Logic for Semantic Networks

Robert J. Bechtel

TECHNICAL REPORT NO. 53

LOGIC FOR SEMANTIC NETWORKS

ROBERT J. BECHTEL

JULY, 1976

Submitted to the faculty of the Graduate School in partial fulfillment of the requirements for the degree Master of Science in the Department of Computer Science.

Abstract

Some method of making inferences is essential to a sophisticated natural language understanding system. Attempts to use classical logic in such systems have met with varying degrees of success, none of them total. We describe a non-standard logic developed for use in a semantic network based system, and the associated network representations of logical constructs. Three features extend the logic beyond standard classical logic. The first is the introduction of non-standard connectives. The connectives introduced, $\overline{\underline{X}}$ (AND-OR) and Θ (THRESH), are generalizations of the familiar symmetric binary connectives. A form of negation is also defined. The second feature is adaptation of a four-valued logic proposed by Belnap as a method of dealing with incomplete and contradictory information. At this point, a type of implication is introduced. Third, non-standard quantifiers are introduced. The new quantifiers, ALMOST-ALL, NONE, and ONE, ease the representation of many natural language constructions. Representations of truth value are developed for the network, and a system of postulates is proposed. Finally, a distinction is drawn between syntactic and semantic inference, and its ramifications are discussed.

I. Introduction

People seem to store, or remember, information they have received through natural language. However, they are also capable of both acting on and communicating information which they have never explicitly received. Such implicit knowlege also seems to be a part of human information stores. Sophisticated natural language understanding systems therefore require a method of obtaining information that is not explicitly stored in their data base. Generally, such methods are based on the application of rules which govern the extraction of implicit information from explicit information. Such rules may be termed <u>deduction rules</u>.

A desirable feature of such rules is an ability to express them independent of the representation in which they may be implemented (e.g. LISP code). One possible independent representation is as a logic system. If an independent, uniform representation is possible, comparisons and contrasts between various sets of deduction rules are easier to make and formulate. This approach is followed here.

Generally, any representation of deduction rules may be viewed on several levels. One level is that of the language chosen for implementation. Another level is that of data structures created and used. A third level might be a classification into "manipulating" and "manipulated" elements. While deduction rules and the information upon which they work are generally represented uniformly at the implementation level, such uniformity is not necessarily the case at the data structure and manipulation levels. In LISP, for example, information might be represented as atoms, with the deduction rules as functions (or lists), designed to deal exclusively with atoms. Most importantly, deduction rules and the information to which they apply can fall into the classifications of "controlling" and "controlled", respectively, at the third level of analysis, or they may span both categories, so that deduction rules, being both "controlling" and "controlled", may treat other rules as information. The latter possibility seems more desirable, as the deduction rules are information. This uniform representation approach will be followed here. This will also allow the representation to model one's understanding of the deduction rules, by representing them as information within the knowlege base.

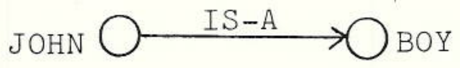
The internal representation of interest here is the semantic network of Shapiro, 1975a. While others have used semantic networks, no success has been reached in devising a useful, uniform, generalized inference mechanism for such a network. The reasons for this lack of success vary, and include deferring the problem of inference to concentrate on a more precise or detailed representation (Shank, 1973), (Fredericksen, 1975). Other attempts may be classified as lacking one of the necessary elements for a suitable mechanism. Quillian, 1968 and Simmons, 1973 lack uniform representation of rules and information, while Palme, 1975 and Anderson and Bower, 1973 are suspect on grounds of generality. Lindsay, 1963 lacks both elements. Some attempts, such as Shapiro, 1971a,b, Hendrix, 1975, and Shubert, 1975, have been successful in achieving theoretical uniformity and generality, but have faced other problems due to their classical approach, as will be discussed below.

The remainder of the paper, which is a more complete presentation of the material first presented in Bechtel and Shapiro, 1976, is in six major sections. Section II provides a brief background on the elements of the semantic network and their interaction. Section III, by far the longest section, discusses some of the problems inherent in classical logic, and develops possible solutions based on non-standard connectives and truth values. Network representations of all new constructions are also developed in this section. Both standard and non-standard quantifiers are introduced in section IV, and the possibility of using them to aid in controlling deductions is discussed. Section V deals with a logic based on the new connectives, and section VI draws an important distiction between types of inferences. Finally, conclusions are dicussed in section VII.

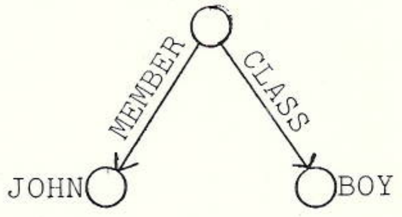
II. Network Representations

The network we are working with consists of nodes connected by edges called relations. The nodes may be constants or variables, with constants representing anything about which information may be given - in short, "conceptual entities". Variables act as "empty slots" that range, in general, over the set of nodes. Nodes are generally labelled, for convenience in referring to them.

Relations may not represent variable data as they are not intended to represent concepts, but rather structural information. The difference may be seen by considering the two following representations of "John is a boy."



(a)

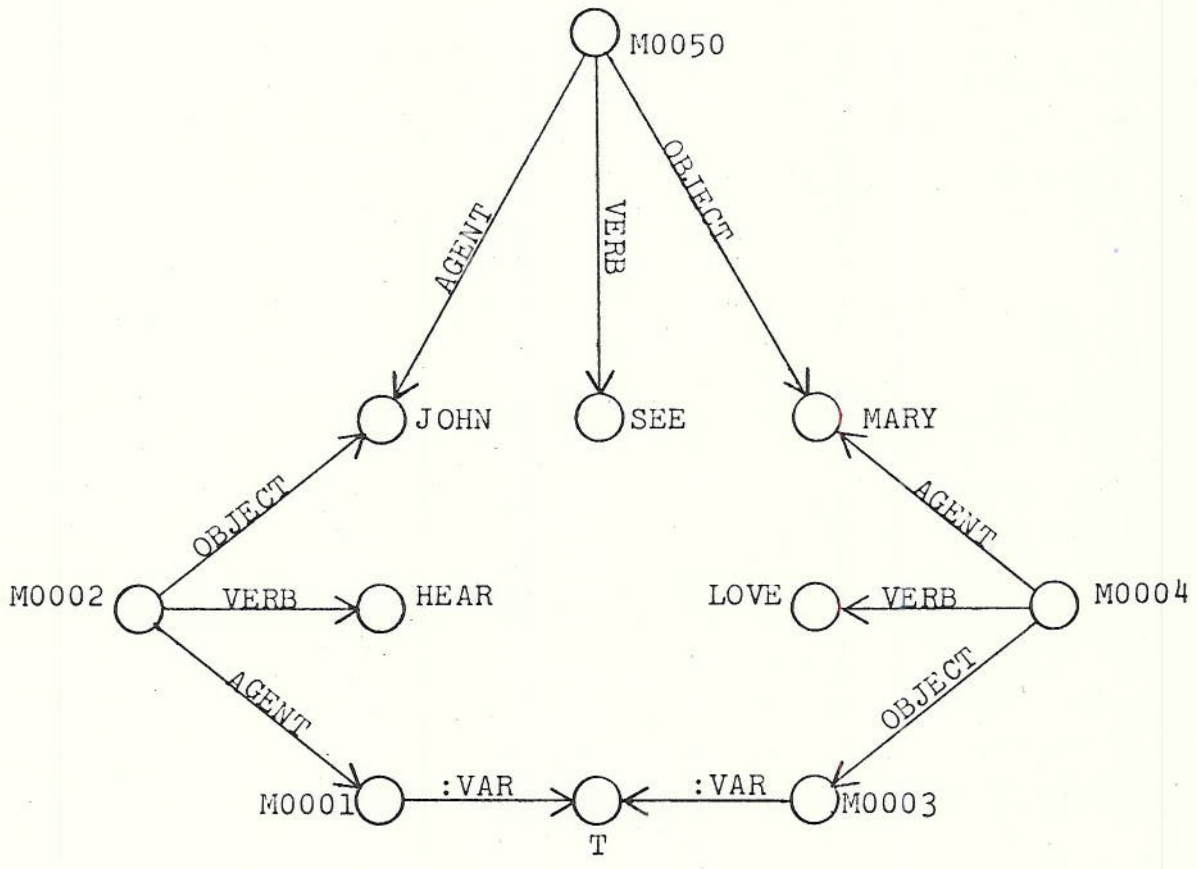


(b)

Notice that in (a) the only nodes, or concepts represented are those of JOHN and BOY. In (b), however, there is a third concept, which corresponds to "John is a member of the class of boys," or briefly, to "John is a boy." Since this is the concept we started with, (b) is the preferable representation.

