INFERENCES IN QUANTIFIED SEMANTIC NETWORKS

Robert F. Simmons & Daniel Chester
Department of Computer Sciences
University of Texas
Austin, Texas 78712

NL-31

Abstract

Conventions are shown for representing information in semantic networks in a linear form called semantic case relations. Representations of variables, truth functions, and quantified statements are provided. Methods for answering questions from the resulting semantic predicates are illustrated and a computational procedure for answering questions from quantified semantic predicates is described.

Acknowledgements

The ideas and procedures presented here are the result of many discussions with Gary Hendrix, and Michael K. Smith. We're particularly grateful to M.K. Smith for helping to implement the LISP procedures for question-answering.

Introduction

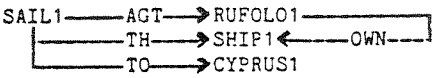
In a recent criticism, "What's in a Link", Woods (1975) noted that with few exceptions, most semantic network systems for representing English meanings were inadequately defined in several ways, and, in particular, lacked suitable conventions to encode the full meaning of quantified statements. Several recent papers, (such as those by Kay 1973, Hendrix 1975, Mylopoulous, et. al. 1975, Shapiro 1976, & Schubert 1975) have continued to explore representation conventions and have generally offered at least minimally acceptable schemes to encode quantificational data. Only rarely, however, is an algorithm described that uses those conventions for answering quantified questions. In our own experience, it is the algorithm for using representation conventions that reveals the computational costs and other consequences for any particular encoding.

In the following pages we briefly develop a quantified predicate notation for semantic networks, and describe a question-answering algorithm and its use for finding and distinguishing the meanings that are encoded.

Semantic Networks and Relations

A semantic network for representing aspects of the meaning of English discourse is comprised of a set of nodes that are interconnected by directed and labelled arcs. A node is a symbolic object that usually represents the conceptual referent of an English expression, although it may be a rule, the name of a function or program, or some special symbol. Arcs typically represent deep case relations (Bruce 1975) that hold between conceptual referents, but they also include special quantifier symbols, and pointers to rules associated with a given node.

If we consider the example sentence, "Rufolo sailed his ship to Cyprus", the following network might result from a semantic case analysis:



In this graph the convention is followed that a word suffixed by a number, e.g. SAIL1, is a uniquely named object that is an instance of the concept associated with the unsubscripted word. Thus, SAIL1 is an INSTance of SAIL. All lexical information is associated with the concept; e.g., SAIL is a kind of MOVE, is a verb, takes arguments of the form, AGT, THeme, FROM, TO, and has rules that define its preconditions and results.

If we were to express the meaning of the sentence in relational form,
(SAIL1 RUFOLO1 SHIP1 NIL CYPRUS1)
(RUFOLO1 OWN SHIP1)
we would depend on ordering conventions to recognize that the source or FROM of the sailing is NIL or not known, RUFOLO1 is the agent and CYPRUS is the TO or goal. Hendrix (1975) shows that the semantic case relations are a variant relational form in which the elements of the relational n-tuple are identified by their case arguments. The example appears as follows:
(HEAD SAIL1, AGT RUFOLO1, TH SHIP1,
TO CYPRUS1)
We follow the convention that the first element of an n-tuple is always its HEAD and thus show the semantic relation by,
(SAIL1 AGT RUFOLO1 TH SHIP1 TO CYPRUS1)
(RUFOLO1 OWN SHIP1)

The phrase "his ship" could be represented as, (OWN1 R1 RUFOLO1 R2 SHIP1), but since OWN is a binary relation, the simpler form, (RUFOLO1 OWN SHIP1) suffices. It should be noticed that the semantic case relation form is a special notation for a set of binary relations, e.g.((SAIL1 SUP SAIL)(SAIL1 AGT RUFOLO1) (SAIL1 TH SHIP1)(RUFOLO1 SUP RUFOLO)(RUFOLO1 OWN SHIP1)...) Each binary relation could be represented uniformly as (RELNAME R1 ARG1 R2 ARG2) but considerable savings in writing and in computation are gained by use of the mixed notation.

