are translated into SNePS network rules using the predefined structures. RULE-STMT is defined as the class of strings that are interpreted by the system as general rules. This class is initially empty and the syntax is to be determined by the teacher. A RULE-STMT must have an ANT-CLAUSE and a CQ-CLAUSE as constituents. The structure resulting from the application of the representational mapping R to a RULE-STMT is illustrated in Figure 10.

The ANT-CLAUSE category is defined as the class of strings that can be used in antecedent position in rules input by the teacher. Similarly, the CQ-CLAUSE category is defined as the class of strings that can be used in consequent position in rules input by the teacher. Both of these classes are initially empty and the syntax of RULE-STMTs, ANT-CLAUSEs, and CQ-CLAUSEs is to be determined by the teacher. An example is discussed in Section 3.

The teacher can add new string categories, whose interpretations are to be represented by non-atomic network structures, to the domain of R and specify the semantics of these string categories by using the semantic rewrite rule capability discussed in Sect. 2.5.3.

2.4.4 Participants in Propositions or Relations; Components of Structured Objects

In the previous section, the category of natural language phrases that assert relations between concepts or objects was discussed briefly. This type of phrase maps to the top node of a molecular representational structure.

Many phrases of natural language refer to individual concepts or objects that are participants in relations or propositions or that are components of structured objects. This type of phrase would map to a slot of one or more network case frame structures. The system has no predefined string categories which map to "participant" slots. The syntax for creating new categories of this type and their associated semantics is discussed in Section 2.5.3.

2.5 Kernel Language

2.5.1 Predefined Terms

As previously indicated, we are attempting to provide a facility with which a person can define a target language and yet keep the core as small and unbiased as possible. It is essential to provide the person (teacher) with a kernel language with which to start building up her language definition. The kernel language (KL) of our system consists of predefined terms, syntactic rewrite rules, and semantic rewrite rules. The predefined terms of the system are the names of the categores discussed in Section 2.2.1.

2.5.2 Syntactic Rewrite Rules

The kernel language includes linguistic rewrite rules to enable the teacher to instruct the system in the basic syntax of her target language.

a) Lexical Entry: The KL includes syntactic production rules of the form ⟨L-CAT⟩ → ⟨LITERAL⟩ where ⟨L-CAT⟩ represents the name of a lexical category that has already been defined. A LITERAL was defined in Section 2.2.1 as consisting of the single-quote followed by a word (the single-quote is part of the KL and indicates that the following word is mentioned rather than used). This form of production rule is the means of entering lexical items such as

L-CAT → 'NOUN
L-CAT → 'PROPER-NOUN
L-CAT → 'DEF-DET
L-CAT → 'INDEF-DET
L-CAT → 'VERB
L-CAT → 'BE-VERB
L-CAT → 'ADVERB
L-CAT → 'ADJECTIVE
NOUN → 'GOOSE
NOUN → 'GEESE
PROPER-NOUN → 'GRADY
PROPER-NOUN → 'GLADYS
DEF-DET → 'THE
INDEF-DET → 'A
VERB → 'HAS
BE-VERB → 'IS

ADVERB → 'THEN

L-CAT → 'PREPOSITION
L-CAT → 'CONJ
L-CAT → 'PROPERTY
S-CAT → 'HEAD-NOUN
S-CAT → 'STRING
VARIABLE → 'X
VARIABLE → 'Y

ADJECTIVE → 'WHITE
ADJECTIVE → 'SINGULAR
ADJECTIVE → 'PLURAL
PREPOSITION → 'OF
CONJ → 'IF
PROPERTY → 'COLOR
PROPERTY → 'NUMBER
UNIQUE-MEANING-CAT → 'ADJECTIVE
UNIQUE-MEANING-CAT → 'PROPERTY

) **Context Free Rules:** The KL includes rules of the form

S-CAT) → ⟨s⟩$_1$. . .⟨s⟩$_k$, k > 0,

/here ⟨S-CAT⟩ represents the name of a string category and for each i, ⟨s⟩$_i$ is either LITERAL, the name of a lexical category previously entered as a member of -CAT as in (a) above, or the name of a string category.

Examples: PROPERTY-CLAUSE → SUBJECT PREDICATE
SUBJECT → NOUN-PHRASE
NOUN-PHRASE → LITERAL
NOUN-PHRASE → VARIABLE
NOUN-PHRASE → PROPER-NOUN
PREDICATE → RELATION-PREDICATE
PREDICATE → BE-PREDICATE
RELATION-PREDICATE → RELATION PREDICATE-ADJ
BE-PREDICATE → BE-VERB PROPERTY-INDICATOR
RELATION → 'HAS PROPERTY-INDICATOR
PROPERTY-INDICATOR → PROPERTY-CLASS-INDICATOR
PROPERTY-INDICATOR → PROPERTY
PROPERTY-CLASS-INDICATOR → PREDICATE-ADJ
RULE-STMT → 'IF ANT-CLAUSE 'THEN CQ-CLAUSE

Context Sensitive Rules: The KL includes syntactic production rules of the form

⟨ls⟩$_1$. . .⟨ls⟩$_n$ → ⟨rs⟩$_1$. . .⟨rs⟩$_n$, n > 0,

here each element ⟨ls⟩$_i$ or ⟨rs⟩$_i$ is either a LITERAL, the name of a lexical category, or the name of a string category; both sides of the rule must have the same umber of elements and for each element ⟨ls⟩$_i$ of the left side,

if ⟨ls⟩$_i$ is a LITERAL or lexical category, then the corresponding element ⟨rs⟩$_i$ of the right side must be the same as ⟨ls⟩$_i$;
if ⟨ls⟩$_i$ is the name of a string category, then the corresponding element ⟨rs⟩$_i$ of the right side can be either a LITERAL, lexical category name, or string category name.

iis facility allows the user to enter context sensitive rules, such as:

RELATION PREDICATE-ADJ → RELATION ADJECTIVE
RELATION PREDICATE-ADJ → RELATION VARIABLE
BE-VERB PREDICATE-ADJ → BE-VERB ADJECTIVE
'IF ANT-CLAUSE → 'IF PROPERTY-CLAUSE
'THEN CQ-CLAUSE → 'THEN PROPERTY-CLAUSE

ie first rule asserts that in the context of a RELATION, an ADJECTIVE is recnized as a PREDICATE-ADJ, the second asserts that in the context of a ELATION, a VARIABLE is parsed as a PREDICATE-ADJ, and the third serts that in the context of a BE-VERB, an ADJECTIVE is parsed as a PREDI-ATE-ADJ. Similarly, the fourth and fifth rules state that following the word "IF" "THEN", a PROPERTY-CLAUSE is parsed as an ANT-CLAUSE or CQ-LAUSE, respectively.

2.5.3 Semantic Rewrite Rules

a) Case Frame Definitions: The KL includes language to enable the teacher to define case frames and instruct the system in their use by using the syntax of a CASE-FRAME-DEFINITION:

⟨string-cat⟩ : : ⟨slot-name⟩$_1$ ⟨constit-name⟩$_1$ . . .
⟨slot-name⟩$_n$ ⟨constit-name⟩$_n$

where n > 0. Such a CASE-FRAME-DEFINITION is used by the system as follows: A string that is identified as being in category ⟨string-cat⟩ is mapped into a case frame such that for each slot identified by ⟨slot-name⟩$_i$, the slot-filler is the interpretation of the constituent string identified by ⟨constit-name⟩$_i$. The constituent strings need not be immediate constituents of the string in category ⟨string-cat⟩. The same ⟨constit-name⟩ can be used to specify the filler for more than one slot. For example, suppose the teacher wants to define a language in which an utterance such as "JOHN BOUGHT A HOUSE", involving the act of purchase, is interpreted to mean that the person bought the object for himself unless otherwise stated. To handle the semantics of such a clause, the teacher might want to define a case frame with AGENT and BENEFICIARY slots which are both filled by the interpretation of the same constituent of the clause. Our CASE-FRAME-DEFINITION facility provides for this eventuality.

If a string is parsed as a ⟨string-cat⟩ but is missing a constituent specified by the semantic rewrite rule, then a default representation of the slot-filler corresponding to the missing constituent is established in the form of an atomic node about which the system knows nothing, other than its being a participant in the case frame. In the context of a RULE-STMT the atomic node will be a variable node (see Section 1.3), otherwise a constant node. For example, to represent the interpretation of a sentence such as "THE HOUSE WAS PURCHASED YESTERDAY" the teacher might want to use the same case frame mentioned in the preceding paragraph. Since an AGENT and BENEFICIARY are implicitly part of the act of purchase, but not explicitly mentioned in the sentence, it is reasonable for the unmentioned participants to be represented in the interpretation of the sentence. The above default representation for the interpretation of a missing constituent provides the teacher with a facility for instructing the system how to interpret such a sentence.

An alternative syntax for the CASE-FRAME-DEFINITION is

⟨string-cat⟩: :⟨constit-name⟩.

The right side of the : : symbol is a degenerate case frame and the definition is interpreted as meaning that the semantics of the string of category ⟨string-cat⟩ is the same as that of the constituent string of category ⟨constit-name⟩. For example,

PROPERTY-CLAUSE: : PROPERTY OF SUBJECT
PROPERTY PROPERTY-INDICATOR
VALUE PREDICATE-ADJ
ANT-CLAUSE : : PROPERTY-CLAUSE
CQ-CLAUSE : : PROPERTY-CLAUSE

SUBJECT :: NOUN-PHRASE
NOUN-PHRASE :: LITERAL
NOUN-PHRASE :: VARIABLE
NOUN-PHRASE :: PROPER-NOUN
PROPERTY-INDICATOR :: PROPERTY-CLASS-INDICATOR
PROPERTY-INDICATOR :: PROPERTY
PREDICATE-ADJ :: ADJECTIVE
PREDICATE-ADJ :: VARIABLE

define the semantics of a PROPERTY-CLAUSE to be a case frame with three slots, such that the PROPERTYOF slot is filled by the interpretation of the SUBJECT of the PROPERTY-CLAUSE, the PROPERTY slot is filled by the interpretation of the PROPERTY-INDICATOR constituent of the PROPERTY-CLAUSE, and the VALUE slot is filled by the interpretation of the PREDICATE-ADJ constituent. The second and third definitions above indicate that the interpretation of an ANT-CLAUSE is the same as the interpretation of its PROPERTY-CLAUSE constituent and that the interpretation of a CQ-CLAUSE is the same as the interpretation of its PROPERTY-CLAUSE constituent, respectively. The next rule defines the interpretation of a SUBJECT to be the same as the interpretation of its NOUN-PHRASE constituent. The next three rules define the interpretation of a NOUN-PHRASE to be the interpretation of either its LITERAL constituent, its VARIABLE constituent, or its PROPER-NOUN constituent, whichever it has. The remaining rules are similarly understood by the system.

