INFERENCES IN QUANTIFIED SEMANTIC NETWORKS

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

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.

Acknowiedgementg

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

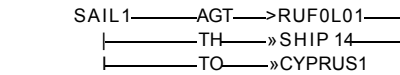
In a recent criticism, "What's in a Link", Woods (1975) noted that with few exceptions, most semantic network systems for representing English meanings were inadequately defined in several ways, and, in particular, lacked suitable conventions to encode the full meaning of quantified statements. Several 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 &nd 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 ooncept; 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 RUF0L01 SHIP1 NIL CYPRUS1)
(RUF0L01 OWN SHIP1)
we would depend on ordering conventions to recognize that the source or FROM of the sailing is NIL or not known, RUF0L01 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 RUF0L01, 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 RUF0L01 TH SHIP1 TO CYPRUS1)
(RUF0L01 OWN SHIP1)

The phrase "his ship" could be represented as, (0WN1 R1 RUF0L01 R2 SHIP1), but since OWN is a binary relation, the simpler form, (RUF0L01 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 RUF0L01) (SAIL1 TH SHIP1)(RUF0L01 SUP RUFOLO)(RUF0L01 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(RUF0L01 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:
(RUF0L01 AGT»(SAIL1 TH SHIP1 TO CYPRUS1)
OWN SHIP1)
(SHIP1 OWN* RUF0L01
TH*(SAIL1 AGT RUF0L01 TO CYPRRUS1))

(CYPRUS1 TO*(SAIL1 AGT RUFOL01
TH (SHIP1 OWN* RUFOL01)))

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 suoh 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 RUF0L01 TH SHIP1 TO CYPRUS!)

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

(WANT1 AGT RUF0L01
TH (SAIL2 AGT RUF0L01 TO ITALY1))

provides an embedded predicate, SAIL2. Under the first convention, this predicate is read as UNDetermined, while the embedding predicate, WANT! 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 RUF0L01 TH JEWEL1)
(SELL1 AGT RUF0L01 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 RUF0L01 TH(0R1 0F(REC0UP1 DIED))
(REC0UP1 AGT RUF0L01 THU0SS1 ASSOC RUF0L01))
(DIE1 TH RUF0L01 DURING TRY1)
(TRY1 AGT RUFOLO! TH REC0UP1)

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 0R1. 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. 0R1 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 exoept that X0R1 would be used in place of 0R1. If we introduce an implication rule that states that if X decides to do Y, then Y happens,

(IMPLY 1 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, RUF0L01 binds to X, and (X0R1 OF (REC0UP1 DIED) becomes the value of Y in both occurrences of Y in the rule. Since the ANTEcedent, DECIDE1, is True, the CONSEquent X0R1 is taken as True.

If we establish that, "Rufolo didn»t die,"

(N0T1 OF (DIE1 TH RUF0L01))

and we have the rule,

(IMPLY2 ANTECAND 0F((N0T 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 REC0UP1 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 RUF0L01)(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 RUF0L01 TH JEWEL1) asserts that EVERY RUF0L01 BOUGHT EVERY JEWELL 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 oaptain. 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,

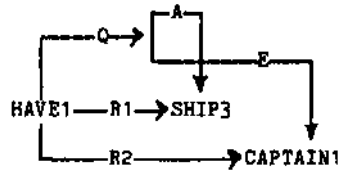
(HAVE1 R1 SHIP3 R2 CAPTAIN1 Q(A SHIP3, E CAPTAIN1))

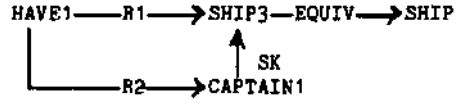
This notation is sufficient to maintain the information about quantifiers and their ordering. The arc, Q, stands for a Quantifier ordering prescription for the predicate node to which it is attached.

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 RKSHIP3 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 predioate 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 0CEAN1
Q (NOT A SHIP4 A 0CEAN1))
QE2 (N0T2 OF (SAIL5 INSTR SHIP5 LOC 0CEAN2
Q (E SHIP5 E 0CEAN2)))

By pushing the NOT of QE1 through A SHIPU we get,

Q(E SHIP4 NOT A 0CEAN1)

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:

(N0T2 OF (SAILM ... Q(E SHIP4 E 0CEAN1)))

which is identical to the form of QE2.