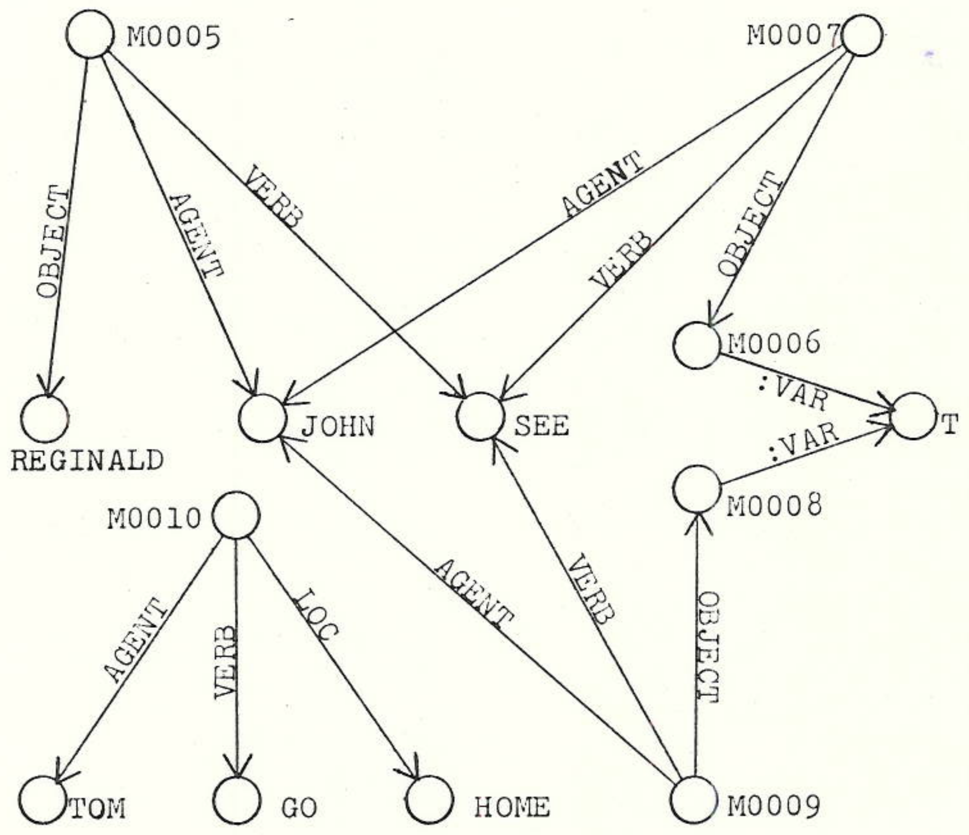
When relations are defined, their converses are also defined, and represented in the network whenever their corresponding relation is. Usually (again for convenience), converse relations are not shown in graphic representations of network structures. Some special relations, called _auxiliary relations_, have no converse. Conventionally, a relation is an arc between a node and another node which represents less information. Such relations are termed _descending_, while their converses are _ascending_. If this convention is followed, it may be said that one node _dominates_ a second if there is a path made up exclusively of descending relations between the first and second nodes.

Nodes may be classified by their dominance over other nodes. Nodes which dominate no others (but which may be dominated) are termed _atomic constants_ or _atomic variables_, depending on their nature. Variable nodes are indicated by the auxiliary relation :VAR, which relates variable nodes to the special node _T_. Nodes such as T, which have only auxiliary relations, are termed _auxiliary nodes_. Nodes which dominate other nodes are termed _molecular nodes_. Molecular nodes whose dominated atoms are all constants are termed _assertions_, while those which dominate variables are termed _patterns_. Examples of network structures are given in figure 1. As mentioned before, atomic variables range over any non-auxiliary node. Figure 2 gives examples of substitutions for an atomic variable.



Atomic Constants	- JOHN, SEE, LOVE, MARY, HEAR
Atomic Variables	- M0001, M0003
Auxiliary Nodes	- T
Patterns	- M0004, M0002
Assertions	- M0050
Relations	- AGENT, VERB, OBJECT
Auxiliary Relations	- :VAR

Figure 1. Network Structures.



All of the nodes above, except T, are substitutions for M0006

Figure 2. Valid Substitutions.

What is needed then, is a method of inferring new network structures from structures already present. A favorite method of accomplishing such derivation of new information from old has been the incorporation of some system of logic. The bulk of this paper is devoted to the development of an appropriate logic.

III. A. <u>Propositions and Connectives</u>

Assertions represent constant propositions, while patterns represent propositions with free variables. Many propositions are formed by compounding simpler propositions. Traditionally, such compounding has been accomplished by use of connectives from classical two-valued (TV) logic, & (AND) and v (OR). Such compounds can be represented in the network by introducing more relations, as shown in figure 3a. Another method of representing compounds, used by Fredericksen, 1975, is to treat standard connectives as relations, as shown in figure 3b.

One of the most apparent drawbacks of both of these approachs is due to the binary nature of the connectives. Each connective can take only two arguments at once. We may write A&B&C, but the expression is meaningful only through adopted conventions regarding association (to the left or to the right). To be strictly correct, the expression should be written either A&(B&C) or (A&B)&C. Fredericksen's method has no allowance for association

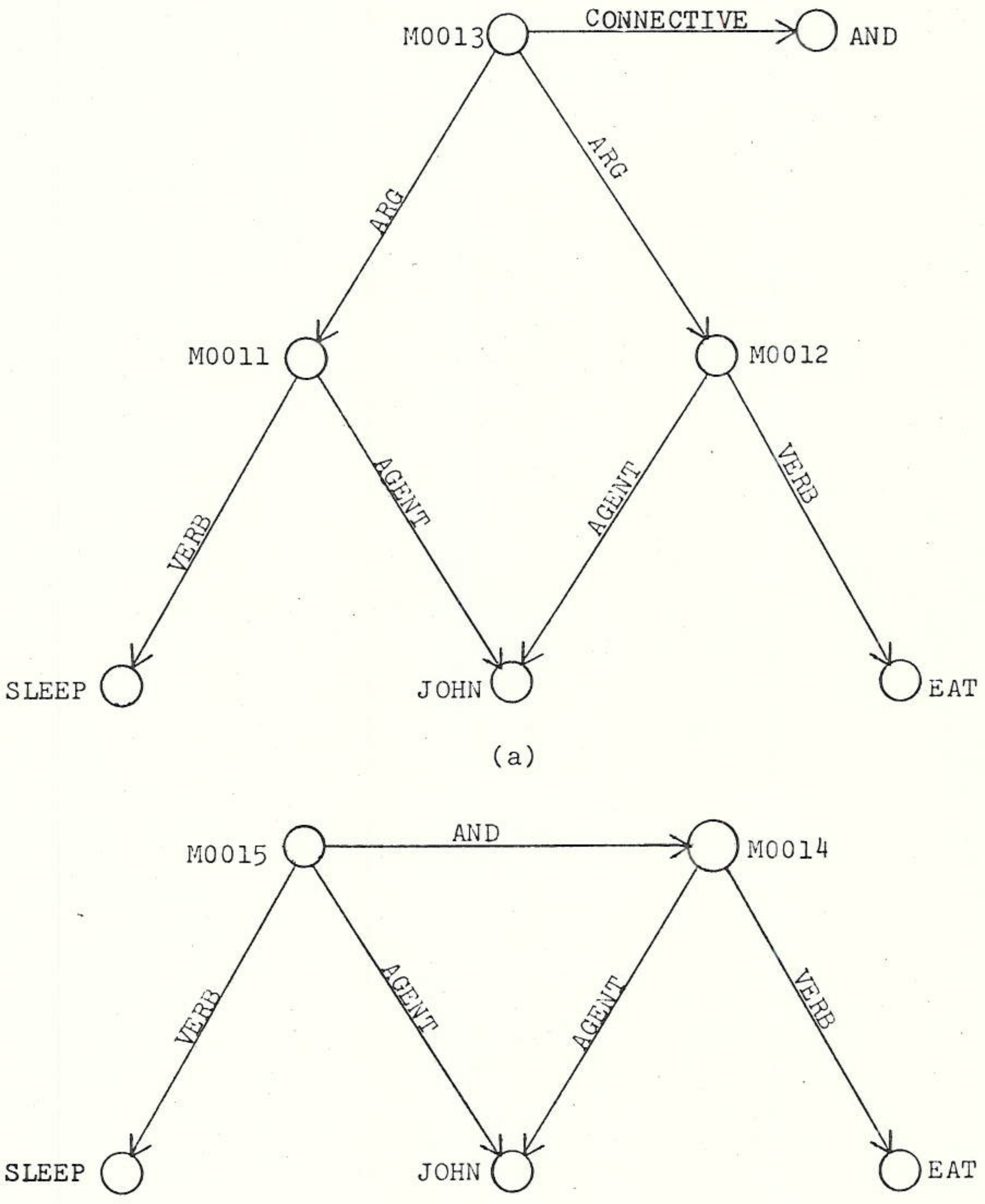


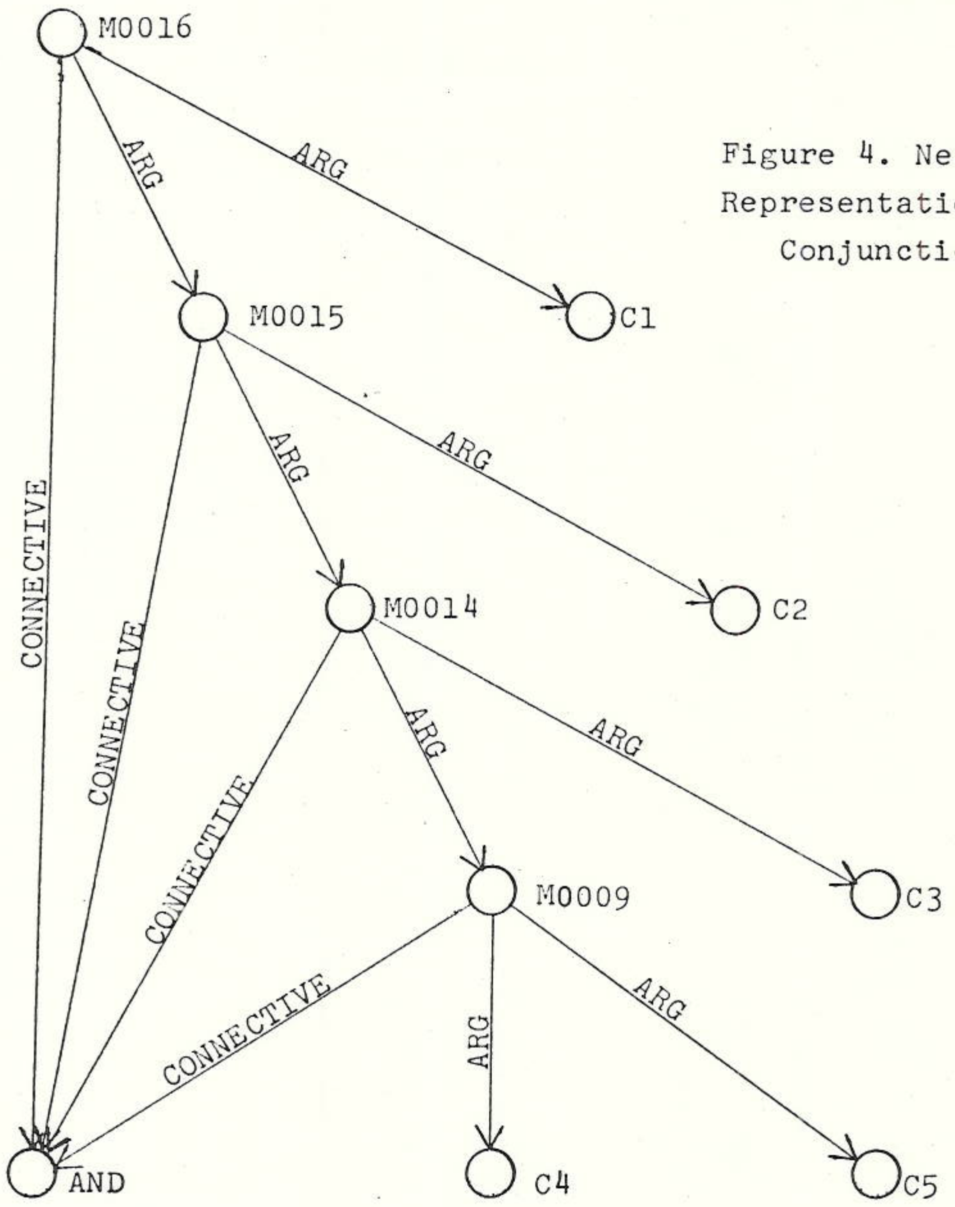
Figure 3. Possible Representations of Conjunction.