The representational mapping R builds a network structure such as that dominated by node M of Figure 11 as the interpretation of a PROPERTY-CLAUSE.

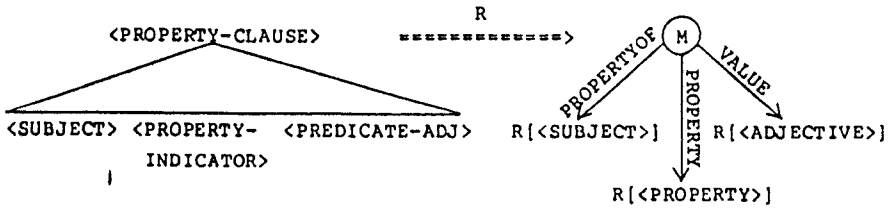


Fig. 11. Representational mapping applied to a PROPERTY-CLAUSE

) Case Frame Slot-Filler Definitions: In order to provide a capability for defining he semantics of a phrase whose interpretation is a slot-filler in a case frame, the ollowing type of semantic rewrite rule is included. The syntax of the CASE-SLOT-DEFINITION is

$$\langle\text{phr-name}\rangle \; >> \; ([\langle\text{slot-name}\rangle_1 \; \langle\text{string-name}\rangle_1 \; \ldots \; \langle\text{slot-name}\rangle_n \; \langle\text{string-name}\rangle_n \;])^+$$

where the square brackets are part of the object language and the + and the parentheses are metasymbols. The ⟨phr-name⟩ is the name selected by the teacher for

of the symbol >>. The object language symbol + must be used in place of at least one ⟨string-name⟩, designating the position of the interpretation of the string ⟨phr-name⟩ in the case frame.

Each set of brackets encloses a case frame definition as described in the previous section. That is, each slot named $\langle\text{slot-name}\rangle_i$ is filled by the interpretation of a string in category $\langle\text{string-name}\rangle_i$, if a string of category $\langle\text{string-name}\rangle_i$ is present as a (not necessarily immediate) constituent of the string of category ⟨phr-name⟩. The + symbol marks the slot whose filler is the interpretation of the ⟨phr-name⟩ string. The system represents the interpretation of the ⟨phr-name⟩ string (1) as a variable atomic node if the semantic rule is used in the context of a RULE-STMT and (2) as a constant atomic node, otherwise. If a slot-filler constituent is specified in a semantic rewrite rule, but is missing from the surface string to which the rule is being applied, a default representation of the slot-filler corresponding to the missing constituent is established in the form of an atomic node about which the system knows nothing, other than its being a participant in the case frame (as in the previous section for a CASE-FRAME-DEFINITION).

Consider the following example CASE-SLOT-DEFINITION:

PROPERTY-CLASS-INDICATOR >> [MEMBER ADJECTIVE PROPERTY-CLASS +]

According to this rule, a PROPERTY-CLASS-INDICATOR should have an ADJECTIVE constituent and the interpretation of a string of the PROPERTY-CLASS-INDICATOR category would be represented by an atomic node which fills the PROPERTY-CLASS slot of a case frame whose MEMBER slot is filled by the interpretation of the ADJECTIVE constituent. The mapping from a surface string to the network representation of its interpretation is illustrated in Figure 12

In order to prepare for the example discussed in Sect. 3, the following rule is input:

INDEF-S-PHRASE >> [BSTR STRING CAT S-CAT STRC+]

According to this rule, the interpretation of a string parsed by the system as an INDEF-S-PHRASE would be represented by an atomic node filling the STRC slot of a case frame whose CAT slot is filled by the interpretation of the S-CAT constituent of the INDEF-S-PHRASE and whose BSTR slot is filled by the STRING constituent. This is illustrated in Figure 13.

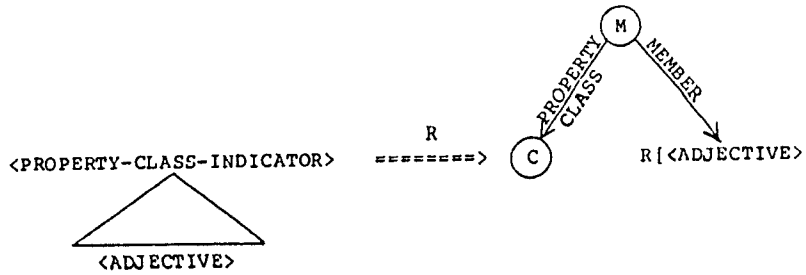
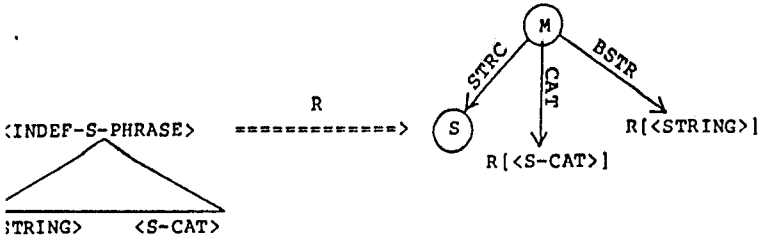
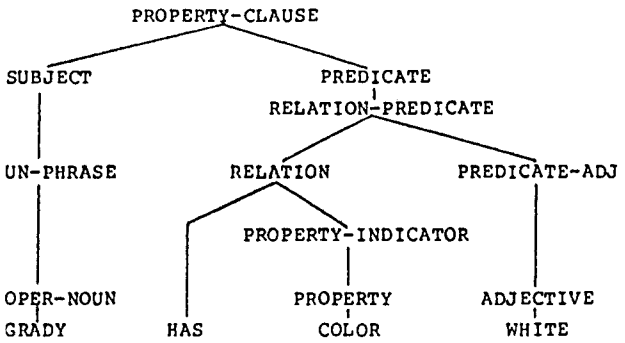


Fig. 12. Representational mapping applied to a PROPERTY-CLASS-INDICATOR



g. 13. Representational mapping applied to an INDEF-S-PHRASE



g. 14. Parse tree for sample input utterance

6 *Use in Language Processing*

illustrate the system's use of the language definition developed via the rewrite es of the preceding sections, we show some sentences of this language which er to the language itself compared with some that refer to a non-linguistic main. Thinking affectionately of her pet geese, the teacher informs the system it "GRADY HAS COLOR WHITE". The system recognizes and builds the rse tree of Figure 14 for the utterance. We show the more conventional form of parse tree, rather than the equivalent network parse tree that the system actu- y builds in order to simplify the figure.

In the preceding sections, the teacher has entered rewrite rules into the system define the semantics for certain string classes (e.g., PROPERTY-CLAUSE, OPERTY-INDICATOR), thereby identifying the conceptually coherent constit- nts for the language definition. The system applies these semantic rewrite rules d builds the structure of Figure 15 as the interpretation of the utterance. The ertion that the parsed input utterance expresses the concept represented by de M75 is also established in the network.

Similarly, the system can process the input utterance "'GOOSE HAS NUM- ER SINGULAR". The resulting parse tree is shown in Figure 16.

The representation of the interpretation of the utterance is shown in Figure 17.

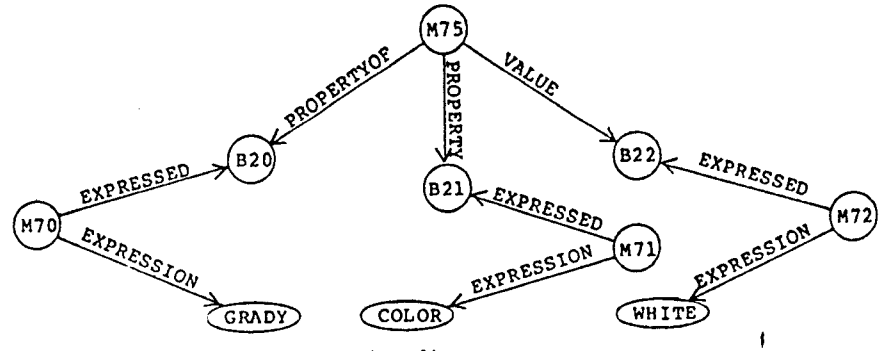


Fig. 15. Representation of the interpretation of input utterance

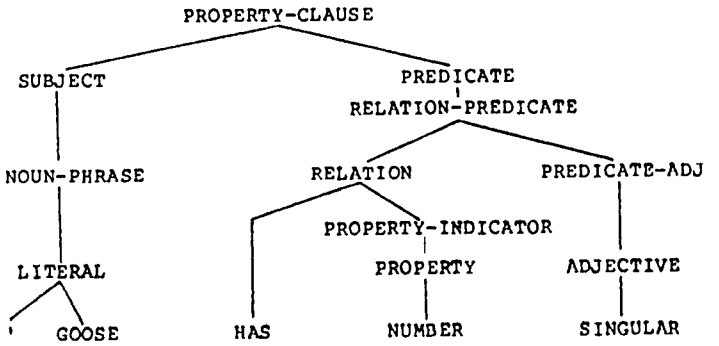


Fig. 16. Parse tree for utterance concerning language

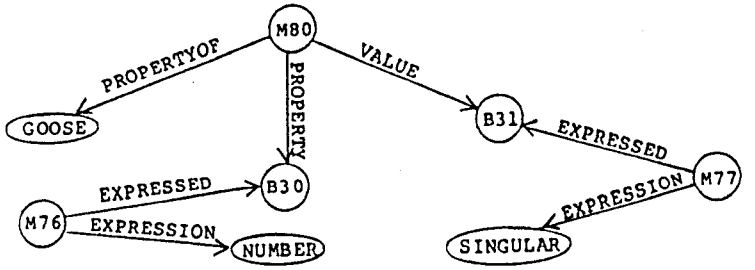


Fig. 17. Representation of interpretation of utterance

As stated in Section 1.5, the NL system distinguishes between a word or phrase and its interpretation. The interpretation of a LITERAL is the word following the quote mark (the word 'GOOSE, in this case). Thus node M80 represents the proposition that singular number is a property of the word 'GOOSE and not of the concept expressed by the word 'GOOSE. On the other hand, the interpretation of the word 'GRADY is represented by node B20 of Figure 15, and it is this entity

that has color white. Comparing these two examples illustrates the knowledge representations we have established as well as the capability for handling strings and their interpretations as domain knowledge, which is fundamental to our theory and system.