After transformation to canonical form, the statements are brought into the following Skolemized form;
(N0T2 OF *(SAIL4* INSTR SHIP4 LOC 0CEAN1))

It is worth noticing that this simple form is the default for what ordinary English statements such as " 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 RUF0L01 TH SHIP1 TO CYPRUS1). It can be notioed in Q1 that the terms in the semantic representation are not subscripted. It is therefore possible to look directly into the lexioon 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)
(RUF0L01 OWN WEALTH1)
(AND3 OF (BUY1 L0AD1) ENABLE SAIL1)
(BUY1_____)
(L0AD1_____)
(SAIL1 AGT RUF0L01 ... 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 2)(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.

   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^1 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.

M. 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 FORI, 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. FORI 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. These functions are patterned after the quantifier functions that were used by Woods (1969) in his 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 FORI 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 notioed 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 beoause 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 thought 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, MIJCAI,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 Inferenoe, 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)
      (CONDUNULL (SETQ J (GETPAIR QI (CAR PR)))) NIL)
          (T(QIMPLY J (CADR PR))) )))
(CANDS (LAMBDA(WD)
  (APPENDdNSTANS WD) (APPEND (EQUIVS WDMSUPSETS WD))) ))
```

Notes: Three mapfunctions FORI, FORSET, FORALL bind their 1st argument to each member of the set which is the second argument, and then evaluate their 3rd argument. FORI is satisfied with one value, FORSET with 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

be true of the patient. If a rule is satisfied, (by some set of ISPECs associated with the source node), it then specifies the creation of an ISPEC for the target node (the node on the "far" side of the link). The result of propagation across a link is a set of ISPECs for the target node, each one produced by a satisfied rule.

2.4 Propagation

When an ISPEC is generated for a node, all the links emanating from that node are potential candidates for propagating the effects of that ISPEC to neighboring nodes. Only those links which have an associated decision table can actually support propagation. For each such link, the decision table is evaluated, and the ISPECs produced are added to the ISPECs list for the node at the far end of the link. The process then repeats recursively, potentially producing a wave of propagation across the semantic net. Each new ISPEC is like a droplet which, when added to our pool of knowledge, causes a ripple which can propagate any distance through the pool.

2.5. Cones

Consider a semantic net node representing a disease, and a set of nodes representing the symptoms of that disease. Whenever an ISPEC is produced for one of the symptom nodes, the truth-status of the disease node may be affected. Furthermore, these effects are generally not independent. What we need in this situation is a way for the symptom nodes to interact in propagation to the disease node.

In IRIS, this is accomplished by associating the same decision table with all the links from symptom nodes to the disease node. We call such a configuration a propagation cone. It is usually visualized with the disease node at the apex, and the other nodes occupying the cone's base.

Cones can also be created in which the direction of propagation is from the apex to the base. In this case, a single decision table is associated with all the apex to base links. We then have a structure which can be characterized as a "scatter cone".

A simple semantic net link connects two nodes. Propagation is defined by the decision table associated with that link. A cone is a generalization of this basic structure. With cones, there can be a set of nodes rather than a single node on one side of the structure.

2.6. QUESTIONS

All of IRIS's specific medical knowledge must ultimately be based on information provided by the user. There is nothing we can conclude about a specific patient a priori. Accordingly, we need a mechanism by which specific medical knowledge can be entered. OUESTIONs provide such a facility.