thus making (A&B)v(C&D) indistinguishable from A&((BvC)&D). It is not clear how such compounds would be represented, as there is no node for the concept of the conjunction or disjunction. The other method provides a node corresponding to the conjunction, but also causes trouble by expanding network structures enormously, as shown by the network structure required for the conjunction of five arguments in figure 4a. This problem may be easily avoided by noting that AND and OR readily lend themselves to expression as n-ary connectives (n>2). Since both of these connectives are associative, commutative, and idempotent, the arguments to n-ary ANDs and ORs may be expressed by sets, which by definition are order independent and contain no duplications. The conjunction of five arguments may be expressed as AND(C1,C2,C3,C4,C5) and in the network as shown in figure 4b.

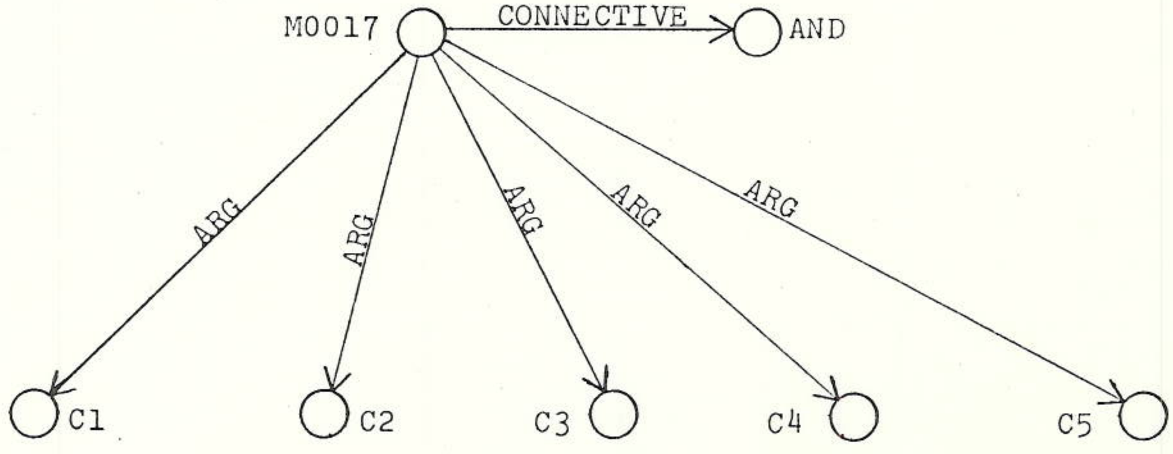
Another problem also concerns the network size required for a correct representation, and may best be demonstrated by use of an example. Consider the statement "of the five students, only two or three did any work." Representing "two or three of five" using TV connectives is not simple, as shown below.

((C1&C2)v(C1&C3)v(C1&C4)v(C1&C5)v(C2&C3)v(C2&C4)v
(C2&C5)v(C3&C4)v(C3&C5)v(C4&C5))&~((C1&C2&C3&C4)v
(C1&C3&C4&C5)v(C1&C2&C3&C5)v
(C1&C2&C4&C5)v(C2&C3&C4&C5))

Figure 4. Network Representation of Conjunction.



(a) binary



(b) n-ary

Not only is this long and involved in standard TV notation, but would require a large amount of network to reproduce. In addition, the standard TV formalism requires the use of negation (NOT or ~), a problem not yet discussed.

A third problem also arises from the binary nature of the connectives. This problem appears with the use of "derived" connectives such as exclusive or (XOR or ⊕) and equivalence (EQUIV or ≡). For the simple binary case, they are harmless enough, but an attempt to expand them to the n-ary case can result in disaster, since even though they are associative and commutative, they are not idempotent, i.e. A⊕A=F and A≡A=T. In addition, they are non-intuitive for more than two arguments. Consider XOR. Using it in the two argument case, one develops an intuition which runs something like "it's true if and only if exactly one of its arguments is." This appears to work for the three valued case until all the arguments are true, when A⊕B⊕C is true too. Intuition strikes out. A similar situation holds for A≡B≡C as shown in the truth table below.

A	B	C	A⊕B⊕C	A≡B≡C
F	F	F	F	F
F	F	T	T	T
F	T	F	T	T
F	T	T	F	F
T	F	F	T	T
T	F	T	F	F
T	T	F	F	F
T	T	T	T	T

we expect $A \equiv B \equiv C$ to be true when all its arguments are the same, but it just isn't so. The classical sense of XOR and EQUIV over n arguments may be summarized as follows:

> XOR$(A_1, \ldots, A_n)$ is TRUE if and only if an odd number of its arguments are TRUE.
>
> EQUIV$(A_1, \ldots, A_n)$ is TRUE if and only if an even number of its arguments are FALSE.

Representing the intuitive understanding of $A \oplus B \oplus C$ and $A \equiv B \equiv C$ by properly associated binary connectives is no easy task. The seemingly innocent statement heading many theorems, "the following are all equivalent" looks like this when seven things follow:

$$(A \equiv B) \& (B \equiv C) \& (C \equiv D) \& (D \equiv E) \& (E \equiv F) \& (F \equiv G).$$

This is a bigger problem, as we would like to represent as simply as possible intuitive relationships and rules used by people in making sense of natural language.

To resolve these problems, we introduce two non-standard connectives: $\mathbb{X}$ (AND-OR) and Θ (THRESH). These will replace AND, OR, EQUIV, and, to a certain extent, NOT.

${}_n\mathbb{X}_i^j$ is the connective that will do most of the work, and is equivalent to the function $\tau_i \bar{\tau}_{j+1}$, independently developed by George Epstein for a different purpose (Epstein, 1958). ${}_n\mathbb{X}_i^j$ takes a set of n arguments (propositions) and is true if at least i and no more than j of its arguments are true. It is, by definition, an n-ary connective, and can readily replace OR and AND as shown:

$${}_n\mathbb{X}^n_1(A_1,\ldots,A_n)=A_1\vee\ldots\vee A_n$$

$${}_n\mathbb{X}^n_n(A_1,\ldots,A_n)=A_1\&\ldots\&A_n$$

In addition, AND-OR enables an easy formulation of the "students who work" example. In this formalism, the statement would be expressed

$${}_5\mathbb{X}^3_2(C1,C2,C3,C4,C5).$$

The intuitive understanding of an n-ary XOR is also very simple:

$${}_n\mathbb{X}^1_1(A_1,\ldots,A_n).$$

Notice that this expression is true if at least one and no more than one of its arguments are true. This overcomes the earlier objection to the classical XOR. Possibly the most unexpected benefit of AND-OR is its ability to replace negation in most, if not all, cases. With the minimum and maximum parameters both 0, AND-OR functions as NOR, but without any associative difficulties that might occur, as with XOR. Such a representation would appear

$${}_2\mathbb{X}^0_0(A,B)=A\downarrow B$$

Also, since it accepts any number of arguments, direct negation is possible.

Of course, we could replace EQUIV with appropriately nested AND-ORs, e.g.

$$A \equiv B = {}_2X_1^1({}_2X_2^2(A,B),{}_2X_0^0(A,B)) \text{ or}$$

$$A \equiv B = {}_1X_0^0({}_2X_1^1(A,B)),$$

but to avoid reintroducing the problems of network proliferation we wish to eliminate, we instead introduce THRESH. This connective grew out of discussions by the SNePS Users' Group during 1975 concerning network representations of natural language constructions. ${}_n\Theta_i$ takes n arguments and is true if less than i or all n are true. Simple equivalence may be represented over n arguments by setting i=1, thus guaranteeing that all the arguments are either true or false. ${}_n\Theta_1(A_1,\ldots,A_n)$ is equivalent to the MUTIMP$(A_1,\ldots,A_n)$ (mutual implication) of Shapiro, 1971a,b. The theorem with seven equivalent statements becomes

$${}_7\Theta_1(A,B,C,D,E,F,G).$$

Notice that EQUIV has been generalized by THRESH so that i becomes a sort of threshold, which, when reached, requires the remaining arguments to be true.