At this stage, the teacher can simplify the language to use with the system for expressing properties. She does this by inputting the following rewrite rules so that property class entries can be made.

PROPERTY-CLASS-ENTRY → ADJECTIVE 'IS 'A PROPERTY
PROPERTY-CLASS-ENTRY :: MEMBER ADJECTIVE
PROPERTY-CLASS PROPERTY

Since the system has previously been informed that 'WHITE is an ADJECTIVE and 'COLOR is a PROPERTY, the utterance "WHITE IS A COLOR" would be recognized by the system as a PROPERTY-CLASS-ENTRY. Also since, in Section 2.5.3, for the purposes of this example, the teacher entered 'ADJECTIVE and PROPERTY into UNIQUE-MEANING-CAT, the surface strings that are in the categories ADJECTIVE and PROPERTY are each treated as having a unique interpretation. Thus different instances of the same string such as 'WHITE are created by the system as having the same interpretation and it uses just one network structure to represent this interpretation. Therefore, using the above semantic rewrite rule, the system builds the structure of node M85 of Figure 18 to represent the interpretation of the utterance, finding the nodes B21 and B22 of Figure 15 to represent the interpretation of 'COLOR and 'WHITE, respectively.

Similarly, the system can be informed that "PLURAL IS A NUMBER" and it builds a structure similar to that of Figure 18 to represent the assertion that the concept expressed by 'PLURAL is a member of the property-class NUMBER.

If the utterance "GLADYS IS WHITE" is now input to the system, the utterance is also recognized as a PROPERTY-CLAUSE as shown in Figure 19.

The semantic rewrite rules of the previous section are used by the system to build the structure dominated by node M90 as the representation of the interpretation of the utterance.

According to the semantic rule for a PROPERTY-CLAUSE, the PROPERTY slot in the case frame is filled by the interpretation of the PROPERTY-INDICATOR constituent of the utterance. Referring to the parse tree of Figure 19, the PROPERTY-INDICATOR consists of the PROPERTY-CLASS-INDICATOR. The semantic rule

PROPERTY-INDICATOR: : PROPERTY-CLASS-INDICATOR

of the previous section instructs the system to use the interpretation of the PROPERTY-CLASS-INDICATOR as the interpretation of the PROPERTY-INDICATOR. The rule for interpreting a PROPERTY-CLASS-INDICATOR is the CASE-SLOT-DEFINITION presented in the previous section:

PROPERTY-CLASS-INDICATOR > > [MEMBER ADJECTIVE PROPERTY-CLASS +]

This rule instructs the system to interpret the PROPERTY-CLASS-INDICATOR as the PROPERTY-CLASS slot-filler of the frame whose MEMBER SLOT is

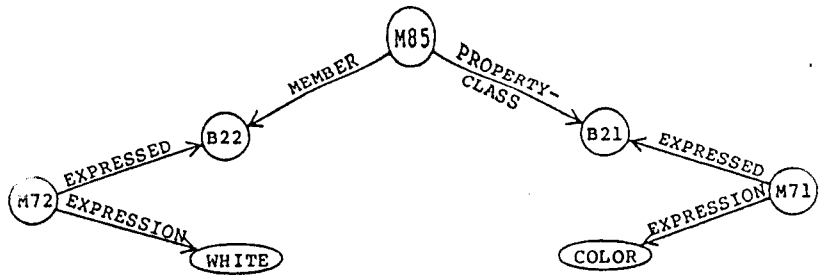


Fig. 18. Representation of interpretation of utterance

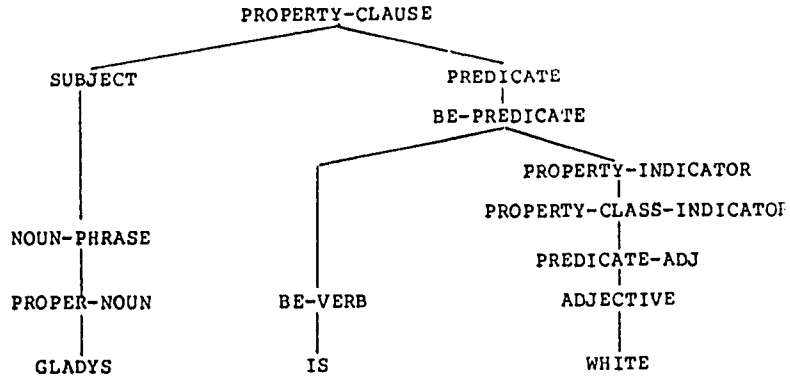


Fig. 19. Parse tree for input utterance

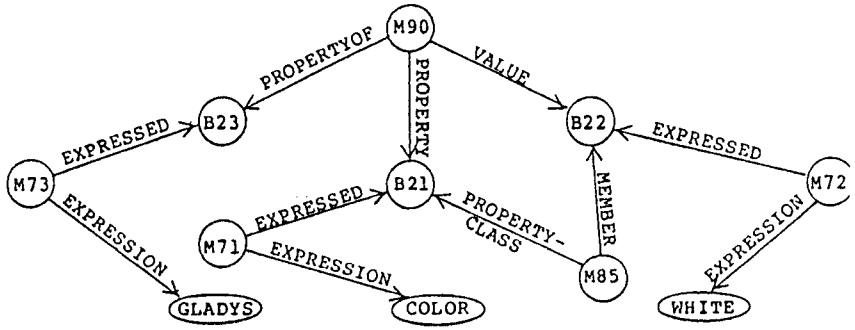


Fig. 20. Representation of interpretation of utterance

filled by WHITE. Node B22 of Figures 15 and 18 is found as the representation of the interpretation of 'WHITE, since the members of the class ADJECTIVE have been defined by the teacher as having "unique semantics". Thus, the system uses node B22 to use as the MEMBER slot-filler for the case frame associated with a PROPERTY-CLASS-INDICATOR. Then B21 of Figure 15 is found and used as the PROPERTY-CLASS slot-filler as shown in Figure 20, since it represents the

PROPERTY-CLASS that has WHITE as a MEMBER. Node B21 is also the representation of the interpretation of the PROPERTY-CLASS-INDICATOR string. In general, a CASE-SLOT-DEFINITION maps a surface string to a participant in a relation or proposition.

In a manner similar to the parsing and interpretation of the utterance "GLADYS IS WHITE", the system also understands the utterance "'GEESE IS PLURAL". The system's language definition is again used as a metalanguage to expand upon the same language itself.

3 Increasing the System's Language Capability Through Its Language Capability

3.1 Motivation

Since we treat linguistic knowledge as domain knowledge, the system teacher user) can add to the knowledge base and instruct the system as to how to process or understand ever more sophisticated language.

Just as a person is continually influenced by interaction with his environment, he data base of our system is modified by each input. The knowledge base is ncrementally enhanced to form a more sophisticated system.

Since we represent language processing knowledge in the same knowledge base nd in the same formalism as other domain knowledge, it is possible to make the ystem's language processing knowledge the subject of its language processing nd this is a fundamental aspect of our approach. Thus by instructing the system n the domain of linguistics as we would expect to be able to do with another omain in an interactive NLU system, we can increase the system's language apability through its language capability. A user can communicate with our sys-em in just one language via one processor without switching "modes" or interact-ng with supportive processors in special purpose languages.

The rewrite rules of the KL are certainly not sufficient for expressing all the ules a teacher would need to define a language of her choice. Therefore, one of ie most important capabilities that the system needs is to understand a more gen-al form of rule statement. A teacher should be able to bootstrap into a more owerful rule statement language from the KL. In the next sections, we present an xample from such a bootstrap process.

2 Defining More-General Rule Forms

ne teacher first extends the system's language definition so that it can begin to nderstand general "IF-THEN" rules. As stated in Section 2.2.1, RULE-STMT is predefined category. The syntax of a RULE-STMT is not predefined, but for the terpretation process, each RULE-STMT must have an ANT-CLAUSE and a Q-CLAUSE constituent. The ANT-CLAUSE constituent is interpreted as the ntecedent of the rule and the CQ-CLAUSE constituent as the consequent of the le. Thus the rewrite rule

RULE-STMT → 'IF ANT-CLAUSE 'THEN CQ-CLAUSE

defines a syntax for the RULE-STMT. The syntax and semantics of ANT-CLAUSE and CQ-CLAUSE must also be defined. This was done is Section 2.5 (see Appendix).