An IRIS QUESTION consists of the following: a set of strings to output to the user, a set of checking functions to ensure that the user's input is in the proper form, and a set of mappings from permissible user inputs to the various actions IRIS can take. These actions include setting system variables, specifying questions to be asked next, and generating ISPECs. The set of ISPECs produced in this way comprises that part of IRIS's specific medical knowledge obtained directly from the user.

2.7 Vines

Once a consultation system has come to a set of conclusions about a patient, it must be able to communicate them to the user. IRIS does this by generating an english language string for each semantic net node which represents a conclusion. This string expresses how the concept represented by the node is true of the patient. We refer to these strings as "vines" because they can be visualized as festooning the semantic net.

There are two ways to generate a vine for a node. The default (taken when the node carries no explicit specification for vine generation) is to create a vine for the node using a standard procedure. This procedure produces a string of the form: {modifiers, time, node-name, strength of belief}. An example would be: "severe sudden past increased intraocular pressure [SB = .85]". Alternatively, we can associate a decision table with a node. This decision table can then specify different ways of generating the vine, or different pre-stored vines. The choice among the various possibilities can be based on the truth-status of any node or set of nodes.

1.5 Explanation A clinical consultation system must be able to explain or justify its decisions in terms acceptable to experts in its application domain. Systems that lack this ability have not gained wide acceptance even when they attained a satisfactory level of correctness in their decisions. Such an explanation should refer to the general medical knowledge on which the decision was based.

In IRIS, this is easily done. Suppose that there are two semantic net nodes, "injury" and "pain", which are connected by a "causes" link (i.e. "injury causes pain") which has an associated decision table. If IRIS is told that the patient has an injury, it can then conclude (via propagation) that the patient is in pain. When asked to explain this reasoning, IRIS can reply: "I conclude that pain is present because I know that there is an injury, and I know that injury causes pain."

3. ISPECS

As stated above, an ISPEC is an assertion that the concept represented by a semantic net node is true of a patient, or of the clinical situation. ISPECs are IRIS's only representation for specific medical knowledge. There are slots in each ISPEC for specifying details about how the concept represented by a node is true. We will now describe these slots, and the currently implemented data structures that fill them.

3.1 Node

The NODE slot of an ISPEC specifies which semantic net node it is an ISPEC for. This slot is usually not necessary, but is needed for propagation in scatter cones (see section 2.5), and for context dependent decision tables (see section 4.2) .

1.2. Strenght of Belief

The strength of belief (SB) slot of an ISPEC specifies how much IRIS believes the assertion represented by that ISPEC. The possible range is from full belief to full disbelief. IRIS has been designed so that the mechanisms for manipulating numerical strength of belief measures are easily changed or replaced. Currently, IRIS uses the strength of belief representation developed by Shortliffe for MYCIN (Shortliffe, 1974). This is described in greater detail in (Trigoboff, 1977).

This representation for strength of belief consists of two numbers: a measure of belief (MB), and a measure of disbelief (MD). Each of these can range from 0 to 1 with 0 representing no (dis)belief, and 1 representing full (dis)belief. The MD is subtracted from the MB to obtain the certainty factor (CF). The CF ranges from -1 (full disbelief) through 0 (no opinion) to 1 (full belief). The SB slot of each ISPEC is filled with a two-element list of the form (MB MD). Thus, IRIS's strength of belief in an ISPEC is derived by using the contents of its SB slot to generate a CF.

3.3 SIDE

The human body is bilaterally symmetrical, and contains many paired structures. An assertion about any such structure must therefore specify which one it refers to. Since IRIS is to perform as an ophthalmological consultant, it needs the ability to deal with this issue.

There are basically two ways to approach this problem. Either we can have semantic net nodes that represent "pain in the left eye" and "pain in the right eye", or we can have one node representing "pain in the eye" and rely on some notation in the ISPECs to tell us whether it is the left or the right eye. In IRIS, we have chosen the second course, avoiding an otherwise unnecessary duplication of semantic net nodes. Each ISPEC has a SIDE slot which can contain either LEFT, RIGHT, or NIL.