In addition, AND-OR can be used to represent all symmetric binary connectives. Along with AND, OR, XOR, EQUIV, and NOR, these include NAND (|) and the constant TRUE and FALSE conditions. These representations are shown below.

$${}_{2}\mathbb{X}_{0}^{1}(A,B)=A|B$$

$${}_{2}\mathbb{X}_{0}^{2}(A,B)=\mathrm{TRUE}$$

$${}_{2}\mathbb{X}_{3}^{3}(A,B)=\mathrm{FALSE}$$

III. B. The Failure of TV

Up to this point, the usefulness of classical logic has been assumed, with the connectives merely replaced with ones better suited to the purposes at hand. Now we will discuss some of the problems inherent in TV logic. The first has faced logicians since at least as far back as Aristotle, and has created problems for others working in natural language understanding. TV assumes that every proposition has a known truth value, that it is either true or false. Aristotle tried to assign a truth value to the proposition "There will be a sea fight at noon tomorrow" (De Interpretatione, Ch. IX) and finally decided that it was impossible. Even without the temporal element to add confusion, any natural language understanding system is faced with the possibility of a reference to something outside its carefully prepared domain of discourse. What truth value should be assigned to a proposition never before encountered? Of course, answering that question is part of the function of deduction rules, but even they may not be useful. The same problem is more apparent if the system is asked to reply "true" or "false" to some proposition, rather than only having to store some truth value for each proposition.

The obvious answer is that the proper reply when faced with insufficient information is "I don't know." This solution has been recognized in almost all question-answering systems. "I don't know" is not, however, a classical truth value. It is the indication of a total lack of a truth value. In some sense, it is undetermined. Propositions which the system does not know about cannot be marked "true" or "false", but must remain unmarked. For this reason, we introduce a third, non-standard truth value, called "none" which will indicate this previously unmarked state.

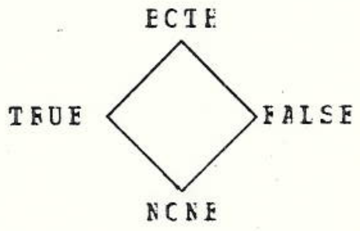
Another problem is probably just as old. This is the problem of contradiction. Most systems of logic exclude contradictions by careful selection of postulates, but a number of features of natural language make this a difficult method for use in that application. Again, TV logic assumes every proposition has one, and only one, of the two possible truth values, when the "propositions" a natural language system has to deal with may have more than one value. This is particularly noticeable when the system accepts without question any new information it is given as fact. The system may then be told that a particular proposition is both true and false. Contradictions which arise in this way we will call _explicit contradictions_.

Another source of contradiction lies in the fact that there is no guarantee that deduction rules which are useful in understanding natural language are consistent. Furthermore, observations of human information processing and belief systems would indicate just the opposite. Therefore, contradictions may also be introduced by postulates or deduction rules, a manner which is avoided by most formal logical systems. Contradictions arising from deduction rules may be termed _implicit contradictions_.

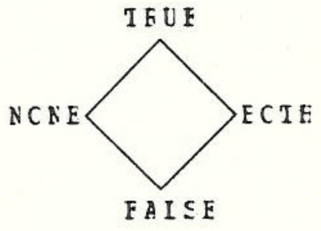
Finally, contradictions may arise out of the ambiguous nature of natural language itself. This ambiguity makes explicit contradiction more likely, if unnoticed, and may also create implicit contradiction, if deduction rules are taken from natural language.

We are not yet prepared to launch a full scale attack on contradiction, but rather we wish to show that it can exist in a natural language understanding system without straining the bounds of the system. At present, we only wish to argue that, since it can exist, we should have some way of representing it. Using the tag or label idea implicitly advanced in the discussion of unknown truth values, it would be possible to have two copies of every proposition, one labelled "true" and the other labelled "false". This is the method used in TV logic, with the prototypical statement of contradiction being A&~A. There are two objections to this scheme. One is our old friend, the problem of network size. Duplicating every contradictory proposition might be all right if there are just a few, but we have no reason to believe that there will be just a few such propositions, or that they will be small and simple. The second is a sort of self-protection against errors. We would like to make any data base check as simple as possible. If the contradictions are "stored" on two separate propositions, finding a single labelled proposition only means that the search is half complete. In the case that a proposition is true, the entire data base must be searched to be sure that an identical proposition is not false, signaling a contradiction. Having established the legitimacy of an empty label for the case of an unknown truth value, we now propose the label "both" or "true, false" to signal a contradiction. Now there are four truth value which a proposition may take, and we are faced with the problem of devising ways of dealing with these truth values outside classical TV logic.

Is there some relation between these truth values, as there is in TV? For an answer, we turn to Belnap, 1975a,b and the idea of an <u>approximation lattice</u>. An approximation lattice is just a complete lattice which also satisfies the intuitive condition that its partial ordering relation may be read "approximates" (e.g. $x \leq y$ is read "x approximates y"). Given our four truth values, an intuitive understanding of "approximates" might be "gives no more information than". With this understanding, the four values form the following lattice (called A4 by Belnap):



This approximation lattice may then be manipulated with a desire to preserve classical TV characteristics to produce L4



in which & is meet and v is join, and negation is defined as follows:

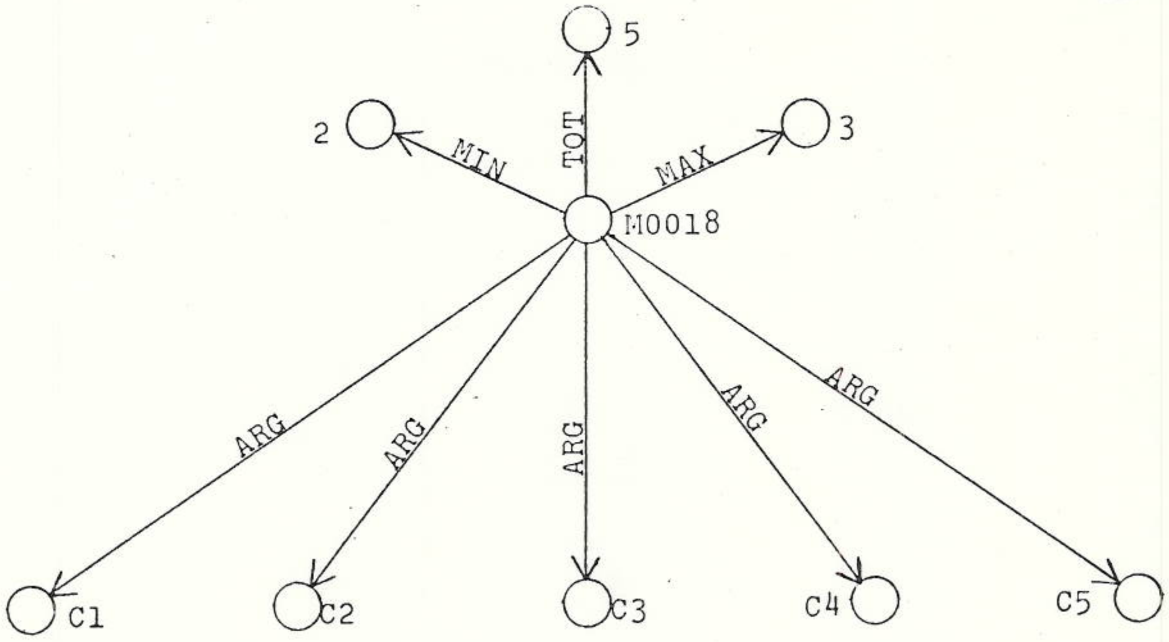
A	NONE	FALSE	TRUE	BOTH
~A	NONE	TRUE	FALSE	BOTH.

For convenience in notation, we shall refer to these four truth values by their first letters, N, F, T, B.

So now we have four truly glorious truth values, but are faced with the old connectives. Fortunately, the replacement of AND, OR, and NOT will be quicker this time. ${}_nW_i^j(A_1,\dots,A_n)$ is marked at least T just in case at least i and no more than j of its arguments are marked with at least T (in <u>A4</u>). ${}_{n}W_{i}^{j}(A_1,\ldots,A_n)$ is marked at least F just in case less than n-j or more than n-i of its arguments are marked with at least F (in <u>A4</u>). ${}_{n}\Theta_{i}(A_1,\ldots,A_n)$ is marked at least T just in case less than i or all n of its arguments are marked at least T, and is marked at least F if more than 0 but less than or equal to n-i of its arguments are marked at least F.

III. C. <u>Network Representation of AND-OR and THRESH</u>

Finally, we must have a way of representing these connectives in the network. Since the number of arguments may vary and order is unimportant, a single type of descending relation could serve to join arguments to a node representing a connective. We define such a relation, called ARG. The parameters on a connective provide the information necessary to its evaluation, and so must be included in the network. On AND-OR, the parameters indicate the total number of arguments and both the minimum and maximum number that may be true to satisfy the connectives. We define the three auxiliary relations TOT, MIN, and MAX to represent these. The THRESH connective may also use TOT, and additionally requires an indication of the threshold number of arguments, for which we will use THRESH as the name of the auxiliary relation. Examples of network structures may be seen in figure 5.