The additional rules that the teacher chooses to input to the system to increase its capability of understanding linguistic-domain language for this example are listed below. In order to make use of the system's ability to use a VARIABLE as an appositive to another phrase and remember the association of the VARIABLE to the prase, the teacher inputs:

DEF-S-PHRASE → DEF-DET S-CAT
INDEF-S-PHRASE → INDEF-DET S-CAT
MAIN-APPOS-PHR VAR-NAME → INDEF-S-PHRASE VARIABLE
VAR-APPOSITION-PHR → MAIN-APPOS-PHR VAR-NAME

To explain to the system how to parse and interpret language which describes one phrase being a constituent of another, the teacher inputs:

SUP-STRING-REF → VAR-APPOSITION-PHR
CONSTIT-REF → DEF-S-PHRASE
CONSTIT-PHRASE → CONSTIT-REF 'OF SUP-STRING-REF
NOUN-PHRASE → CONSTIT-PHRASE
    CONSTIT-PHRASE> >[CONSTIT + CONSTITOF SUP-STRING-REF]
    [BSTR STRING CAT DEF-S-PHRASE STRC +]
SUP-STRING-REF :: VAR-APPOSITION-PHR
MAIN-APPOS-PHR :: INDEF-S-PHRASE
NOUN-PHRASE :: CONSTIT-PHRASE
DEF-S-PHRASE :: S-CAT

These rules will be used in the next sections.

3.3 Parsing Strategy

The parsing strategy applied by our NL system is a combined bottom-up, top-down strategy. As each word of an input string is read by the system, the network representation of the string is extended as discussed in Section 2.3 and relevant rules stored in the SNePS network are triggered. All applicable rules are started in parallel in the form of processes created by our MULTI-processing package (McKay and Shapiro, 1980). These processes are suspended if not all their antecedents are satisfied and are resumed if more antecedents are satisfied as the reading of the string proceeds. As parsing proceeds, the annotated parse (tree(s) for an input utterance is (are) represented in the system's network knowledge base. Our system builds and retains network structures corresponding to alternative analyses of a given input string. Retention of the alternatives avoids the reanalysis of previously processed surface strings that occurs in a backtracking system.

Processing is controlled by the SNePS Inference Package (Shapiro et al., 1982), which employs bi-directional inference. This is a form of inference resulting from

interaction between forward and backward inference and loosely corresponds to bi-directional search through a space of inference rules. This technique focuses attention towards the active parsing processes and prunes the search through the space of inference rules by ignoring rules which have not been activated. This cuts down the fan out of pure forward or backward chaining. New rules are activated only if no active rules are applicable.

Consider the sample input utterance "IF THE HEAD-NOUN OF A NOUN-PHRASE X HAS NUMBER Y THEN X HAS NUMBER Y". When the first word is read by the system, it is recognized as matching the word 'IF in the rules

RULE-STMT → 'IF ANT-CLAUSE 'THEN CQ-CLAUSE
'IF ANT-CLAUSE → 'IF PROPERTY-CLAUSE

and parsing begins in a bottom-up manner. Both rules are triggered in parallel by the SNePS MULTI package. When originally input, each of the above rules was interpreted by the system and stored in the form of a network rule which we paraphrase as follows (NOTE: In all the paraphrased rules of this section, V_1 and V_2 are universally quantified variables):

(1) If a word of an input string is the word 'IF, then
(2) if V_1 follows the word 'IF and V_1 is an ANT-CLAUSE, then
(3) if the word 'THEN follows V_1, then
(4) if V_2 follows the word 'THEN and V_2 is a CQ-CLAUSE,
then the string consisting of 'IF followed by V_1
followed by 'THEN followed by V_2 is a RULE-STMT.
(5) If a word of an input string is the word 'IF, then
(6) if V_1 follows the word 'IF and V_1 is a PROPERTY-CLAUSE,
then V_1 is an ANT-CLAUSE.

(The numbers in parentheses are rule numbers, not line numbers. Thus, for example, nested rule (3) begins with "if the word 'THEN" and continues to the period at the end of the sentence.)

Since the antecedent of rule (1) above is satisfied, the system questions whether a string immediately following the word 'IF is an ANT-CLAUSE. When a SNePS rule is triggered, a process is created forming the active version of the rule for the purpose of such activities as data collection and variable binding. Some of these processes act as demons, waiting for instances of their antecedents so that instances of their consequents can be deduced. This is the case for the nested rule (2). Since no string follows the word 'IF yet, the process for rule (2) is suspended.

These active processes, with their communication links, form the equivalent of a hypothesized parse tree with associated expectations. The inference system ignores unactivated rules as long as there are applicable active rule processes awaiting data, essentially parsing in a top-down manner in this situation. The hypothesized parse tree corresponding to the process of rule (2) is illustrated in Figure 21. The parsing strategy of our system is similar to "left-corner bottom-up parsing" (Burge, 1975) in that construction of a parse tree begins at the bottom left corner, processing of a surface string proceeds in a left-to-right manner, and whenever an initial segment of a string has been parsed, the system attempts to establish a goal analysis of the string or substring thereof. In the following figures, the bro-

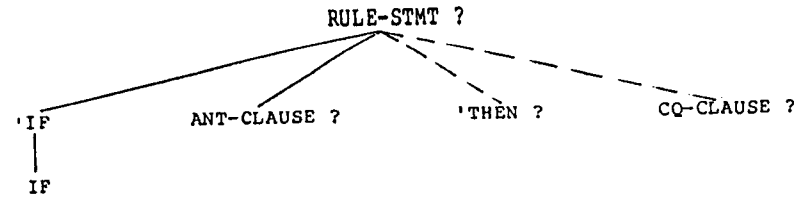


Fig. 21. Hypothesized parse tree

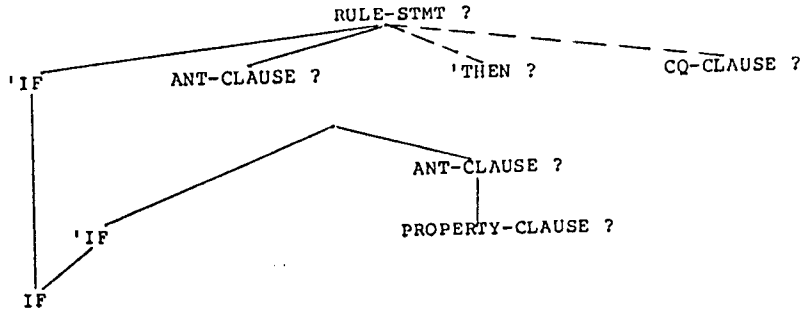


Fig. 22. Hypothesized parse tree

ken lines indicate goals or expectations represented by antecedents of nested rules for which active demons have not yet been created. The question-marks indicate expectations which have not yet been satisfied.

The antecedent of rule (5) is also satisfied. This is a context sensitive rule which constrains the parsing process. According to this rule, a PROPERTY-CLAUSE is parsed as an ANT-CLAUSE in the context of the word 'IF. A process is created forming the active version of rule (6) and this process awaits a PROPERTY-CLAUSE following the word 'IF. Figure 22 reflects the current state of the system in terms of its active processes, implicit expectations, and the tokens that it has consumed.

When the word 'THE is read by the system, the rule

DEF-S-PHRASE → DEF-DET S-CAT

is triggered as parsing continues in a bottom-up manner. This rule is paraphrased as:

(7) If V_1 is a DEF-DET, then
(8) if V_2 follows V_1 and V_2 is an S-CAT,
then the string consisting of V_1 followed by V_2 is a
DEF-S-PHRASE.

The antecedent of rule (7) is satisfied and a process is created for nested rule (8) to await an S-CAT following the DEF-DET. The active processes form another hypothesized parse tree shown in Figure 23.

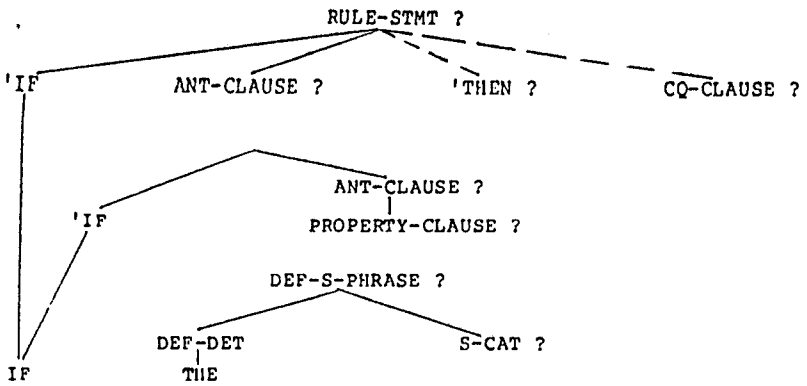


Fig. 23. Hypothesized parse tree

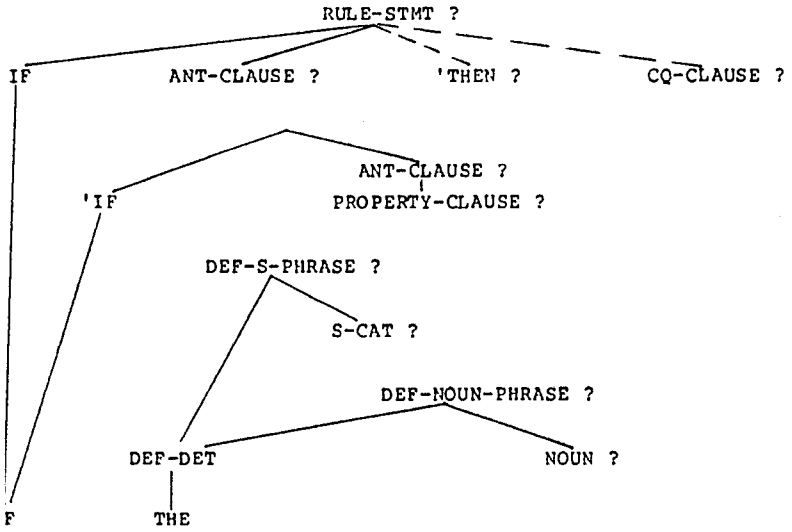


Fig. 24. Hypothesized parse trees

Suppose another rule such as DEF-NOUN-PHRASE → DEF-DET NOUN had been entered by the teacher and is present in the network knowledge base. This rule is paraphrased as:

(9) If V_1 is a DEF-DET, then
(10) if V_2 follows V_1 and V_2 is a NOUN, then the string
consisting of V_1 followed by V_2 is a DEF-NOUN-PHRASE.