Nesting of relations in a manner analogous to embedding English relative clauses is natural in this form, so:
(SAIL1 AGT(RUFOLO1 OWN SHIP1)
TH SHIP1 TO CYPRUS1)
Every arc has an inverse, usually signified by adding the suffix, *, to its symbol. Thus the following semantic relations mean the same thing:
(RUFOLO1 AGT*(SAIL1 TH SHIP1 TO CYPRUS1)
OWN SHIP1)
(SHIP1 OWN* RUFOLO1
TH*(SAIL1 AGT RUFOLO1 TO CYPRRUS1))

(CYPRUS1 TO*(SAIL1 AGT RUFOLO1
TH (SHIP1 OWN* RUFOLO1)))

The last three expressions if given to a certain language generator would result in the English sentences:

It was Rufolo who sailed his ship to Cyprus.
It was his ship that Rufolo sailed to Cyprus.
It was to Cyprus that Rufolo sailed his ship.

From these examples we can see that in expressing a network in semantic case relational form, the conventions for nesting and the fact that every arc has its inverse allows for many alternative equivalent relational expressions of the same network. Each of the expressions when given to a semantic network compiler results in exactly the same network, because for every node-arc-node, Ni-arc-Nj, the compiler creates both Ni-arc-Nj and the inverse, Nj-arc*-Ni.

In general any semantic network can be translated into linear form by taking the starting node, Ni, as the first element of a relation, then taking Ni's first arc and the node which is the arc's value as the next two elements. If we desire to nest, the procedure is recursively followed on each value-node. If we wish an un-nested form, each node which is the value of an arc is put on a list and the procedure is iterated over the members of the list. Alternatively, we can produce the set of triples that represent a network by taking the first node and forming a triple with each of its arcs and that arc's value-node, and iterating the process for the value-nodes, until all value-nodes are terminals, i.e. produce no new triples.

It is also apparent that any set of relations can be represented by a semantic network. If we assume that an ordinary ordered n-tuple such as (SAIL RUFOLO SHIP NIL CYPRUS) is decoded by means of some template such as (ACT AGENT THEME FROM TO), then it can be translated into a semantic case expression by pairing each element of the template with the corresponding element of the n-tuple. The inverse operation can also be used to convert from a semantic case expression to the simpler n-tuple.

Semantic Predicates

If we add conventions to our semantic networks for marking truth values and for representing variables, truth functions and quantifiers, the heads of semantic case relations become logical predicates and we are justified in referring to them as semantic case predicates, or more simply, semantic predicates. It should be noted that ours is an omega order logic as our predicates can be n-ary relations between other predicates. Such "compound" predicates are propositions about how the other predicates are related and as such can have truth values.

In a network each subscripted instance of a concept, e.g. SHIP1, represents one or more members of its SUPerclass. The expression, (SHIP1 PLural T) signifies that more than one instance of the concept, SHIP, is referred to. The expression, (SHIP3 EQUIV SHIP) signifies that SHIP3 refers to every instance of the concept, SHIP. Symbols such as W, X, Y, Z are reserved to represent free variables, and often occur in rules such as,

(IMPLY ANTE (NOT OF (NOT OF X)) CONSE X).

A node representing a semantic relation can be marked True, False or UNDetermined. Two conventions are followed to reduce the amount of marking:

1. If an unmarked predicate is embedded in another (i.e. its node has backlinks such as AGT*, TH*, ...*,ignoring SUP, INST, BEFORE, AFTER, ENABLE, RESULTOF) the embedded predicate is dependent on the other and is UNDetermined unless there is a rule or convention that allows True or False to be inferred from the embedding predicate.

2. Otherwise a predicate represented by an unmarked node is True.

The previous example, "Rufolo sailed his ship to Cyprus"

(SAIL1 AGT RUFOLO1 TH SHIP1 TO CYPRUS1)

is taken as True. But, "Rufolo wanted to sail to Italy",

(WANT1 AGT RUFOLO1
TH (SAIL2 AGT RUFOLO1 TO ITALY1))

provides an embedded predicate, SAIL2. Under the first convention, this predicate is read as UNDetermined, while the embedding predicate, WANT1 is True under the second convention.

The truth functions, AND, OR, NOT, IMPLY are generally treated as compound predicates. For example, "Rufolo bought and sold jewels" is encoded:

(AND1 OF ((BUY1 AGT RUFOLO1 TH JEWEL1)
(SELL1 AGT RUFOLO1 TH JEWEL1)))
(JEWEL1 PL T)

The object AND1 is an instance of AND, and the arc, OF connects it with a list of predicates. Generally the English OR is taken as inclusive even in the following context:

"Rufolo decided to recoup his loss or die trying."