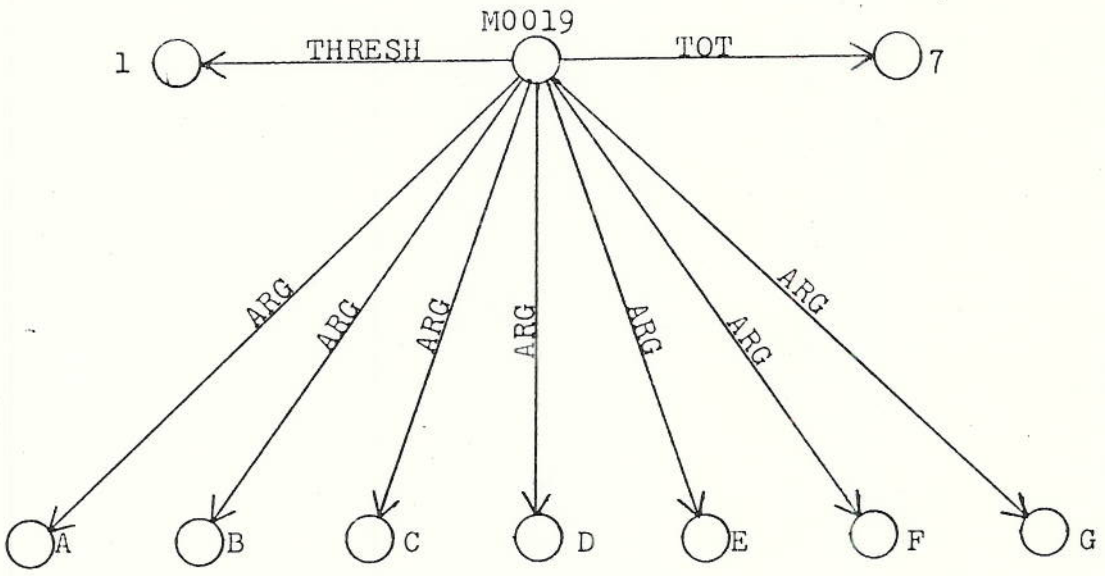


Figure 5. Network Representation of AND-OR and THRESH.

III. D. Inference

Having established a useful set of truth values and more convenient connectives we still lack one of the most important elements of a logic, the ability to use known information to infer new information. In TV, this ability is vested in material implication (⊃ or IMP). However, the suitability of material implication for such uses has been attacked by logicians (Anderson and Belnap, 1975) and psychologists (Kintsch, 1974). One of their objections is particularly relevant to a natural language understanding system. The objection, simply stated, is to the fact that, in TV, a false proposition implies anything, and a true proposition is implied by anything. Do we wish to say that any implication whose antecedent is false holds? Again, consider the problem of contradiction. A contradiction, at least classically, is always false, and so implies anything. Anderson and Belnap (1975) have made perhaps the broadest and most complete criticism of material implication, in which they assert "Material implication is no more a 'kind' of implication than a blunderbuss is a 'kind' of buss." Along with the criticism, they propose alternative relations, which are intended to more closely approximate the intuitive understanding of implication. One of these new relations surfaces again in (Belnap, 1975a,b) and is simply defined on L4 as "going uphill". If moving from the antecedent to the consequent in the lattice involves taking a downward path, the implication (now an <u>entailment</u>, in Anderson and Belnap's terms) is false. Rather than being always (at least) T, an entailment whose antecedent is contradictory is now T only if its consequent contains T. Truth tables for IMP (defined as ~AvB) and entailment (-> or ENTAILS) show other interesting differences.

⊃	N	F	T	B
N	N	N	T	T
F	T	T	T	T
T	N	F	T	B
B	T	B	T	B

->	N	F	T	B
N	T	F	T	F
F	T	T	T	T
T	F	F	T	F
B	F	F	T	T

Notice in particular that every entailment has a single, well-defined truth value (T or F) and is never undetermined or contradictory. Consider the expression A->A. If we have a well behaved logic, we would certainly hope that this implication would always be true, regardless of the truth value assigned to A. As can be seen, this is not the case for IMP. As Belnap has noted, ENTAILS also preserves truth and falsity, in the sense that it never leads from truth to the absence of truth or from the absence of falsity to its presence. IMP is guilty on both counts of non-preservation, as can be seen from the truth table.

Having selected a suitable implication connective, we must now find a way to represent it in the network. Entailment differs from the simpler connectives AND and OR in that, while it is also binary, it is <u>not</u> symmetric. B->A does not necessarily follow from A->B. (Indeed, one would be greatly disturbed if it did!) Generalizing to any number of arguments, as was done in AND-OR and THRESH, $(A_1,\ldots,A_n)\text{->}(B_1,\ldots,B_m)$ may be read "the disjunction of the As entails the conjunction of the Bs", and abbreviates the n times m rules of the form $A_i\text{->}B_j$, $1<i<n$, $1<j<m$. As in AND-OR and THRESH, a use of ENTAILS will be represented as a node, with descending relations to the arguments. Since the sets of arguments are distinguished, they must be uniquely identified by use of different descending relations. We choose to call these relations ANT and CQ, intended to be suggestive of antecedent and consequent respectively. Examples are shown in figure 6.

III. E. <u>Network Representation of Truth Values</u>

All assertions and patterns must have their truth value indicated in some way, as the classical convention extends to only two truth values. Classically, to state "A" is equivalent to stating "A is true." By the same convention, "~A" is equivalent to "A is false." The presence or absence of the negation sign is sufficient to determine the truth value of the expression. However, a binary choice is not sufficient to distinguish among four truth values, such as we have in L4. So, the last construction needed for a useful propositional logic system is a method of indicating the truth value of the

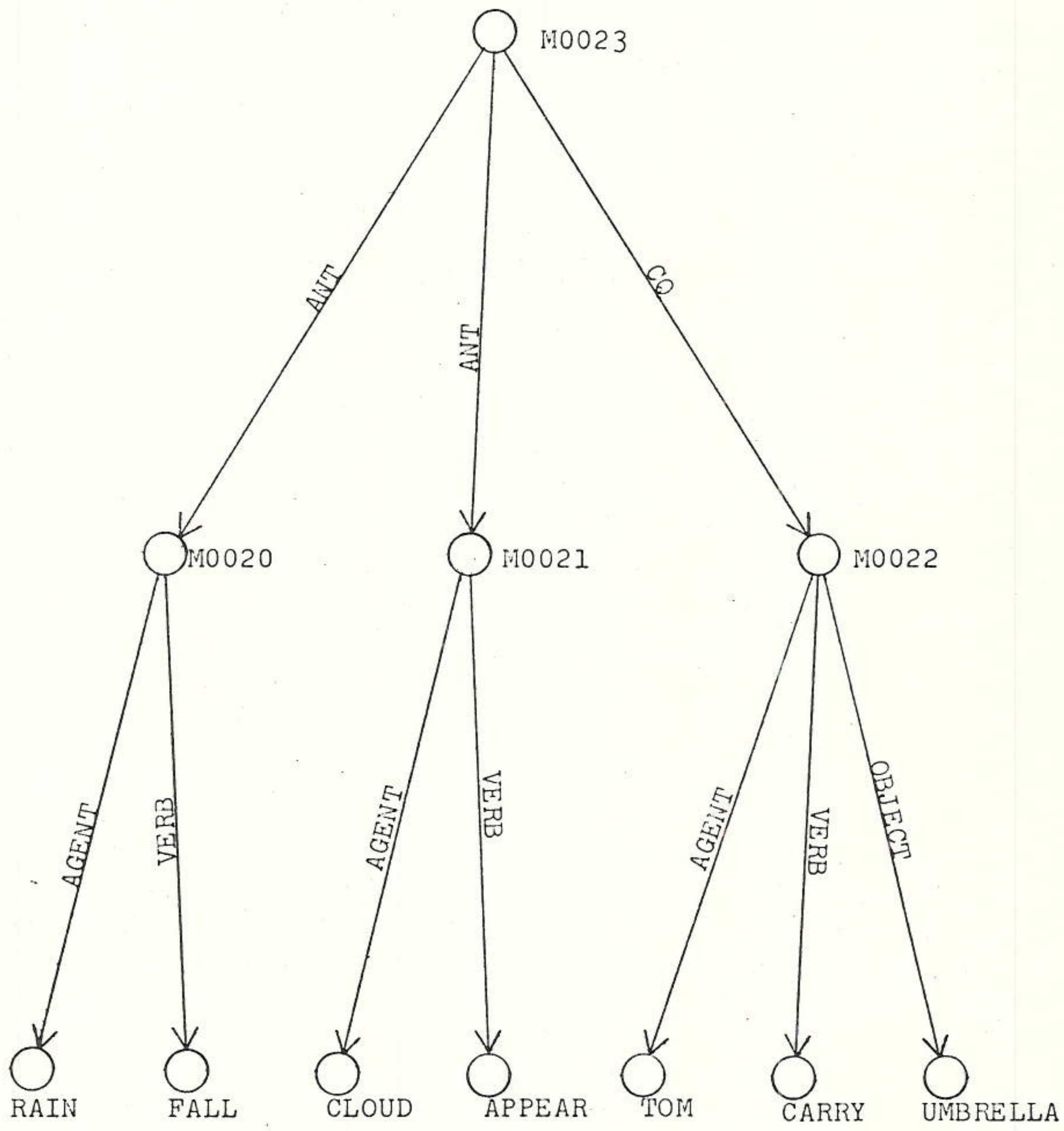


Figure 6. Network Representation of ENTAILS (ANT-CQ) with multiple ANTecedents.

assertions and patterns. This is handled by introducting an auxiliary relation, called TVAL, which relates an assertion or pattern node to another node which represents the truth value of the expression represented by the first node. This method grows out of the earlier idea of labelling, and examples are shown in figure 7. One unusual feature needs to be pointed out; expressions that are not found in the network either explicitly or implicitly (deduced) are assumed to have a truth value of NONE, that is, they are neither true nor false. This accords with the motivation for the establishment of the NONE truth value.

IV. A. Quantifiers

Now that connectives and truth values are settled, it would be desirable to include more than constants in the deduction rules. In classical logic, variables are included through the use of quantifiers and predicates; a similar path will be followed here. Assertions and patterns, which have been funny looking propositions (being made up of many parts, rather than being single entities) may now be viewed as predicates. The variables in the patterns range (generally), as noted before, over the universe of nodes in the network, being validly substituted for by other nodes.