This latter rule is also triggered by the system's reading of the word 'THE and the processes created for rules (9) and (10) form another set of hypothesized parse trees as illustrated in Figure 24.

The parse trees of Figure 24 dominated by DEF-S-PHRASE? and DEF-NOUN-PHRASE? represent alternative possibilities for the parse of the string beginning with the word 'THE. A process such as the process for rule (10) waiting for a NOUN may remain suspended indefinitely if the expected data is not forthcoming.

When the next word 'HEAD-NOUN is read, the system recognizes it as an S-CAT and the process corresponding to rule (8) is resumed since it is waiting for an S-CAT following the word 'THE. Thus the string "THE HEAD-NOUN" is recognized as a DEF-S-PHRASE by application of the teacher's rules. This DEF-S-PHRASE then triggers the network version of the following rule and the DEF-S-PHRASE is then recognized as a CONSTIT-REF:

CONSTIT-REF → DEF-S-PHRASE

Recognition of a CONSTIT-REF triggers the rule

CONSTIT-PHRASE → CONSTIT-REF 'OF SUP-STRING-REF

whose network representation can be paraphrased as

(11) If V_1 is a CONSTIT-REF, then
(12) if the word 'OF follows V_1, then
(13) if V_2 follows the word 'OF and V_2 is a SUP-STRING-REF,
then the string consisting of V_1 followed by the word 'OF followed by V_2 is a CONSTIT-PHRASE.

Activation of rule (11) is analogous to bottom-up processing again. A process is established for rule (12) to await the word 'OF in the input stream.

When the next word 'OF is read by the system, the demon corresponding to rule (12) is activated and since the antecedent of rule (12) is satisfied, a process is established for rule (13) to expect a SUP-STRING-REF following the word 'OF. No other rules are activated by the reading of the word 'OF since an active process was waiting for this word in the input stream.

The system parses the next string "A NOUN-PHRASE" as an INDEF-S-PHRASE by application of the rule

INDEF-S-PHRASE → INDEF-DET S-CAT

This triggers the rule

MAIN-APPOS-PHR VAR-NAME → INDEF-S-PHRASE VARIABLE

which is paraphrased as

(14) If V_1 is an INDEF-S-PHRASE, then
(15) if V_2 follows V_1 and V_2 is a VARIABLE,
(16) then V_1 is a MAIN-APPOS-PHR and V_2 is a VAR-NAME.

Since the antecedent of rule (14) is satisfied, a process is set up for rule (15). When the next word 'X is read, it is recognized as a VARIABLE and since the active process for rule (15) is waiting for a VARIABLE, no unactivated rules are applied. An example of such an unactivated rule is

NOUN-PHRASE → VARIABLE

hich we previously input to the system. Thus an alternative parse is blocked by ıe expectation of a VARIABLE by the process for rule (15). By application of the lles

VAR-APPOSITION-PHR → MAIN-APPOS-PHR VAR-NAME
SUP-STRING-REF → VAR-APPOSITION-PHR

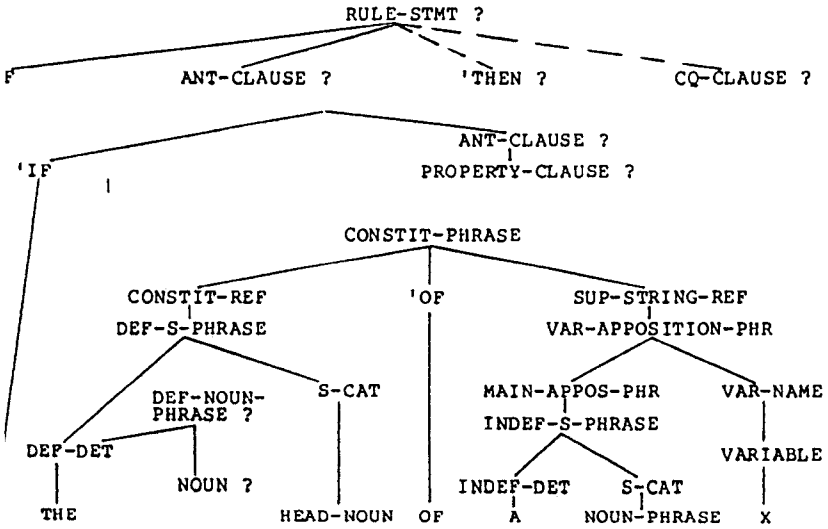
.e expected SUP-STRING-REF of rule (13) is satisfied and the string "THE EAD-NOUN OF A NOUN-PHRASE X" is parsed as a CONSTIT-PHRASE. y application of the rule

NOUN-PHRASE → CONSTIT-PHRASE

e string is also recognized as a NOUN-PHRASE. Notice that the term NOUN-HRASE is *mentioned* in the input string and *used* in the application of the above le.

At this point in the parsing process the hypothesized parse trees are illustrated Figure 25.

As parsing proceeds using the rules introduced in this and preceding sections of is chapter, the resulting parse of the entire input statement is shown in Figure 26. ıe string category identifiers in the tree that are underlined are the categories at are included in the domain of the representational mapping. These are the tegories for which the teacher has defined a rule to determine the interpretation any member of the category (i.e., the underlining identifies the string categories fined by the teacher as the conceptually coherent constituents of the utter-ce).



.25. Hypothesized parse trees

In this section on parsing, we have illustrated the following characteristics of our system's strategy:

(1) the parallel processing of applicable rules;
(2) constraint of the parsing process by the use of context sensitive rules;
(3) constraint of the parsing process by the SNePS Inference Package focusing on active rule processes - the manifestation being the blocking of multiple parses by previously established expectations;
(4) suspension and resumption of rule processes during the parsing process.

The retention of alternative analyses of a string, which avoids the reanalysis of certain strings in the case of a backtracking system, was not illustrated by the example of this section, but is a characteristic of our system.

Also of importance in this section is the fact that the system is again using its acquired language definition as a metalanguage to understand another instruction from the teacher concerning the language itself.

3.4 Interpretation of the Input Rule Statement

During the interpretation process, a VARIABLE of the user's language is translated into a variable node of the semantic network. The scope of a user VARIABLE is the utterance in which it occurs. The association of a user VARIABLE to its interpretation is maintained on a list only during translation of the utterance in which the VARIABLE occurs.

The interpretation of a user VARIABLE is as follows: If a VARIABLE is used as the VAR-NAME of a VAR-APPOSITION-PHR, discussed briefly in Section 2.2.1, then the system uses the interpretation of the MAIN-APPOS-PHR as the interpretation of the VARIABLE, and stores this association on the variable association list. Otherwise, the system checks the variable association list for a corresponding interpretation already established. Otherwise, a new variable node is created as the interpretation of the user VARIABLE, the new pair once again being added to the variable association list.

As shown in Figure 25, the phrase "A NOUN-PHRASE X" was recognized by the system as a VAR-APPOSITION-PHR, with "A NOUN-PHRASE" recognized as the MAIN-APPOS-PHR and 'X as the VAR-NAME. Thus the interpretation of the phrase "A NOUN-PHRASE" is remembered by the system as the interpretation of 'X. The string "A NOUN-PHRASE" has been recognized as an INDEF-S-PHRASE and thus the semantic rule

INDEF-S-PHRASE > > [BSTR STRING S-CAT STRC +]

of Section 2.5.3 applies. As discussed in Section 2.5.3, if a constituent is mentioned in a semantic rule but is missing from the surface string to which the rule applies, then the system represents the interpretation of the constituent as an atomic node. Furthermore, this atomic node is a variable node in the context of a RULE-STMT. Since the slot-filler constituent of category STRING is not present in our example INDEF-S-PHRASE, an atomic variable node (V2 of Fig. 26) is built to represent