(DECIDE1 AGT RUFOLO1 TH(OR1 OF(RECOUP1 DIE1)))
(RECOUP1 AGT RUFOLO1 TH(LOSS1 ASSOC RUFOLO1))
(DIE1 TH RUFOLO1 DURING TRY1)
(TRY1 AGT RUFOLO1 TH RECOUP1)

It is possible that even if Rufolo acts on his decision, he might accomplish either, or both the acts of recouping and dying. As the sentence stands, the RECOUP, and DIE, predicates are embedded in OR1. The OR truth function signifies that one or both of the predicates is true but since we don't know which, the two predicates are taken as UNDetermined. OR1 is embedded as the TH argument of DECIDE so its value is also UND.

If we ask the question, "Did Rufolo die?" the above statement does not provide an answer; but it is relevant and the question-answering algorithm must find it and reserve DECIDE1 for further processing. This will be discussed in the

next section.

For the sentence, "Rufolo decided either to recoup his losses or die trying," the representation would be as before except that XOR1 would be used in place of OR1. If we introduce an implication rule that states that if X decides to do Y, then Y happens,
(IMPLY1 ANTE(DECIDE AGT X TH Y) CONSE Y)
then we could use it to assign a truth value to what Rufolo decided to do. DECIDE1 instantiates the rule, RUFOLO1 binds to X, and (XOR1 OF (RECOUP1 DIE1)) becomes the value of Y in both occurrences of Y in the rule. Since the ANTEcedent, DECIDE1, is True, the CONSEquent XOR1 is taken as True.

If we establish that, "Rufolo didn't die,"
(NOT1 OF (DIE1 TH RUFOLO1))
and we have the rule,
(IMPLY2 ANTE(AND OF((NOT OF X)(XOR OF (X Y))))
CONSE Y)
which means "IF NOT X AND EITHER X OR Y, THEN Y", then by binding DIE1 to X, and RECOUP1 to Y throughout the rule, we can establish that Rufolo recouped his losses.

Although this is a valid argument pattern, unfortunately the rule IMPLY1 is fallacious on semantic grounds. If a person decides to do something, then he will try to do it, but if X tries Y, Y still remains UNDetermined. If we use such rules as IMPLY1 with arguments whose truth values are UND, we can suggest expectations that may be substantiated by further text, but we cannot deduce truth.

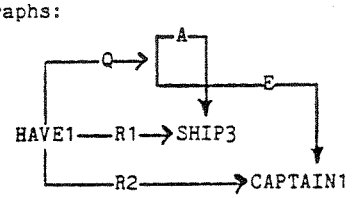
Additional implementation conventions are used to minimize the number of nodes and arcs required to represent conjunction. First, the value of any argument arc from a predicate node is a list of one or more nodes--an implicit representation of AND. Second, the set of semantic predicates in the network forms an implicit conjunction. The explicit AND is only recorded when additional arguments apply to all members of the conjunction as in the example, "Rufolo danced and sang while the music played."
(AND2 OF (DANCE1 SING1) DURING PLAY1)
(DANCE1 AGT RUFOLO1)(SING1 AGT RUFOLO1)
(PLAY1 TH MUSIC1)
Representations of OR and NOT are always explicitly encoded.

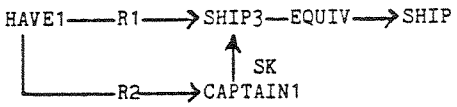
In all the above examples, universal quantification has been assumed. BUY1, RUFOLO1, and JEWEL1 are predicates true of particular instances of their superset classes, BUY, RUFOLO, and JEWEL. (BUY1 AGT RUFOLO1 TH JEWEL1) asserts that EVERY RUFOLO1 BOUGHT EVERY JEWEL1. Additional conventions are needed for a more general treatment of quantification.

In the example, "Every ship has a captain"
WRONG:(HAVE1 R1 (SHIP2 EQUIV SHIP) R2 CAPTAIN1)
we would have a false representation that signifies that every ship has CAPTAIN1 as its captain. What is meant is that each ship has some person as its captain and the person is not necessarily the same for every ship.