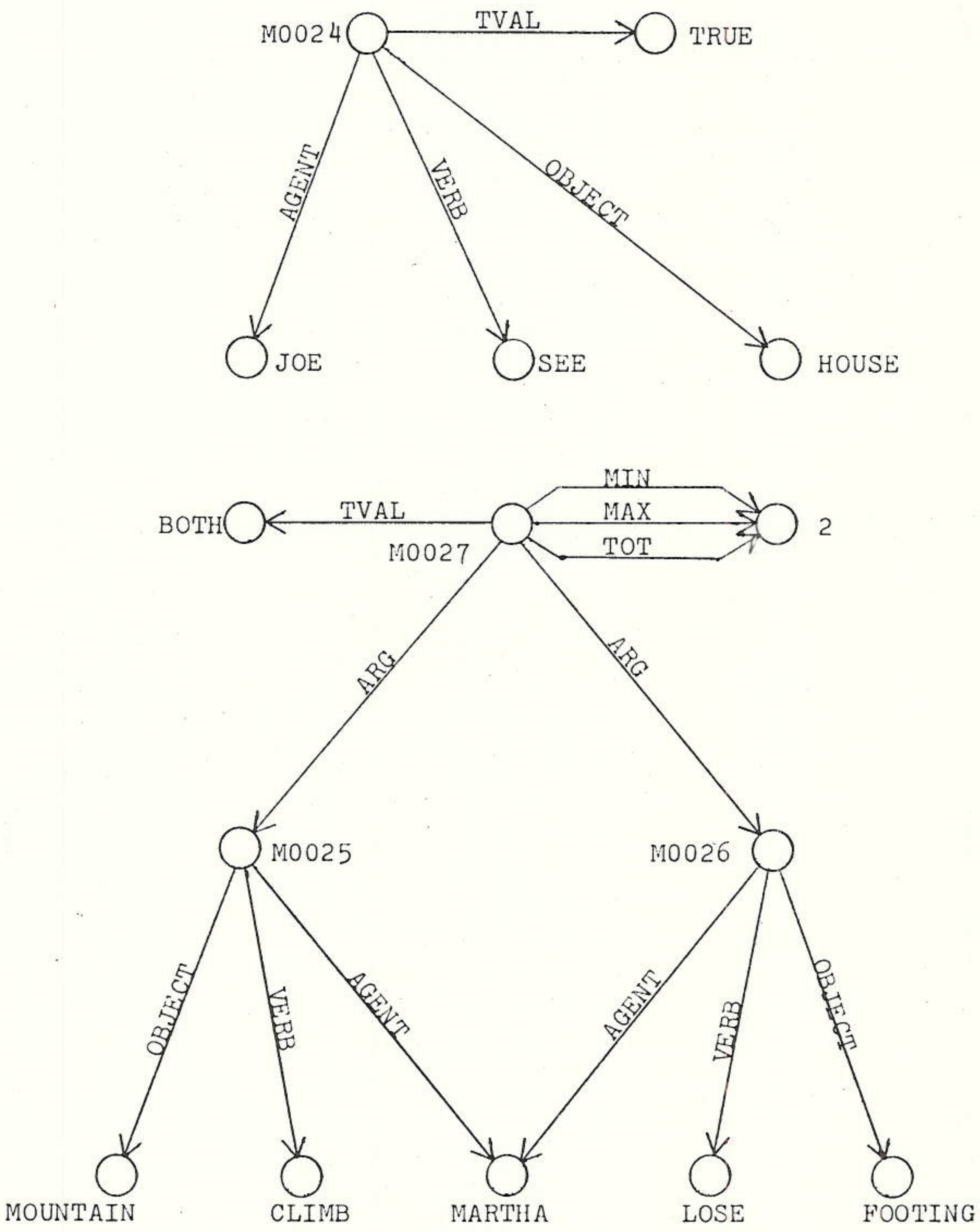


Figure 7. Network Representation of TVAL.

Classically, quantifiers serve to specify boundaries (or lack of boundaries) on the class of substitutions or instantiations which may be made for the variables they bind, such that those substitutions will retain the validity of the expression in the scope of the quantifier. The classical quantifiers ∀ (ALL) and ∃ (EXISTS) specify boundaries as follows: ALL has no upper bound, and its lower bound is equal to the number of possible substitutions; EXISTS has the same upper boundary, but has a lower bound of one valid substitution. Put another way, ALL insures that any substitution for its variables is valid, while EXISTS insures that there is at least one valid substitution.

The power of these quantifiers, particularly that of ALL, is difficult to use effectively. It is hard, for example, to think of a predicate which is true for all elements of a universe as possibly diverse as that of a semantic network. For this reason, exclusive use of an implicit ALL quantifier on expressions is not acceptable, forcing the inclusion of at least both of the classical quantifiers.

We also wish to be able to add expressions to the network which, while not tautologies, may be useful in understanding natural language. This should be possible without interfering with expressions already in the network. It would be possible to restrict the new closely related to intuitionistic logic, and is interpreted as "not the case that Q(x) is marked FALSE." The idea is that the expression holds for any substitution which does not directly contradict the consequent. In the absence of such a contradiction, ALA functions in exactly the same way as ALL.

A number of natural language constructions may be represented by use of these quantifiers, but many of these representations (particularly those involving ALL and EXISTS) are very complex, resulting in disproportionate increases in network size for the intuitive complexity of the concepts expressed. In addition, studies indicate that classical quantifiers (ALL and EXISTS) are highly artificial and non-intuitive to untrained users (Martin, 1975), (Thomas, 1975). For these reasons, we introduce two more non-standard quantifiers in an attempt to more closely model intuitive concepts.

NONE is formed by collapsing a connective into a quantifier. It aids in the simple representation of concepts such as "no one in their right mind would try that," and may be expressed classically as shown:

NONE(x)(P(x)->Q(x)) = ALL(x)(P(x)->~Q(x))

(where ~Q(x) is interpreted "Q(x) is marked FALSE").

The last new quantifier is ONE, which is the same as the ∃! used by classical logicians. For this reason, it is non-standard only by virtue of being a direct implementation rather than an abbreviation. ONE sets boundaries on valid substitutions in the same way as ALL and EXISTS, except that its upper and lower boundaries are both one. ONE asserts the existence of exactly one validating substitution for the variables it binds. An equivalent expression may be constructed using classical quantifiers, as shown.

ONE(x)(P(x)) = EXISTS(x)(P(x)&ALL(y)(P(y)->(x=y)))

The new quantifier is superior to the classical construction using EXISTS and ALL because it more closely models intuition, it reduces network size, and it avoids (or at least postpones) the problem of representing equality.

The application of quantifiers is straightforward. Deduction rules are "used" by finding assertions in the network which fully instantiate a rule's antecedent side, and using the substitutions derived from such assertions, constructing instantiations of the consequent side. EXISTS and ONE may be dealt with by generating a constant to fill the positions held by variables, thus changing any deduction rule quantified by one of these quantifiers into a non-quantified assertion dominating only constants (possibly including fully bound rules). In the case of EXISTS, this replacement is performed without exception. To preserve the uniqueness indicated by ONE, however, it is necessary to first search the network to find if a constant already exists which would satisfy the rule. If such a constant exists, it is used as the instantiation. If not, a new constant is generated, just as with EXISTS.

ALL and NONE do not limit substitutions. Therefore, under the proviso that substitutions within a rule must be uniform, finding a network structure which will fully instantiate the antecedent of a rule is sufficient to permit the assertion of the consequent of the rule (negation of the consequent, in the case of NONE) under the same substitutions. ALA, which has a further restriction, requires in addition that the negation of the consequent under the substitutions of the antecedent _fail_ to be found (or quickly inferred) in the network before the consequent may be asserted.

IV. E. _Quantifiers and Rule Ordering_

The concept of likelihood of valid substitutions, as reflected in the preceding interpretations of quantifiers, may be used to reach a view of the _applicability_ of rules. It would seem that the more likely the possibility (or impossibility) of valid substitutions, the more generally applicable the rule is. We are concerned now only with quantifiers over an entire deduction rule. Quantification over portions of a deduction rule is covered elsewhere (Shapiro, _et al_, 1976).

EXISTS and ONE need not even be considered in such an explication of applicability since, as mentioned before, they may be replaced by constants which instantiate the variables they quantify, as in a Skolem function. This leaves only ALL, ALA, and NONE. ALA is useful only if the negation of its consequent is not found explicitly or quickly deduced in the network, i.e., there may be cases in which it is not applicable. ALL and NONE show no such restrictions. With the idea of applicability, quantifiers of deduction rules can be used to order proof trees. If, for example, two rules are available to reach a particular conclusion, it is possible that one will be more likely to be applicable than the other. In light of the discussion above, an ordering based on applicability of quantifiers would place ALL and NONE first, followed by ALA.