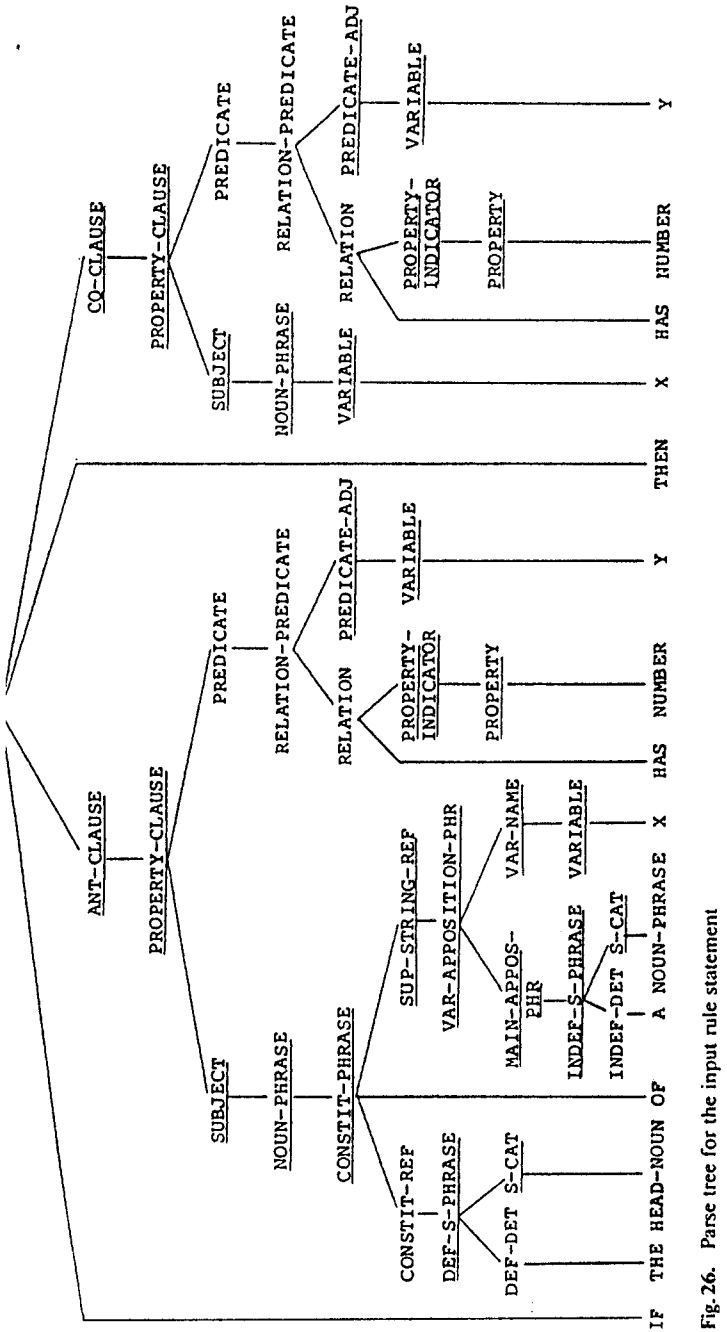


Fig. 26. Parse tree for the input rule statement

the interpretation of the missing STRING constituent. The representation of the interpretation of the S-CAT constituent "NOUN-PHRASE" is node B25, representing the category of NOUN-PHRASEs. The STRC slot-filler becomes the interpretation of the INDEF-S-PHRASE. This slot-filler is also represented by an atomic variable node (V1 of Fig. 26) as explained in Sect. 2.5.3. The + symbol in the rewrite rule marks the participant of the proposition represented by the case frame whose representation is also the representation of the interpretation of the INDEF-S-PHRASE. Thus the interpretation of the INDEF-S-PHRASE "A NOUN-PHRASE" is node V1 of Figure 26. That is, the INDEF-S-PHRASE is interpreted as a variable node to be instantiated by a structure representing an analyzed surface string which has an associated bounded-string (see Sect. 2.2.3) and category NOUN-PHRASE. V1 is also the interpretation of user VARIABLE 'X due to the string "A NOUN-PHRASE X" being a VAR-APPOSITION-PHR and the association of V1 and 'X is stored on the variable association list.

The input string "THE HEAD-NOUN OF A NOUN-PHRASE X" was parsed as a CONSTIT-PHRASE (refer to Fig. 26). The rule for interpreting a CONSTIT-PHRASE was given in Section 3.3 as

CONSTIT-PHRASE > > [CONSTIT + CONSTITOF SUP-STRING-REF]
[BSTR STRING CAT DEF-S-PHRASE STRC +]

This rule stipulates that the interpretation of a CONSTIT-PHRASE is a participant in two case frames as defined in the two sets of brackets and the + symbol marks the slot-filler that is the interpretation of the CONSTIT-PHRASE. Again an atomic variable node is built to represent this slot-filler which also represents the CONSTIT-PHRASE. The SUP-STRING-REF ist the constituent "A NOUN-PHRASE X" (refer to Fig. 26), whose interpretation is represented by node V1 of Figure 27. The structure representing the case frame defined in the second set of brackets is built in a manner similar to that used in building the structure of Figure 27 and described above.

The interpretation of the example CONSTIT-PHRASE "THE HEAD-NOUN OF A NOUN-PHRASE X" is represented by node V3 of Figure 28. The node V1 of Figure 28 ist the same node as V1 of Figure 27.

Assembling the interpretations of the constituents of our RULE-STMT from Figures 27 and 28 and completing the interpretation of the RULE-STMT as the system does using the semantic rewrite rules of this article, node M86 of Figure 29 represents the interpretation of the RULE-STMT. All of the variable nodes V1, V2,

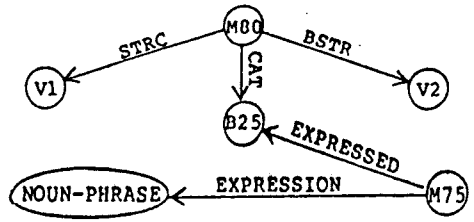
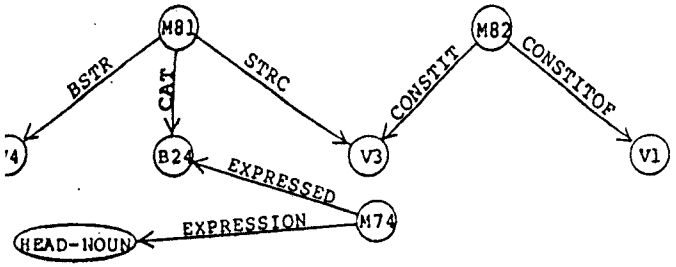
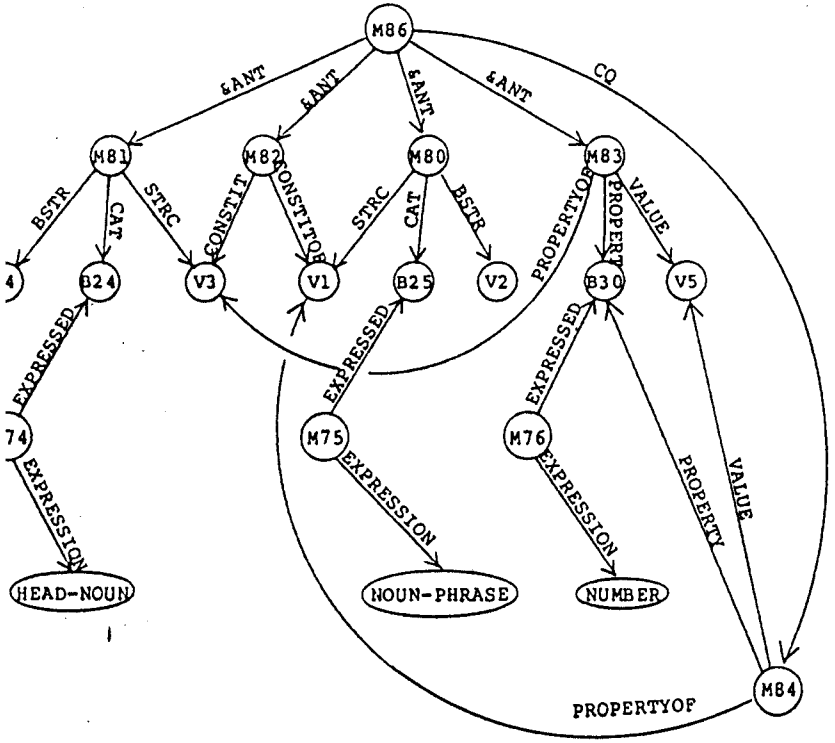


Fig. 27. Node V1 represents the interpretation of "A NOUN-PHRASE"



g. 28. Node V3 represents the interpretation of the CONSTIT-PHRASE



g. 29. Node M86 represents the interpretation of the input rule

), V4, and V5 are universally quantified (refer to Shapiro (1979a) for the network presentation of the quantification that is not shown in the figure and for more tails on the rule structures of SNePS). The &ANT and CQ arcs are the SNePS stem arcs used in the network representation of "&-entailment", the entailment any of a set of consequents by the conjunction of one or more antecedents.

4 Language Use-Mention Distinction

In order for our system to treat linguistic knowledge as domain knowledge and to receive instruction in the use of this knowledge, it is essential for the system to distinguish between use and mention of language (Quine, 1951). We have already seen examples of this capability in our system. When words are entered into their appropriate lexical categories as in Section 2.5.2a, they are *mentioned*. Those lexemes that are themselves names of lexical categories are subsequently *used* to refer to their corresponding lexical categories. For example, the word 'VERB is *mentioned* when entered into the category L-CAT of lexical category names and is subsequently *used* to refer to the category of verbs (see Sect. 2.5.2). The word 'GOOSE is *mentioned* in the example sentence of Section 2.6 when specifying that its number is singular, but, in a similar sentence, the word 'GRADY is *used*.

As a more sophisticated example combining use and mention, we illustrate our system's processing of an equivalent version of the classic sentence of Tarski (1944) "'SNOW IS WHITE' IS TRUE IF AND ONLY IF SNOW IS WHITE". We do not treat truth relative to possible worlds. Our semantic network represents only the belief space of the system, and asserted propositions are those believed by the system.

We continue to build upon the language definition thus far input to the system in this article. The additional lexical entries that we input are:

L-CAT → 'MASS-NOUN ADJECTIVE → 'TRUE
PROPERTY → 'TRUTH-VALUE ADJECTIVE → 'FALSE
MASS-NOUN → 'SNOW

We explain to the system that

TRUE IS A TRUTH-VALUE
FALSE IS A TRUTH-VALUE

to be parsed and interpreted by the system as PROPERTY-CLASS-ENTRIES as shown in Section 2.6. Additional syntax rules such as the following are needed:

NOUN-PHRASE → MASS-NOUN
NOUN-PHRASE → LITERAL-STRING

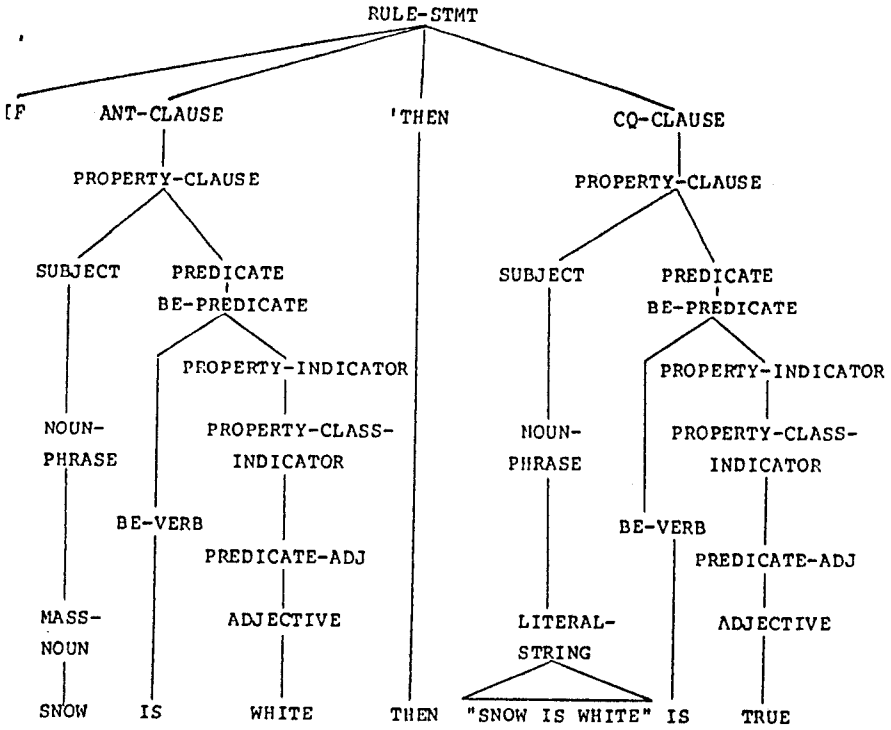
Upon input of the sentence

If SNOW IS WHITE THEN "SNOW IS WHITE" IS TRUE

the system builds the parse tree shown in Figure 30 for the utterance.