In a customary predicate logic notation the quantifier symbols (A FORALL, E FORSOME) are associated with the variables in the order in which they are to be applied, e.g.,
A SHIP3 E CAPTAIN1 (HAVE1 R1 SHIP3 R2 CAPTAIN1)
This convention can easily be adopted for semantic predicates with the following notation,
(HAVE1 R1 SHIP3 R2 CAPTAIN1 Q(A SHIP3, E CAPTAIN1))
This notation is sufficient to maintain the information about quantifiers and their ordering. The arc, Q, stands for a Quantifier ordering prescription for the predicate node to which it is attached.

For the sake of more effective computation we have found it desirable to treat the Q arc as a function which transforms its predicate into a Skolem form as below:
(HAVE1 R1(SHIP3 EQUIV SHIP) R2(CAPTAIN1 SK(SHIP3)))
These two quantifier forms have the following graphs:





The second or lower graph provides the simplest structure.

The first graph is referred to as the Q-arc form and it appears to be a desirable intermediate representation that will allow transformations into a canonical quantifier form using quantifier transformations such as those described by Quine (1959) and Chang and Lee (1973). The canonical form we have chosen is one in which the negation if any, is brought to bear on the predicate by transforming every negated quantifier. The following examples show how two logically equivalent statements become identical in canonical form:

E1 Not every ship sails every ocean.
E2 Some ship doesn't sail some ocean.

Their Q-arc forms:
QE1 (SAIL4 INSTR SHIP4 LOC OCEAN1
Q (NOT A SHIP4 A OCEAN1))
QE2 (NOT2 OF (SAIL5 INSTR SHIP5 LOC OCEAN2
Q (E SHIP5 E OCEAN2)))
By pushing the NOT of QE1 through A SHIP4 we get,
Q(E SHIP4 NOT A OCEAN1)
Then pushing NOT through A OCEAN1, we get,
Q(E SHIP4 E OCEAN1 NOT)
The NOT now applies to the main predicate rather than to the quantifiers, so we embed the predicate in it:
(NOT2 OF (SAIL4 ... Q(E SHIP4 E OCEAN1)))
which is identical to the form of QE2.

After transformation to canonical form, the statements are brought into the following Skolemized form;
(NOT2 OF (SAIL4 INSTR SHIP4 LOC OCEAN1))

It is worth noticing that this simple form is the default for what ordinary English statements such as "a ship didn't sail the ocean" signify in quantified form. It will be seen in the next section that the procedure for matching quantifier conditions assumes that a term not explicitly marked by an SK arc is not dependent on any other quantified term.

The Inference Algorithm

In research oriented toward the eventual development of a text understanding system we have studied and programmed several approaches to answering questions from semantic networks. These approaches reduce logically to the idea that a semantic network representing a discourse is an interconnected set of true statements. The lexical portion of the network contains additional true statements and rules for forming new true statements from existing ones. The question is taken as a hypothesis and the inference algorithm must accomplish the task of determining whether the question is TRUE, FALSE or UNDetermined with respect to the semantic network. It is also required to return an answer that instantiates any variables that are in the question.

A question is usually composed of class symbols, i.e. unsubscripted words, and case markers and variables. An answer is either a node whose associated arc-value pairs instantiate each element of the question or a general statement whose every element is instantiated by elements of the question. The content terms of a question are not limited to class symbols and subscripted terms may be included .

This paradigm is not the only one that is followed in question answering work and some criticisms of it will be discussed in the concluding section. Lehnert reports an unusual approach using scripts and conceptual dependencies (Lehnert 1976).

The following three examples will help to show the essential operation of the question answering procedure.

Q1. Did Rufolo sail to Cyprus?
(SAIL AGT RUFOLO TO CYPRUS)

Q2. Where did Rufolo go?
(GO AGT RUFOLO TO X)

Q3. Why did Rufolo go to Cyprus?
(GO AGT RUFOLO TO CYPRUS RESULTOF X)

The first question is answered by a direct match of a semantic predicate, (SAIL1 AGT RUFOLO1 TH SHIP1 TO CYPRUS1). It can be noticed in Q1 that the terms in the semantic representation are not subscripted. It is therefore possible to look directly into the lexicon to find the word, SAIL and retrieve its INSTances, SAIL1, SAIL2, etc. Each instance is a semantic predicate which is then examined to discover if it includes all the terms of the question.

The second question requires two miniscule inferences; first that SAIL is an instance of GO, and second that CYPRUS matches X. The first inference is accomplished with the use of the lexical structure, (SAIL SUP(MOVE SUP GO)). The relation SUP has the inverse INST, and SUP and INST are transitive, so SAIL1 is an INSTance of GO. Free variables such as X match anything, so once again, (SAIL1 ...TO CYPRUS1) is the answering predicate. The short answer is the element corresponding to X, CYPRUS1). The question words, what, where, who, etc. are treated very much the same as free variables except that each can limit the candidates it can match by a semantic class.