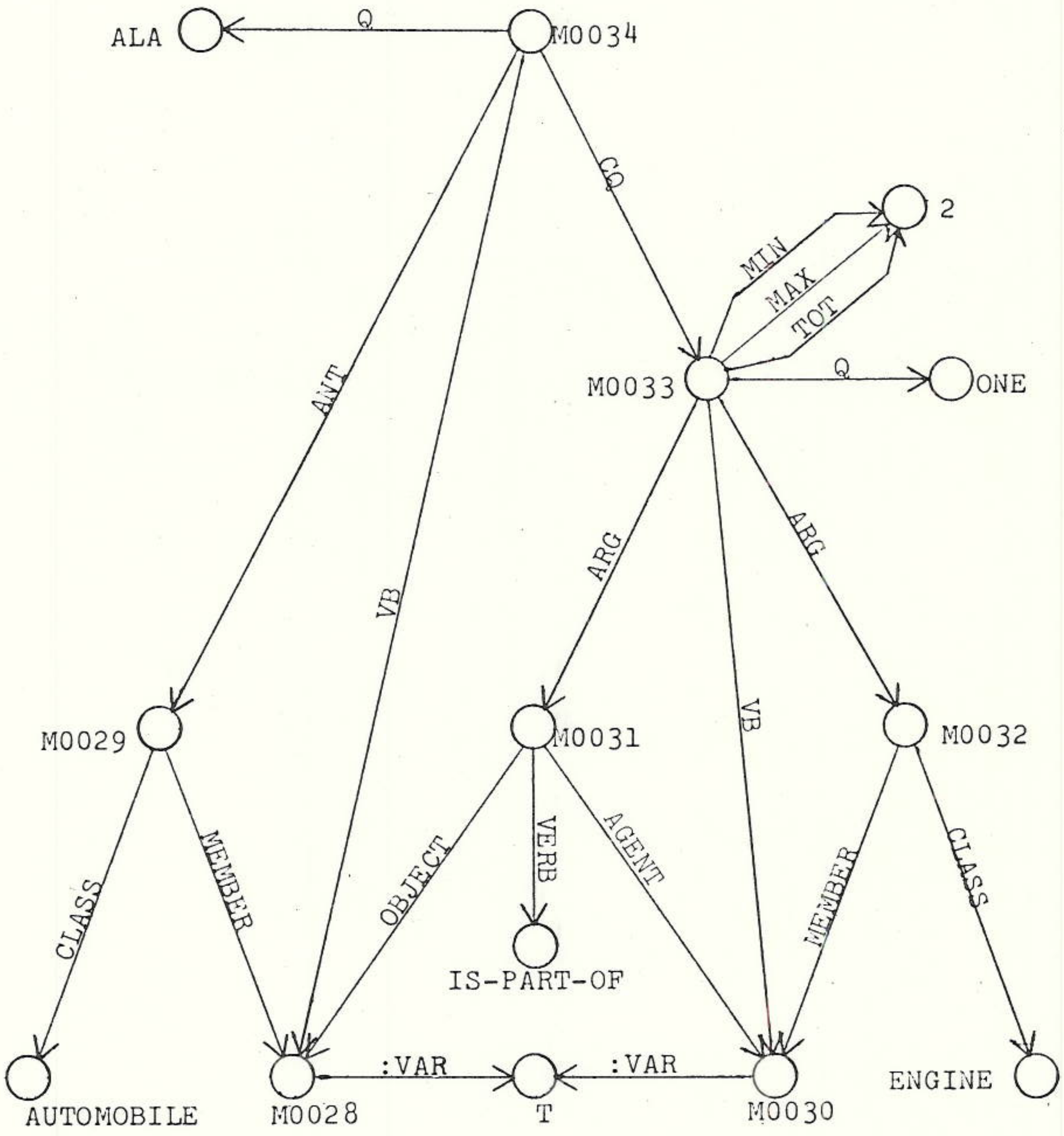
Another way of looking at rules is in terms of anticipated effort. This would be a measure of the effort expected to use the rule. Again ALL and NONE are useful, with effort expended only in setting up bindings by finding a network structure to instantiate their antecedents, while ALA requires an additional check on possible substitutions by searching for the negation of the consequent before making the bindings. Ordering on anticipated effort would then result in the same order as applicability. Using these methods of ordering can (hopefully) reduce the resources expended on a given proof or derivation.

IV. C. Network Representation of Quantifiers

Now it is necessary to define a network representation for these quantifiers. We define an auxiliary relation, Q, which relates a pattern which represents the expression to be quantified to a node representing the quantifier. The node linked to the quantifier should be a pattern rather than an assertion, as there is no sense in quantifying over constants. To indicate which variables are bound by the quantifier, a relation named VB is inserted between the pattern node and all those variables bound by the quantifier related to that node. This is made possible by the fact that, for any quantifier Q, $QxQy[P(x,y)]=QyQx[P(x,y)]$, thus making possible the abbreviation $Q(x,y)[P(x,y)]$. If a pattern node is related to a quantifier and all the variables it dominates are bound by VB relations from that pattern node or other pattern nodes it dominates, the pattern node represents a deduction rule with no free variables, and is thus an assertion. An example of a network representation of a quantified deduction rule is shown in figure 8.

V. A Logic

The most familiar representation of logical systems is axiomatic, with a number of postulates given which allow derivation of all expressions valid in the logic, and no others. These postulates consist of axioms, or



"Most cars have an engine."

Figure 8. Fully Quantified Deduction Rule.

constructions which are defined as valid, and which may therefore be used at any point, and rules, which govern the insertion and deletion of connectives and expressions. Most axiomatic systems consist of axiom schemata, which provide a general format for the axioms, which are formed by uniform substitution for the variables used in the schemata. As axiom schemata are constructions just as any other, with the exception that they are guaranteed valid, they may be represented directly in the network as deduction rules.

A common rule is modus ponens, or detachment. Application of this rule results in assertion of the consequent of an entailment, when the antecedent is true. This process cannot be represented directly in the network, and so must be handled by a procedure that creates (builds) a new assertion which is identical to the consequent of the entailment, provided the appropriate conditions for application of the rule hold. Similar procedures are required for other rules, such as adjunction. The procedures which provide the power of rules are a part of an interpreter whose purpose is to use explicit information to derive implicit information. The interpreter (described in Shapiro, et al, 1976) is the facility which utilizes quantifiers as indicators of effort, and which needs an indication of truth value. It is essentially a theorem prover, with the rules (which may be changed) as a subpart. Since the axioms and rules are not fixed, but may be changed by providing different network constructions and procedures, it is possible to experiment with different logic systems.

The simple logic system presently being implemented was developed along with the network representation now in use. For the most part, it simply grew, with little or no thought given to a formal, representation independent expression of postulates. The rules, imbedded in the interpreter, are variations on the theme of detachment, designed so that one may, given A->B, infer B directly from information sufficient to show A. The first rule premits the use of the ENTAILS connective this way.

R1. From A->B and A to infer B.

In addition, four rules permit AND-OR and THRESH to be used as the main connective of a deduction rule, allowing a single ARG to be inferred when appropriate information is available about the other ARGs. These rules are very useful, as they, in combination with the other four, permit a single predicate to be derived from an arbitrarily complex deduction rule. The special rules for AND-OR and THRESH are:

R2. From ${}_n\mathbb{W}_i^j(A_1,\ldots,A_n)$ and $n,i>0$ and ${}_{n-1}\mathbb{W}_{i-1}^{i-1}(A_1,\ldots,A_{k-1},A_{k+1},\ldots,A_n)$ to infer A_k.

R3. From ${}_n\Theta_i(A_1,\ldots,A_n)$ and $i<n$ and ${}_{n-1}\mathbb{W}_i^{n-1}(A_1,\ldots,A_{k-1},A_{k+1},\ldots,A_n)$ to infer A_k.

R4. From ${}_n\mathbb{W}_i^j(A_1,\ldots,A_n)$ and $j<n$ and ${}_{n-1}\mathbb{W}_j^j(A_1,\ldots,A_{k-1},A_{k+1},\ldots,A_n)$ to infer $\sim A_k$ [mark A_k FALSE].

B5. From ${}_{n}\Theta_{i}(A_{1},\ldots,A_{n})$ and $0<i<n$ and ${}_{n-1}W^{i-1}_{i-1}(A_{1},\ldots,A_{k-1},A_{k+1},\ldots,A_{n})$ to infer $\sim A_{k}$ [mark A_{k} FALSE].

Further postulates, shown in table 1, aid in specifying the interaction of various connectives and truth values. They follow from the earlier definitions of the connectives. In conjunction with the rules given above, they permit the substitution of one construction for another, which hopefully is simpler (fewer components) or easier to prove.

A1 describes THRESH in terms of AND-OR. Since the relationship is essentially an equivalence (as are those in A2 through A8), R6 allows either argument to be used as the antecedent. A2 and A3 permit the elimination or introduction of certain nestings of AND-ORs, notably for the pure conjunctive and pure disjunctive cases. It is impossible to deal with the general case so easily, as every AND-OR may not have a unique set of arguments. A4 relates the negation case of AND-OR as applied to a general form to a different nesting. A5 and A6 treat the simplification of AND-ORs with arguments of known value, while A7 and A8 deal with the possibility of out of bound parameters. Finally, A9 and A10, which are entailments rather than equivalences, permit the rewriting of certain AND-OR constructions while preserving truth.

Table 1.

Postulates

A1. $${}_2\Theta_1({}_n\Theta_i(A_1,\dots,A_n),{}_2\mathbb{X}^1_1({}_n\mathbb{X}^n_n(A_1,\dots,A_n),{}_n\mathbb{X}^{i-1}_0(A_1,\dots,A_n)))$$

A2. $${}_2\Theta_1({}_m\mathbb{X}^m_m({}_{n_1}\mathbb{X}^{n_1}_{n_1}(A_1,\dots,A_{n_1}),\dots,{}_{n_m}\mathbb{X}^{n_m}_{n_m}(A_{\left(\sum_{j=1}^{m-1}n_j\right)+1},\dots,A_{\sum_{i=1}^{m}n_i})),$$
$${}_{\sum_{i=1}^{m}n_i}\mathbb{X}^{\sum_{i=1}^{m}n_i}_{\sum_{i=1}^{m}n_i}(A_1,\dots,A_{\sum_{i=1}^{m}n_i})),\ A_v\neq A_w$$

A3. $${}_2\Theta_1({}_m\mathbb{X}^m_1({}_{n_1}\mathbb{X}^{n_1}_1(A_1,\dots,A_{n_1}),\dots,{}_{n_m}\mathbb{X}^{n_m}_1(A_{\left(\sum_{j=1}^{m-1}n_j\right)+1},\dots,A_{\sum_{i=1}^{m}n_i})),$$
$${}_{\sum_{i=1}^{m}n_i}\mathbb{X}^{\sum_{i=1}^{m}n_i}_1(A_1,\dots,A_{\sum_{i=1}^{m}n_i})),\ A_v\neq A_w$$

A4. $${}_2\Theta_1({}_1\mathbb{X}^0_0({}_n\mathbb{X}^j_i(A_1,\dots,A_n)),$$
$${}_2\mathbb{X}^2_1({}_n\mathbb{X}^{i-1}_0(A_1,\dots,A_n),{}_n\mathbb{X}^n_{j+1}(A_1,\dots,A_n)))$$