Applying the teacher's rules, the system builds the network rule of Figure 31 as the interpretation of the input sentence. Node M92 represents the generic string "SNOW IS WHITE" and not just an instance of the string. Node M92 dominates a pattern that is matched by any instance of the string, with V8 a universally quantified variable node.

If the system believes that snow is white, then the rule shown in Figure 31 is used appropriately and if we query the system regarding any instance of the string "SNOW IS WHITE" it indicates that the string is true.



.30. Annotated parse tree

complete the original bi-conditional statement, the converse statement

IF "SNOW IS WHITE" IS TRUE THEN SNOW IS WHITE

ı also be entered the system and the converse of the rule of Figure 31 is built ɔ the network as its interpretation.

Summary

s article has presented our approach to NLU: an approach that focuses on the ability of a natural langauge to be used as its own metalanguage. It is essential his approach to have the system's parsing and linguistic knowledge be an inte- part of its domain knowledge. It is our view that linguistic knowledge about a d or phrase is a part of its meaning or significance and, furthermore, there is clear boundary line separating syntactic, semantic, and world knowledge. For e reasons we represent linguistic knowledge along with other domain knowl- e in an integrated knowledge base. Furthermore, the linguistic rules of the sys- 's knowledge base comprise the system's knowledge of language understand- in the same way that the rules of any rule-based system comprise that system's

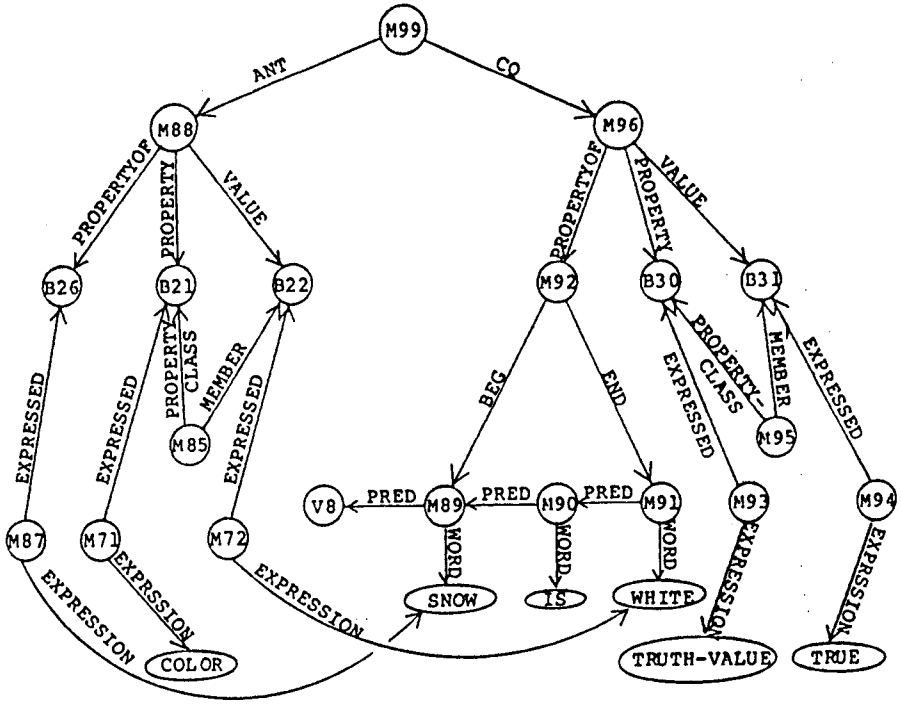


Fig. 31. Interpretation of the input utterance "IF SNOW IS WHITE THEN 'SNOW IS WHITE' IS TRUE"

knowledge of its domain of application. Our system also incorporates the use-mention distinction for language.

We are exploring the possibility of a NLU system's becoming more adept in its use of some language by being instructed in the use of the language. We wish this explanation to be given in an increasingly sophisticated subset of the language being taught. The system must start with some language facility, and we are interested in seeing how small and theory-independent we can make the initial kernel language.

In this chapter, we have discussed the core knowledge and representations of our system, including the kernel language, which consists of predefined terms, and syntactic and semantic rewrite rules with which to bootstrap into a more sophisticated language definition. We have demonstrated the capability of increasing the system's language facility by using the very same facility to instruct the system about language understanding. We built up the system's capability to the stage at which it processed the sentence "IF THE HEAD-NOUN OF A NOUN-PHRASE X HAS NUMBER Y THEN X HAS NUMBER Y". We presented additional examples of language being treated as the topic of discourse including the system's parsing and interpretation of the sentence "'SNOW IS WHITE' IS TRUE IF AND ONLY IF SNOW IS WHITE".

We discussed the system's parsing strategy, which is a combined bottom-up, p-down strategy. Our system's parser is a general rule-based inference system in nich applicable rules are activated in parallel in the form of processes or :mons. The inference system employs bi-directional inference to cut down the n out of pure forward or backward chaining.

:knowledgements. We would like to thank the SNePS Research Group of the ate University of New York at Buffalo for their constructive comments and inions during the course of this research. In particular, we thank William J. Raport for his comments on an earlier version of this chapter.

ppendix *Chronological Summary of Input to the System as Presented in This Chapter*

tion mber	Input
.2	L-CAT → 'NOUN
2	L-CAT → 'PROPER-NOUN
2	L-CAT → 'DEF-DET
2	L-CAT → 'INDEF-DET
2	L-CAT → 'VERB
2	L-CAT → 'BE-VERB
2	L-CAT → 'ADVERB
2	L-CAT → 'ADJECTIVE
2	L-CAT → 'PREPOSITION
2	L-CAT → 'CONJ
2	L-CAT → 'PROPERTY
2	S-CAT → 'HEAD-NOUN
2	S-CAT → 'STRING
2	VARIABLE → 'X
2	VARIABLE → 'Y
2	NOUN → 'GOOSE
2	NOUN → 'GEESE
2	PROPER-NOUN → 'GRADY
2	PROPER-NOUN → 'GLADYS
2	DEF-DET → 'THE
2	INDEF-DET → 'A
2	VERB → 'HAS
2	BE-VERB → 'IS
2	ADVERB → 'THEN
2	ADJECTIVE → 'WHITE
2	ADJECTIVE → 'SINGULAR
2	ADJECTIVE → 'PLURAL
2	PREPOSITION → 'OF
2	CONJ → 'IF
2	PROPERTY → 'COLOR
2	PROPERTY → 'NUMBER
.2	UNIQUE-MEANING-CAT → 'ADJECTIVE
.2	UNIQUE-MEANING-CAT → 'PROPERTY
2	PROPERTY-CLAUSE → SUBJECT PREDICATE

Section Number	Input
2.5.2	SUBJECT → NOUN-PHRASE
2.5.2	NOUN-PHRASE → LITERAL
2.5.2	NOUN-PHRASE → VARIABLE
2.5.2	NOUN-PHRASE → PROPER-NOUN
2.5.2	PREDICATE → RELATION-PREDICATE
2.5.2	PREDICATE → BE-PREDICATE
2.5.2	RELATION-PREDICATE → RELATION PREDICATE-ADJ
2.5.2	BE-PREDICATE → BE-VERB PROPERTY-INDICATOR
2.5.2	RELATION → 'HAS PROPERTY-INDICATOR
2.5.2	PROPERTY-- INDICATOR → PROPERTY-CLASS-INDICATOR
2.5.2	PROPERTY-INDICATOR → PROPERTY
2.5.2	PROPERTY-CLASS-INDICATOR → PREDICATE-ADJ
2.5.2	RULE-STMT → 'IF ANT-CLAUSE 'THEN CQ-CLAUSE
2.5.2	RELATION PREDICATE-ADJ → RELATION ADJECTIVE
2.5.2	RELATION PREDICATE-ADJ → RELATION VARIABLE
2.5.2	BE-VERB PREDICATE-ADJ → BE-VERB ADJECTIVE
2.5.2	'IF ANT-CLAUSE → 'IF PROPERTY-CLAUSE
2.5.2	'THEN CQ-CLAUSE → 'THEN PROPERTY-CLAUSE
2.5.3	PROPERTY-CLAUSE : : PROPERTY SUBJECT PROPERTY PROPERTY-INDICATOR VALUE PREDICATE-ADJ
2.5.3	ANT-CLAUSE : : PROPERTY-CLAUSE
2.5.3	CQ-CLAUSE : : PROPERTY-CLAUSE
2.5.3	SUBJECT : : NOUN-PHRASE
2.5.3	NOUN-PHRASE: : LITERAL
2.5.3	NOUN-PHRASE: : VARIABLE
2.5.3	NOUN-PHRASE: : PROPER-NOUN
2.5.3	PROPERTY-INDICATOR : : PROPERTY-CLASS-INDICATOR
2.5.3	PROPERTY-INDICATOR : : PROPERTY
2.5.3	PREDICATE-ADJ : ADJECTIVE
2.5.3	PREDICATE-ADJ : : VARIABLE
2.5.3	PROPERTY-CLASS-INDICATOR > > {MEMBER ADJECTIVE PROPERTY-CLASS +}
2.5.3	INDEF-S-PHRASE > > {BSTR STRING CAT S-CAT STRC +}
2.6	GRADY HAS COLOR WHITE
2.6	'GOOSE HAS NUMBER SINGULAR
2.6	PROPERTY-CLASS-ENTRY → ADJECTIVE 'IS 'A PROPERTY
2.6	PROPERTY-CLASS-ENTRY : : MEMBER ADJECTIVE PROPERTY-CLASS PROPERTY
2.6	WHITE IS A COLOR
2.6	PLURAL IS A NUMBER
2.6	GLADYS IS WHITE
2.6	'GEESE IS PLURAL
3.2	DEF-S-PHRASE → DEF-DET S-CAT
3.2	INDEF-S-PHRASE → INDEF-DET S-CAT
3.2	MAIN-APPOS-PHR VAR-NAME → INDEF-S-PHRASE VARIABLE
3.2	VAR-APPOSITION-PHR → MAIN-APPOS-PHR VAR-NAME
3.2	SUP-STRING-REF → VAR-APPOSITION-PHR
3.2	CONSTIT-REF → DEF-S-PHRASE
3.2	CONSTIT-PHRASE→ CONSTIT-REF 'OF SUP-STRING-REF
3.2	NOUN-PHRASE → CONSTIT-PHRASE