In order that the third question be answered, a discourse network such as the following is needed.
(WANT2 AGT RUFOLO1 TH DOUBLE1 ENABLE AND3)
(DOUBLE1 AGT RUFOLO1 TH WEALTH1)
(RUFOLO1 OWN WEALTH1)
(AND3 OF (BUY1 LOAD1) ENABLE SAIL1)
(BUY1 ....)
(LOAD1 ....)
(SAIL1 AGT RUFOLO1 ... TO CYPRUS1 RESULTOF AND3)
The two relations, ENABLE and RESULTOF are inverses and are transitive. If (X ENABLE Y) then X precedes Y and the conditions resulting from X include those that are pre-requisite to Y. Rule forms for computing causal links such as ENABLE and RESULTOF are described in another paper, (Simmons 1977).

The matching operation for Q3 is similar to that for Q2, and it is also matched by (SAIL1 ... RESULTOF AND3). Since (WANT2 ENABLE AND3), AND3 has the backlink, RESULTOF WANT2. A complete answer to a WHY question appears to be the entire causal chain, WANT2 ENABLE AND3 ENABLE SAIL1 ENABLE... In answering the why of an agentive act such as (GO AGT RUFOLO...) it is probably desirable to seek backward on the causal chain for a motive such as (WANT AGT RUFOLO...) and forward to some corresponding outcome such as (SELL AGT RUFOLO...FOR PROFIT). We have but little experience with WHY-questions, but refer the interested reader to an excellent discussion by Lehnert (1976). Questions concerning HOW MANY are treated most thoroughly in Woods (1969).

So far we have seen essentially a matching process supported by the limited inferences associated with the properties, INVERSE and TRANSITIVE. A more general approach to inference is given by an abbreviation of the IMPLY structure in the form of CONSEQUENT rules such as the following:

((X LOC Z)(Y PARTOF Z)(X LOC Y))

If a question matches the first element of the rule, e.g. (RUFOLO LOC ITALY) then the elements of the question corresponding to the variables X and Z are bound to these variables throughout the rule,
((RUFOLO LOC ITALY)(Y PARTOF ITALY)(RUFOLO LOC Y)).

The first element of the rule is then detached and the remainder is then substituted as a new set of questions which if successfully answered prove the first element is true. Thus if the semantic data base contains,
(RUFOLO LOC RAVELLO) (RAVELLO PARTOF ITALY)
then, (RUFOLO LOC ITALY) is True.
A more complicated example is:
((GO AGT X TO Y INSTR Z)
(WANT AGT X TH(Y LOC* X))
(X LOC Z) (CONTROL AGT X TH Z))
This rule might be used to answer the question, "How can Rufolo go to Cyprus?" Rufolo can go to Cyprus by ship, if he wants to be located at Cyprus, is located at a ship, and if he controls the ship.

This is the form of consequent rule described by Fischer Black in 1964. Since then THANTES, THCONSES, and rule-forms for establishing pre- and postconditions have occurred frequently in AI literature.

We can now describe the question-answering procedure in the following steps,

1. For each question obtain a set of nodes that are candidates for answers:

   a. Candidates are nodes that are in EQUIV, INST, or SUP relations to the content terms of the question.

   b. If the first term of the question is a variable, transform the question so the first term is a word. If there are only variables in the question refuse it.

2. For each candidate, for each arc-value pair in the question, match the arc-value pair in the candidate. Reject any candidate that does not match every arc-value pair of the question.

   a. Two arc-value pairs match if the arcs are identical and if the candidate value QIMPLIES the question value, i.e. (QIMPLY CV QV).

   b. QIMPLY CV QV is true if: CV=QV, CV is a variable, or QV is a variable, or if CV is an INSTance or EQUIV to QV, or if CV is a SUPerset of QV or of its EQUIVs.

   c. If QIMPLY fails and the arc is transitive and the CV is connected by an identical arc to some CV', then QIMPLY CV' and QV.

   d. If the arc in the question is an EQUIV* or SKolem it signifies a quantification condition on the question. (A EQUIV* B) is satisfied only if the CV corresponding to A is EQUIV or SUP to B. (A SK B) is satisfied only if the CV corresponding to A has no SK arc, signifying a free variable, or has an SK arc whose value is an ordered subset of the value of A's Sk value.