A5. $${}_2\Theta_1({}_n\mathbb{X}^j_i(A_1,\dots,A_{k-1},F,A_{k+1},\dots,A_n),$$
$${}_{n-1}\mathbb{X}^j_i(A_1,\dots,A_{k-1},A_{k+1},\dots,A_n))$$

A6. $${}_2\Theta_1({}_n\mathbb{X}^j_i(A_1,\dots,A_{k-1},T,A_{k+1},\dots,A_n),$$
$${}_{n-1}\mathbb{X}^{j-1}_{i-1}(A_1,\dots,A_{k-1},A_{k+1},\dots,A_n))$$

A7. ${}_2\Theta_1({}_n\mathbb{X}^j_i(A_1,\dots,A_n),F)$ where $i>j$ or $i>n$

A8. ${}_2\Theta_1({}_n\mathbb{X}^j_i(A_1,\dots,A_n),{}_n\mathbb{X}^n_i(A_1,\dots,A_n))$ where $j\geq n$

A9. ${}_n\mathbb{X}^j_i(A_1,\dots,A_n)\rightarrow{}_n\mathbb{X}^{j+1}_i(A_1,\dots,A_n)$

A10. ${}_n\mathbb{X}^j_i(A_1,\dots,A_n)\rightarrow{}_n\mathbb{X}^j_{i-1}(A_1,\dots,A_n)$ where $i>0$

As the purpose of these postulates is merely to provide a method of manipulating network structures, rather than a formal proof theory, no attempt will be made to prove formal properties, such as consistency and completeness, for the system.

VI. Semantic Versus Structural Inference

As noted before, it is possible to represent all of these axioms in the network. Whether they really belong there is another question. Let us contrast a typical deduction rule, such as the one in figure 8, and one of the axioms, say A9. There is a striking difference. Notice that while the deduction rule contains a number of atomic constants, the only "constant" in the axiom is the quantifier, which is really an auxiliary node. In addition, it is not clear how to represent n arguments (where n is a variable) using the ARG relation, or how the network will do the arithmetic to find j+1 without "cheating" (using the specialized computational functions of the CPU). Similar problems are reflected in all the axioms. While all axioms have at least one atomic constant in the universal quantifier, and some have others in the parameters to AND-OR and THRESH, these nodes seem to be somehow different than nodes such as JOHN and AUTOMOBILE. Also, many of the axioms seem to require some sort of specialized "massaging" or condition checking (e.g. in A10, is i>0?) which seems outside the realm of the network we have discussed so far. Can this apparent difference between axioms and common inference rules be formalized, expanded, and used to make inferences more efficient?

To find an answer, we return briefly to classical logic. Logic traditionally has been syntactic, that is, it provides methods of manipulation based solely on the form of the objects with which it deals, rather than on their content, or semantics. In the semantic network, the form (structure or syntax) of objects is determined by the relations which connect the semantic concepts in the object. If an inference rule contains no fixed semantic concepts (atomic constants), then its applicability and results depend solely on the relations that exist, i.e. the syntax of the rule.

The network representations we have introduced have brought with them a number of new relations and, in many cases, new auxiliary nodes (e.g. quantifiers, truth values, parameters for connectives). With the exception of these new auxiliary nodes, the axioms are purely syntactic, as they depend only on the relations involved. However, the auxiliary nodes introduced by the logic do not have the same "feel" about them as do such atomic constants as BOY. They are merely convenient methods of storing information needed by a syntactic process. With this distinction in mind, all the axioms may be classified as syntactic inference rules, independent of semantic content.

The original conception of the network made a further distinction between nodes and relations. Relations are to be viewed as non-conceptual information, a sort of glue to hold the network together. As such, they cannot be discussed, visualized or dealt with directly in the network. To work with relations, as the axioms do, we must go outside the network. The obvious place to go is to the deduction rule interpreter. Recalling that the rules of the system are already here, we may recast the axioms into rule form to facilitate their incorporation into the interpreter. Those axioms which were stated as equivalences will become two rules, while those stated as entailments will become one rule. A typical rewriting is shown below.

RA9. Given ${}_nW_i^j(A_1,\ldots,A_n)$ to infer ${}_nW_i^{j+1}(A_1,\ldots,A_n)$.
[from A9]

In the network representation presently in use, several other possible inference rules are syntactic, and so should properly be included in the interpreter, rather than being constructed in the network. These rules deal with set membership and the subset - superset relations, and the transitive temporal relation BEFORE (see Shapiro, 1975b). The format for these rules is described just before the rules themselves.

$a \xleftarrow{R1} c \xrightarrow{R2} b$ is written as a R1-R2 b

RR1. Given x SUBSET-SUPERSET y and y SUBSET-SUPERSET z, to infer x SUBSET-SUPERSET z.

RR2. Given x MEMBER-CLASS y and y SUBSET-SUPERSET z, to infer x MEMBER-CLASS z.

RR3. Given x BEFORE y and y BEFORE z, to infer x BEFORE z.

As with all other rules in the interpreter, these rules are universally quantified over all variables appearing in them.

As work with the network and representations continues, a close watch should be maintained to insure that any inferences which are purely syntactic, rather than semantic, are removed from the network and placed in the interpreter.

VII. Conclusion

We have developed and presented a formalism for making inferences in a semantic network, and have drawn a distinction between two different types of inferences. This formalism was motivated by examining certain shortcomings and inconveniences inherent in classical two-valued logic, such as difficulty in representing a lack of truth value or contradiction and difficulty in expressing intuitively simple concepts. In developing the formalism, two new non-standard connectives, AND-OR and THRESH, were introduced, along with a four-valued logic and three non-standard quantifiers, ALMOST-ALL, NONE, and

CNE. The problems resolved by each of these introductions were discussed. It is hoped that this work will expand the usefulness and possibilities of semantic networks as models of human information systems, and that investigations will continue in this area, both to discover the strategies actually used by people and to incorporate them into improved models.

Bibliography

Anderson, A.R. and Belnap, N.D. Jr. *Entailment: The Logic of Relevance and Necessity* (vol. I). Princeton University Press, 1975.

Anderson, J.R. and Bower, G.H. *Human Associative Memory*. Winston and Sons, 1973.

Bechtel, R. and Shapiro, S.C. A Logic for Semantic Networks. Presented at the 1976 ACM Computer Science Conference. Available as Technical Report No. 47, Computer Science Department, Indiana University, Bloomington, March, 1976.

Belnap, N.D. Jr. How a Computer Should Think. *Contemporary Aspects of Philosophy*. Proceedings of the Oxford International Symposium, 1975a, forthcoming, 1976.

--------. A Useful Four-Valued Logic. *Modern Uses of Multiple-Valued Logic*, ed. G. Epstein and J.M. Dunn. Proceedings of the 1975 International Symposium on Multiple-Valued Logic. 1975b, Reidel, forthcoming, 1976.

Epstein, G. Synthesis of Electronic Circuits for Symmetric Functions. *IRE Transactions on Electronic Computers*. May, 1958.

Fredericksen, C.H. Representing Logical and Semantic Structure of Knowlege Aquired from Discourse. *Cognitive Psychology*. Volume 7, Number 3, July, 1975.

Hendrix, G.G. Expanding the Utility of Semantic Networks Through Partitioning. *Advance Papers of the Fourth International Joint Conference on Artificial Intelligence*. International Joint Council on Artificial Intelligence, 1975.

Kintsch, W. *Representation of Meaning in Memory*. Lawrence Erlbaum Associates, 1974.

Lindsay, R.K. Inferential Memory as the Basis of Machines Which Understand Natural Language. *Computers and Thought*, ed. E.A. Feigenbaum and J. Feldman. McGraw-Hill, 1963.

Martin, E. Jr. The Psychological Unreality of Quantificational Semantics. Working Paper, Philosophy Department, Indiana University, Bloomington, 1975.

Palme, J. The SQAP Data Base for Natural Language. *American Journal of Computational Linguistics*, Microfiche 24, 1975.

Quillian, M.R. Semantic Memory. *Semantic Information Processing*, ed. M. Minsky. MIT Press, 1968.

Shank, R.C. Identification of Conceptualizations Underlying Natural Language. *Computer Models of Thought and Language*, ed. R. Shank and K.M. Colby. W.H. Freeman, 1973.

Shapiro, S.C. *The MIND System: A Data Structure for Semantic Information Processing*. The Rand Corp., R-837-PR, 1971a.

-------. A Net Structure for Semantic Information Storage, Deduction, and Retrieval. *Advance Papers of the Second International Joint Conference on Artificial Intelligence*, British Computer Society, 1971b.

-------. An Introduction to SNePS. Technical Report No. 31. Computer Science Department, Indiana University, Bloomington, November, 1975a.

-------., Generation as Parsing from a Network into a Linear String. *American Journal of Computational Linguistics*. Microfiche 33, 1975b.

-------., Bechtel, R., McKew, J., and Eastridge, N. Implementing Inference Rules in a Semantic Network. forthcoming, 1976.

-------., and Wand, M. The Relevance of Relevance. Technical Report No. 46. Computer Science Department, Indiana University, Bloomington, March, 1976.

Shubert, L.K. Extending the Power of Semantic Networks. *Advance Papers of the Fourth International Joint Conference on Artificial Intelligence*. International Joint Council on Artificial Intelligence, 1975.

Simmons, R.F. Semantic Networks: Their Computation and Use for Understanding English Sentences. *Computer Models of Thought and Language*, ed. R. Shank and K.M. Colby. W.H. Freeman, 1973.

Thomas, J.C. Quantifiers and Question-Asking. IBM Technical Report (in preparation), 1975.

Woods, W.A. Semantics for a Question-Answering System. Ph.D. Thesis, Division of Engineering and Applied Physics, Harvard University, 1967.