Not every node in the semantic net refers to a duplicated structure. "Diabetes" is a concept which is not side-related. Nodes of this sort are marked as SINGLE nodes. Any ISPEC for a SINGLE node must have a SIDE slot value of NIL. Conversely, any ISPEC for a non-SINGLE node must have a SIDE slot value of either LEFT or RIGHT.

3.4 Time

Clinical situations evolve through time, and the configuration of assertions describing a patient changes as new events become known to the consultation system. Any assertion must therefore be associated with a specification of the time interval it applies to. In IRIS, each ISPEC has a TIME slot. TIME slots are filled with a list of two dates of the form (from to). The first is the date the ISPEC initially became true of the patient, and the second is the final date the ISPEC was true.

The creation of ISPECs is controlled by means of create-expressions which specify characteristics for the ISPEC to be created. Values for the TIME slot are generally specified in create-expressions as one of the following: past, present, past-or-present, future. When an ISPEC is created from a create-expression, the current date (obtained from the system) is used to translate these specifications into IRIS's internal time representation (the list of two dates).

The time representation is part of the mechanism for dealing with multiple-visit cases. Most of the decision rules (i.e. decision table columns) in IRIS require that something be true of the patient in a specified time period. Again, these specifiers are generally chosen from: past, present, past-or-present, future.

If an ISPEC is created using a create-expression that specifies a time of present, its TIME slot will contain the

date it was created on. It will therefore be able to satisfy decision rules that specify a time of present. At a subsequent visit, this ISPEC will no longer satisfy those decision rules. Instead, it will now satisfy ones that specify past-or-present or past.

3.5 Modifiers

The SB and TIME slots are necessary for all ISPECs. The SIDE slot is needed in a high percentage as well (relatively few nodes in IRIS's semantic net are SINGLE). There are other slots which we need only occasionally. For example, ISPECs for the node representing "intraocular pressure" need a VALUE slot, so that when we measure a patient's intraocular pressure, we can express that measurement with an ISPEC. Thus, an intraocular pressure of 35 would result in an ISPEC with a VALUE slot value of 35.

It would be wasteful to allocate a VALUE slot in all ISPECs, since only a few nodes require it in their ISPECs. DEGREE, COLOR, and WIDTH are other such occasionally-needed slots. We have dealt with this in IRIS by implementing a modifiers (MODS) slot in the ISPEC. The format of the value of the MODS slot is (propertyl valuel property2 value2 ...). Thus, in the example above, the semantic net node "intraocular pressure" would get an ISPEC with a MODS slot value of (VALUE 35). This allows us to specify any slot we desire in any ISPEC without the overhead involved in explicitly defining that slot for all ISPECs.

We do not have a semantic net node representing "severely increased intra-ocular pressure". ISPECs representing severely increased intraocular pressure are actually ISPECS for the node "increased intraocular pressure" which have a MODS slot value of (DEGREE SEVERE). The MODS slot thus allows ISPECS to further specify the concepts represented by nodes.

If this were not the case, there would be a limit to the resolution of IRIS's representation for specific medical knowledge. If "increased intraocular pressure" were the highest resolution node available, then ISPECs without a MODS slot would provide us with no way of representing "severely increased intraocular pressure". The MODS slot thus allows us to achieve any level of resolution we desire in our representation of specific medical knowledge.

3.6

Each ISPEC has a TYPE slot. The value of this slot specifies the way in which its ISPEC is to be semantically interpreted. This is a feature which provides IRIS's knowledge representation with considerable power. There is no limit to the number of ISPEC types that can be defined.

Currently, IRIS has seven ISPEC types. They are: NIL, FAMILYHISTORY, PTHISTORY, CHOSEN, COVEREDBY, SUBSUMEDBY, and TREATEDBY NIL is the basic, standard ISPEC type. An ISPEC of type NIL means that the fact it represents is (or was) believed true (with the specified strength of belief) of the patient during the specified time. FAMILYHISTORY ISPECs represent facts which are true of people in the patient's family. PTHISTORY ISPECs represent facts about the patient which have been obtained from the patient,