3. Save the surviving candidates as answers.

4. For each question get every inference rule associated with its first term and bind the corresponding elements of the question with the variables in the rule.

5. For every bound rule, detach the first predicate expression, and repeat steps 1 through 5 on the remaining predicate expressions.

This procedure, a direct descendent of Fischer Black's and of Schwarz, Burger and Simmons, finds all answers to a set of questions. It can be noticed that if the system had inference rules associated with every node, it would not terminate. Also certain classes of rules can establish infinite recursions (see Black 1969). Although these events can be avoided in other ways, the procedure can be protected in steps 2 and 3 by limiting the number of questions and answers that it is allowed to accumulate. The function QIMPLY is incomplete by design. It uses only SUPerset, INStance, and EQUIValence arcs to infer the match of a question term with a candidate word and does not apply general IMPLY rules. Experience will show if the proportion of answers it misses is outweighed by the irrelevant computations it eliminates.

Truth functions and truth values are invisible to this procedure. If a semantic predicate is QIMPLYed by a candidate answer, the candidate is returned as relevant to the question. A higher level function examines the truth functions on both the question and its relevant candidates to mark each candidate as True, False or UNDetermined with respect to the question. If no candidate matches, the null answer is taken as unknown, and the truth value of the question is UNDetermined.

Our current system is implemented in about twenty concise LISP functions. Three functions of great utility are FOR1, FORSET, and FORALL. These are mapping functions of three arguments; a variable, a set, and a function with its arguments (usually including the variable). The variable is assigned to the first member of the set, and the function is then evaluated. FOR1 is satisfied if one member of the set causes the function to return a non-nil value; FORALL requires every member of the set to cause the function to return non-nil; and FORSET goes through the entire set and returns the set of non-nil values. The functions are patterned after the quan- functions that were used by Woods (196- airlines data base work.

The system is organized in a depth-first search, but a best-first search is easily arranged by applying an appropriate ordering function to the sets of candidate nodes. For a relatively small data base, the overhead for computing best-first is probably too great to justify its use. A switch called MODE is provided to substitute FOR1 for certain calls to FORSET, thus providing a single answer mode.

Figure 1 is a brief definition of a set of LISP functions that outline the top-level organization of the inference system.

Discussion

In previous sections we have introduced conventions for representing variables, truth functions and quantification in our form of semantic relations and taken this as justification for using the term, semantic predicates. The question answering procedure was described as an inference method that is computationally effective. The intent was not to argue that answering all English questions is simply a theorem proving operation, but rather to show that deductive question answering is one aspect of questioning a textual data base that can be clearly defined.

We noticed in Section III that a statement with an UNDetermined truth value, may nonetheless be relevant to a question and conceivably participate in further computations to establish an answer. In that section it was also apparent that it is easy to write plausible rules of inference that are in fact, false. Generally our experience with inference rules on English meanings indicates that they suggest possibilities that may be validated by the preceding or oncoming text. Along with several others in the field, we doubt that ordinary English text is organized or understandable in purely deductive fashion. Instead of establishing deductive chains, ordinary discourse creates plausible connectivities. For example, " Rufolo was a wealthy man. He wanted to double his wealth. He bought a ship, loaded it with goods that he paid for himself, and sailed to Cyprus." What will he do in Cyprus? What will he do with the goods? How will he double his wealth? These and others are questions for which the text suggests possible answers that can be obtained by the use of rule-forms that look like deductive inference rules but which will establish only plausible outcomes.

The text provides additional sentences, such as: " Rufolo arrived in Cyprus, he discovered many ships carrying the same goods. He was forced to sell at a loss. In fact he was ruined." The new statements support previous plausible inferences that Rufolo was on a trading voyage, that his intention was to double his wealth by selling goods and that because of competition he lost his wealth. So ordinary narrative discourse suggests many plausible connections, continuations, causal and purposive chains which must be tested against the definite facts of the narrative as they emerge. This is where rules for causal organization (Simmons 1977) and scripts (Schank 1975) are applied to augment the text with what we expect is the author's intention.

On the other hand, some deductive inferences are possible. If Rufolo bought the goods, he paid for them. If he paid for goods, he bought them. Goods are merchandise. If he loads a ship with goods, the goods are on the ship. If he wants to double his wealth he will either succeed or fail to do so. If he is a wealthy man, he is a man.