ection Number	Input
.2	CONSTIT-PHRASE > > [CONSTIT + CONSTITOF SUP-STRING-REF] [BSTR STRING CAT DEF-S-PHRASE STRC +]
.2	SUP-STRING-REF : : VAR-APPOSITION-PHR
.2	MAIN-APPOS-PHR : : INDEF-S-PHRASE
.2	NOUN-PHRASE : : CONSTIT-PHRASE
.2	DEF-S-PHRASE : : S-CAT
3	IF THE HEAD-NOUN OF A NOUN-PHRASE X HAS NUMBER Y THEN X HAS NUMBER Y
	L-CAT → 'MASS-NOUN PROPERTY → 'TRUTH-VALUE MASS-NOUN → 'SNOW ADJECTIVE → 'TRUE ADJECTIVE → 'FALSE
	TRUE IS A TRUTH-VALUE FALSE IS A TRUTH-VALUE NOUN-PHRASE → MASS-NOUN NOUN-PHRASE → LITERAL-STRING
	IF SNOW IS WHITE THEN "SNOW IS WHITE" IS TRUE IF "SNOW IS WHITE" IS TRUE THEN SNOW IS WHITE

eferences

len, J. (1978): Anatomy of LISP. McGraw-Hill, New York

ver, T.G. (1970): The Cognitive Basis for Linguistic Structures. In: Hayes, J.R. (ed.): Cognition and the Development of Language. Wiley, New York pp.279-352

ver, T.G. (1973): Serial Position and Response Biases do Do Not Account for the Effect of Syntactic Structure on the Location of Brief Noises During Sentences. Journal of Psycholinguistic Research **2**, 187-288 (1973)

brow, R.J. (1978): The RUS System. BBN Report No.3878

brow, R.J. and Webber, B. (1980): Knowledge Representation for Syntactic/Semantic Processng. Proc. AAAI-80, pp.316-323

ıchman, R.J. (1977): What's in a Concept: Structural Foundations for Semantic Networks. Int. J. Man-Machine Studies **9**, 127-152 (1977)

ıchman, R.J. (1978a): A Structural Paradigm for Representing Knowledge. BBN Report No.3605

ıchman, R.J., Ciccarelli, E., Greenfeld, N., and Yonke, M. (1978b): KLONE Reference Manıal. BBN Report No.3848, July 1978

ıchman, R.J. (1979): On the Epistemological Status of Semantic Networks. In Findler, N. (ed.): Associative Networks. Academic Press, New York, pp.3-50

own, G. and Yule, G. (1983): Discourse Analysis. Cambridge University Press, Cambridge

ıce, B. (1975): Case Systems for Natural Language. Artificial Intelligence **6**, 327-360 (1975)

rge, W.H. (1975): Recursive Programming Techniques. Addison-Wesley, Reading

arniak, E. (1981): A Common Representation for Problem-Solving and Language-Comprehension Information. Artificial Intelligence **16** (3), 225-255 (1981)

hl, V. (1979): Quantification in a Three-Valued Logic for Natural Language Question-Answering Systems. Proc. IJCAI 79, pp.182-187

hl, V. (1981): Translating Spanish into Logic Through Logic. AJCL **7** (3), 149-164 (1981)

lmore, C. (1968) : The Case for Case. In: Bach, E. and Harms, R. (eds.): Universals in Linguistic Theory. Holt, Rinehart, and Winston, pp.1-90

Fodor, J.A. and Garrett, M.F. (1967): Some Syntactic Determinants of Sentential Complexity, Perception and Psychophysics **2**, 289-296 (1969)

Hayes, P. (1977): On Semantic Nets, Frames and Associations. Proc. IJCAI 77, pp.99-107

Hendrix, G.G. (1978): The Representation of Semantic Knowledge. In: Walker, D.E. (ed.): Understanding Spoken Language. Elsevier North-Holland, Amsterdam

Hendrix, G.G. (1979): Encoding Knowledge in Partitioned Networks. In: Findler, N. (ed.): Associative Networks. Academic Press, New York

Kaplan, R.M. (1973): A Multi-processing Approach to Natural Language. Proceedings of the National Computer Conference. AFIPS Press, Montvale, NJ, pp.435-440

Kay, M. (1973): The Mind System. In Rustin, R. (ed.): Natural Language Processing. Algorithmics Press, New York, pp.153-188

Levelt, W.J.M. (1970): Hierarchical Chunking in Sentence Processing. Perception and Psychophysics **8**, 99-102 (1970)

Levelt, W.J.M. (1974): Formal Grammars in Linguistics and Psycholinguistics, Vol.3: Psycholinguistic Applications. Mouton, The Hague

Maida, A.S. and Shapiro, S.C. (1982): Intensional Concepts in Propositional Semantic Networks. Cognitive Science **6**, 4 (1982)

McCord, M.C. (1982): Using Slots and Modifiers in Logic Grammars for Natural Language. Artificial Intelligence **18** (3), 327-367 (1982)

McKay, D.P. and Shapiro, S.C. (1980): MULTI - A LISP Based Multiprocessing System. Conference Record of the 1980 LISP Conference, Stanford University, pp.29-37

Pereira, F.C.N. and Warren, D.H.D. (1980): Definite Clause Grammars for Language Analysis - A Survey of the Formalism and a Comparison with Augmented Transition Networks. Artificial Intelligence **13**, 231-278 (1980)

Pollack, J., and Waltz, D. (1982): Natural Language Processing Using Spreading Activation and Lateral Inhibition. Proc. Conf. of Cognitive Science Society, pp.50-53

Quillian, R. (1968): Semantic Memory. In: Minsky, M. (ed.): Semantic Information Processing. MIT Press, Cambridge

Quillian, R. (1969): The Teachable Language Comprehender: A Simulation Program and the Theory of Language. CACM **12**, 459-476 (1969)

Quine, W.V. (1951): Mathematical Logic. Harper and Row

Quine, W.V. (1948): On What There Is. Review of Metaphysics **2**. Reprinted in: Linsky, L. (ed.): Semantics and the Philosophy of Language. University of Illinois Press, Chicago, 1952, pp.189-206

Robinson, J.A. (1965): A Machine-oriented Logic Based on the Resolution Principle. JACM **12**, 23-41 (1965)

Roussel, P. (1965): Prolog: Manuel de Reference et d'Utilisation. Groupe d'Intelligence Artificielle, Universite de Marseille-Luminy, September, 1975

Rumelhart, D. and Norman, D. (1973): Active Semantic Networks as a Model of Human Memory. Proc. IJCAI 73, pp.450-457

Schubert, L. (1976): Extending the Expressive Power of Semantic Networks. Artificial Intelligence **7** (2), 163-198 (1976)

Schubert, L.K., Goebel, R.G., and Cercone, N.J. (1979): The Structure and Organization of a Semantic Net for Comprehension and Inference. In: Findler, N. (ed.): Associative Networks. Academic Press, New York

Schubert, L.K. and Pelletier, F.J. (1982): From English to Logic: Context-Free Computation of 'Conventional' Logical Translation. AJCL **8** (1), 26-44 (1982)

Shapiro, S.C. (1971): A Net Structure for Semantic Information Storage, Deduction and Retrieval. Proc. IJCAI 71, pp.512-523

Shapiro, S.C. (1979a): The SNePS Semantic Network Processing System. In: Findler, N. (ed.): Associative Networks - The Representation and Use of Knowledge by Computers. Academic Press, New York, pp.179a-203

Shapiro, S.C. (1979b): Using Non-Standard Connectives and Quantifiers for Representing Deduction Rules in a Semantic Network. Invited paper presented at Current Aspects of AI Research, a seminar held at the Electrotechnical Laboratory, Tokyo

Shapiro, S.C. and the SNePS Implementation Group (1981): SNePS User's Manual. Department of Computer Science, SUNY at Buffalo, NY

Shapiro, S.C. and Neal, J.G. (1982): A Knowledge Engineering Approach to Natural Language Understanding. Proc. ACL, pp. 136-144

Shapiro, S.C., Martins, J., and McKay, D. (1982): Bi-Directional Inference. Proc. of the Cognitive Science Society, pp. 90-93

Simmons, R. (1973): Semantic Networks: Their Computation and Use for Understanding English Sentences. In: Schank, R. and Colby, K. (eds.): Computer Models of Thought and Language. Freeman

Tarski, A. (1944): The Semantic Conception of Truth. Philosophy and Phenomenological Research 4. Reprinted in: Linsky, L. (ed.): Semantics and the Philosophy of Language. University of Illinois Press, Chicago, 1952, pp. 13-47

Warren, D.H.D., and Pereira, F.C.N. (1982) An Efficient Easily Adaptable System for Interpreting Natural Language Queries. AJCL 8 (3-4), 110-119 (1982)

Woods, W.A. (1975): What's in a Link: Foundations for Semantic Networks. In: Bobrow, D.G. and Collins, A.M. (eds.): Representation and Understanding. Academic Press, New Xork, pp. 35-82