ISPECs of type CHOSEN are very important in the operation of IRIS. These ISPECs are used to represent decisions made by the system. When IRIS chooses a diagnosis (by whatever strategy), the fact of this choice must be included in the store of patient-specific knowledge. IRIS will then be able to use this fact in its subsequent medical reasoning. The other ISPEC types represent the various sorts of inferences IRIS can make. A disease or treatment node is given a CHOSEN ISPEC when IRIS has decided to either make that diagnosis or perform that treatment.

SUBSUMEDBY ISPECs propagate from disease nodes which have enough positive strength of belief to be candidates for being CHOSEN. They propagate to other disease nodes which should not be CHOSEN when there is a sufficiently high level of belief in a subsuming disease.

When a disease node is given a CHOSEN ISPEC, it can then cause propagation of COVEREDBY ISPECs. COVEREDBY ISPECs propagate from CHOSEN disease nodes to symptom nodes that the disease can explain. This is part of IRIS's mechanism for knowing whether any symptoms remain unexplained.

When a treatment node is given a CHOSEN ISPEC, it can then cause propagation of TREATEDBY ISPECs. ISPECs propagate from CHOSEN treatment nodes to disease manifestations that the treatment has an effect on. This is part of IRIS's mechanism for knowing if every need for treatment can be satisfied by the current treatment regimen.

Au Mciaifin l&klej&

IRIS's decision tables control propagation of ISPECs along the links of the semantic net. A decision table can be thought of functionally as a set of rules, but the actual mechanisms of a system based on decision tables differ somewhat from those of a rule based system.

A decision table in IRIS consists of a set of conditions, and a set of columns. Each condition is a predicate which tests whether the fact (or facts) it specifies is true of the patient. Columns function

as independent decision rules. Each column tests for a specified pattern of truth-values for a subset of the conditions. A column is said to be true if the truth-values of the conditions match its pattern. In this case, an ISPEC is created according to a specification also contained in the column. The ISPECs that propagate across a link are those created by the true columns of the link's decision table.

There are certain patterns of propagation which we need to specify for many different links in the semantic net. One such pattern is used for most of the "cause3" 1inks in the semantic net. The pattern can be paraphrased as: "If A causes B, and A is true, then conelude B with a strength of belief equa1 to the strength of belief in A times a constant associated with the causal link. Specify that B occurs in time after A.",

We want to be able to implement sueh a pattern as a single decisi on tab1e which can be associated with more than one semantic net link, This is accomplished in IRIS through the use of context dependent decision tables. Propagation is always done with respect to a particular link, That link is the propagation context. A context dependent decision table specifies how to create an ISPEC for the node on the far side of the 1ink, if the node on the near side of the link is true. The identities of these nodes are obtained from the propagation context (the link being propagated with respect to), rather than from the decision table. Thus, we need to specify only one decision table that embodies the desired propagation pattern.

Zu. Xk£ *LLQ* £.£££§.££

The basic task facing any AIM system is to draw conelusions about the clinical situation, based on current knowledge of that situation. In IRIS, this task is performed by the propagation process. So far, we have described this process at the level of single semantic net links. While a description at this leve1 can give an intuitive view of how propagation works, there are issues which become apparent only when propagation is viewed as occurring within a network rather than along a single link Propagation is a locally defined process which operates successfuny because •certaln global aspects of its behavior have been taken into account.

The propagation process is designed to be quite general and flexible. We want a system which can automatically make all the desired inferences from a given set of facts with respect to a given model. Furthermore, if we then change some fact or facts, we want the set of conclusions to be altered automatically to reflect the new set of facts. The results of the propagation process will then be dependent only on changes in the input data. Any such change wil1 result in an appropriate change in the output of the model without any further intervention on the part of the user.