Our point of view is to accept deductive logic as one mode of tnought needed for understanding language and a primary method for establishing that two statements may mean the same thing. Deductions are a necessary part of any more sophisticated system for plausible inference. And in applying rules of plausible inference, variables must still be bound and simple questions must be answered from the discourse content with the use of deductive inferences that preserve whatever truth values inhere.

REFERENCES

Black,F., A Deductive Question Answering System, in Minsky,M.(ed.) SEMANTIC INFORMATION PROCESSING, MIT Press, Boston,1969.

Bruce,B.C., Case Systems for Natural Language, Bolt Beranek and Newman Inc., BBN Rept. No. 3010, A.I. Rept No. 23, Cambridge, Mass., 1975.

Chang,C., & Lee, R.C. SYMBOLIC LOGIC AND THEOREM PROVING, Academic Press, New York, 1973.

Hendrix,G.G., PARTITIONED NETWORKS FOR THE MATHEMATICAL MODELING OF NATURAL LANGUAGE SEMANTICS, Dept. Comp. Sci. Tech. Rep. NL28,Univ. Tex.,(Diss.),Austin,1975.

Hendrix,G.G., Expanding the Utility of Semantic Networks through Partitioning, 4IJCAI,1975,115-121.

Kay,M, The Mind System, in Rustin,R.(ed) NATURAL LANGUAGE PROCESSING, Algorithmics Press, New York,1973.

Lehnert,W., Question Answering in a Story Understanding System, COGNITIVE SCIENCE, vol 1,#1,1977,47-73.

Mylopoulos,J.,Cohen,P.,Borgida,A., & Sugar,L. Semantic Networks and the Generation of Context, 4IJCAI, 1975.

Quine,W.V.O., METHODS OF LOGIC, Henry Holt,New York,1959.

Schank,R.C., Using Knowledge to Understand, in Schank,R. & Nash-Webber, B.L.(eds), THEORETICAL ISSUES IN NATURAL LANGUAGE PROCESSING,BBN,Cambridge,1975 117-121.

Schubert,L.K., Expanding the Expressive Power of Semantic Networks, 4IJCAI,1975,158-164.

Schwarcz,R.,Burger,J.,& Simmons,R., A Deductive Question- Answerer for Natural Language Inference, CACM,vol 13, #3,1970,167-183.

Shapiro,S.C., An Introduction to SNePS (Semantic Net Processing System). Tech. Rept. No. 31, Comp. Sci. Dept., Indiana Univ., Bloomington, 1976.

Simmons,R.F., Rulebased Computations on English, Dept.Comp. Sci., Tech. Report NL31, Univ. of Texas, Austin, 1977.
Woods,W.A., SEMANTICS FOR A QUESTION-ANSWERING SYSTEM, Report NSF19, Aiken Lab., Harvard,(Diss.) 1969,
Woods,W.A., What's in a Link, in Bobrow,D. & Collins,A.,(eds) REPRESENTATION AND UNDERSTANDING, Academic, New York, 1975.

```
(ANSQ(LAMBDA(QSET)
     (FORALL Q QSET (QANS Q)) ))
(QANS (LAMBDA(Q) (PROG (ANS QI)
     (SETQ ANS
        (FORSET QI (CANDS (CAR Q)) (ASK Q QI) ),
     (RETURN (APPEND ANS
       (FORSET QI(GET(CAR Q)'TRULE)(ANSQ(BIND Q QI)) ))))))
(ASK(LAMBDA(Q QI)
     (FORALL PAIR (CDR Q) (MATCHPAIR PAIR QI)) ))
(MATCHPAIR (LAMBDA(PR QI)
     (COND((NULL (SETQ J (GETPAIR QI (CAR PR)))) NIL)
        (T(QIMPLY J (CADR PR))) )))
(CANDS (LAMBDA(WD)
  (APPEND(INSTANS WD)(APPEND (EQUIVS WD)(SUPSETS WD))) ))
```

Notes: Three mapfunctions FOR1, FORSET, FORALL bind their 1st argument to each member of the set which is the second argument, and then evaluate their 3rd argument. FOR1 is satisfied with one value, FORSET witn any number of values greater than zero, and FORALL requires that every member of the set have a non-nil value.

Figure 1. Top-Level Organization of the Inference System