When propagation is attempted aeross a link, there is always the possibility that propagation has prevlously taken place across that same link. This complicates the logic of propagation somewhat. When propagation takes place across a link, the decision table associated with that link is evaluated. This produces a list of ISPECs for the target node. The target node may aiready have ISPECs which propagated from the source node, so the following three conditions must be checked for in determining whether to alter the ISPECs on the target node. If a column of the decision table which was formerly not true is now true, the ISPEC produced must be added to the target node's ISPECs. If a column of the decision table which was formerly true is not now true, the ISPEC it produced must now be deleted from the target node's ISPECs. Finally, if a formerly true column is still true, but has now produced an ISPEC which is different in some respect (e.g. the SB slot value is a function of the strength of belief in the node being propagated from), the ISPEC it produced must be replaced by the new one. These ISPEC matching tests are based on information carried within each ISPEC specifying the decision table and column which produced it. If any of the above conditions is true, the ISPECs of the target node are changed, leading in turn to propagation from the target node.

LM. Clinical Strategy

IRIS's clinical strategy is impleraented through the six nodes "chosen diagnoses", "possible diagnose tiunexplained syraptoms" chosen treatments", "possible treatments", and "untreated pathology". Any symptorn for which an explanation is necessary propagates an ISPEC to "unexplained symptoras" when it is found to be true of the patient In the same way, any condition which needs to be treated propagates an ISPEC to "untreated pathology" IRIS attempts t0 choose a diagnosis '(or diagnoses) which will explain the maximum number of unexplained symptoms, and a treatment regimen which will halt as many pathological processes as possible.

Initiany, IRIS collects information about the patient. At the end of this phase, IRIS's knowledge of the patient consists of the current set of ISPECs in

the semantic net. These ISPECs have been produced directly from answers to questions, or indirectly as the result of propagation. Any disease which has collected a sufficiently high CF via propagation will in turn have propagated an ISPEC to "possible diagnoses'".

IRIS now chooses a diagnosis. Each disease node which has propagated an ISPEC to "possible diagnoses" is taken in turn and given an ISPEC of type CHOSEN. This causes COVEREDBY ISPECs to propagate from that disease node to all symptoms whose presence the disease can explain. Symptom nodes with COVEREDBY ISPECs do not propagate to "unexplained symptoms". At this point the number of symptom nodes still propagating to "unexplained symptoms" is noted. Comparing this to the number that propagated to "unexplained symptoms" before the disease was "chosen" gives us a measure of the disease's ability to explain the observed symptoms. The CHOSEN ISPEC is then removed from the disease node.

By doing this for each node that propagated to "possible diagnoses", we get a comparative measure of how the alternative diagnostic choices perform in explaining the observed symptoms. This measure of performance, along with the strength of belief which has propagated to the disease node are the criteria IRIS uses in choosing a diagnosis. Once the diagnosis node has been chosen, it is again given a CHOSEN ISPEC (permanently, this time). If there are still "unexplained symptoms", the process then repeats. At the end, some set of disease nodes will have been chosen, and each of these will have propagated an ISPEC to "chosen diagnoses".

The possible treatments for a disease are represented by nodes which are linked to the disease node by "treatmentfor" links. Propagation to the treatment nodes along these links takes place only when the disease node is chosen. Any treatment node which has accumulated a sufficiently high strength of belief in this way will propagate to "possible treatments" just as diseases do to "possible diagnoses".

IRIS decides on treatments using basically the same choice algorithm it uses for diagnostic decisions. Each treatment node which has propagated to "possible treatments" is in turn given a CHOSEN ISPEC. This causes TREATEDBY ISPECs to propagate from the treatment node to all nodes representing conditions this treatment can affect. They, upon receiving such ISPECs, cease propagating to "untreated pathology". From this process we get a measure of the alternative treatments' suitability for the patient. We then can choose one which gets a CHOSEN ISPEC permanently. Treatment nodes with CHOSEN ISPECs propagate to "chosen treatments".

We will illustrate this process in operation using a simplified (from a medical standpoint) example. In figure 6.1, the nodes "angle closure", "increased intraocular pressure", and "visual field loss" have received ISPECs as a result of user-supplied information. As a result, propagation (solid arrows) takes place from the three symptom nodes to "angle closure glaucoma", "unexplained symptoms", and "untreated pathology". Propagation then takes place from "angle closure glaucoma" to "possible diagnoses". In figure 6.2, "angle closure glaucoma" has received a CHOSEN ISPEC. Propagation to "possible diagnoses" ceases (broken arrow), and propagation now takes place to "chosen diagnoses" and "surgery". COVEREDBY ISPECs propagate from "angle closure glaucoma" to each of the three symptom nodes, "surgery" propagates to "possible treatments", and propagation from the three symptom nodes to "unexplained symptoms" ceases (a result of the COVEREDBY ISPECs). In figure 6.3, "surgery" is given a CHOSEN ISPEC, causing cessation of propagation to "possible treatments", propagation to "chosen treatments", and propagation of TREATEDBY ISPECs to the three symptom nodes. Propagation to "untreated pathology" from the three symptom nodes then ceases (caused by the TREATEDBY ISPECs).

This approach to clinical strategy enables us to restrict the points of contact between the semantic net/decision table knowledge base and the program that uses it. Only the six nodes mentioned at the beginning of this section are known to the program. IRIS's clinical strategy can be summarized as: 1) give CHOSEN ISPECs to nodes that propagate to "possible diagnoses" in such a way as to minimize propagation to "unexplained symptoms"; 2) give CHOSEN ISPECs to nodes that propagate to "possible treatments" in such a way as to minimize propagation to "untreated pathology"; 3) at the end of a session, produce output based on the ISPECs of nodes that are currently propagating to "chosen diagnoses" and "chosen treatments".

The advantage of such a restricted area of contact between the program and its knowledge base is that the knowledge base can be extensively altered without requiring changes in the program. The program only requires of its knowledge base that the six nodes be present, and that giving CHOSEN ISPECs to disease and treatment nodes will have effects on propagation to "unexplained symptoms" and "untreated pathology". The exact mechanism of these effects is unknown to and irrelevant to the program. It need not be done via COVEREDBY and TREATEDBY ISPECs.

This research was supported in part by grant RR-243 of the NIH Biotechnology Resources Program.

Brown, J.S., Bell A.G., Burton R., sophisticated instructional environment XPX TEACHING Electroninc Troubleshooting, Report AFHRL-TR-74-77, Technical Training Division, Lowry AFB, Colorado, 1974.

Kulikowski, C.A., Safir, A., Trigoboff, M., and Weiss, S.M., "Clinical Consultation and the Representation of Disease Processes: Some Artificial Intelligence Approaches", in Proceedings of the 1976 AISB Conference, Edinburgh, Scotland, 1976.

Kulikowski, C.A. and Weiss S.M., Computer Based Models Report CBM-TR-3, Department of Computer Science, Rutgers U., 1971.

Pauker, *S.G.,* Gorry, G.A., Kassirer, J.P., and Schwartz, W.B., "Towards the Simulation of Clinical Cognition", The American Journal of Medicine, 60:981, June 1976.

Pople, H.E. Jr., Myers, J.D., and Miller, R.A., "DIALOG: A Model of Diagnostic Logic for Internal Medicine", in Proc. 4th IJCAI, 2:849, 1975.

Rubin, A.D., Hypothesis Formation and Evaluauoji In Medical Diagnosis, MIT Artificial Intelligence Laboratory Technical Report AI-TR-316, January 1975.

Shortliffe, E.H., MYCIN - A Rule-Based Computer Pftram. for Advising Physicians on Antimicrobial Therapy Selection Memo 251, Computer Science Department, Stanford U., 1974.

Trigoboff, M., Ph.D. thesis, in preparation, 1977.

Trigoboff, M., "Propagation of Information in a Semantic Net", in Proceedings of the 1976 AISB Conference, Edinburgh, Scotland, 1976.

Weiss, S.M., A System for Model, I-Based Computr Aided Diagnosis and Therapy, Report CBM-TR-27, Department of Computer Science, Rutgers U., 1974.

Woods, W.A, "What's- in Link: Foundations for Semantic Networks", in Bobrow, D. and Collins, A. (eds), Representation and.. Understanding, Academic Press, New York, 1975.

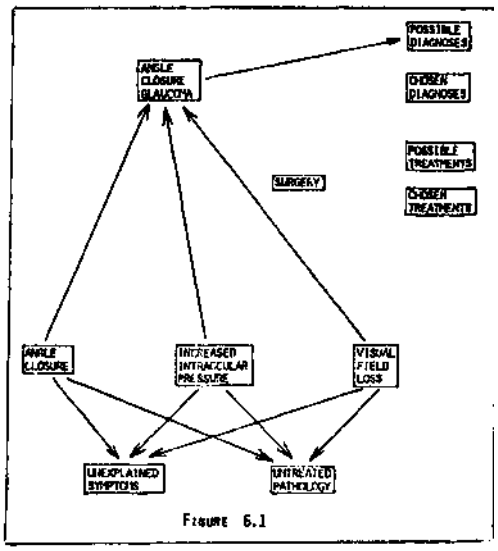


FIGURE 6.1

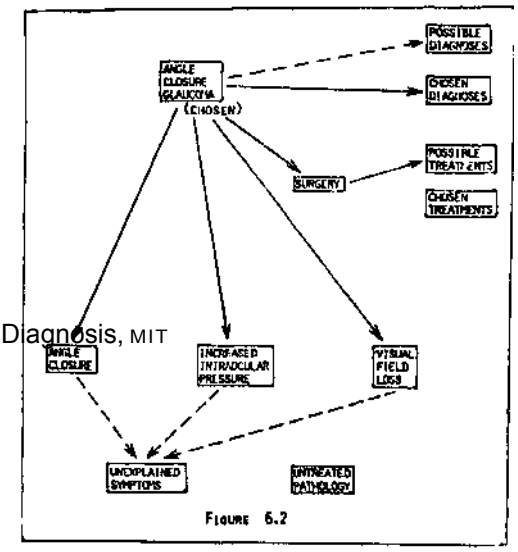


FIGURE 6.2

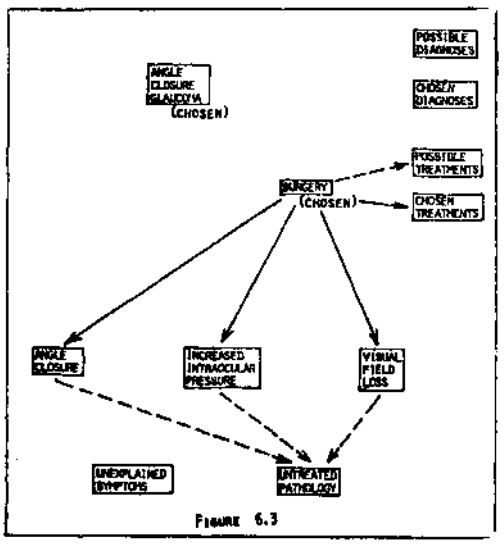


FIGURE 6.3