SOME RELATIONS BETWEEN PREDICATE CALCULUS* AND SEMANTIC NET REPRESENTATIONS OF DISCOURSE

TECHNICAL REPORT NO. NL-2

R. F. Simmons
and
B. C. Bruce

June 1971

to appear in Proceedings 2nd

INT. CONF. ON ARTIFICIAL

INTELLIGENCE, LONDON Sept. 1971

Supported By:

THE NATIONAL SCIENCE FOUNDATION,
Grant GJ 509X

Department of Computer Sciences

The University of Texas

Austin, Texas

SOME RELATIONS BETWEEN PREDICATE CALCULUS* AND SEMANTIC NET REPRESENTATIONS OF DISCOURSE

Robert F. Simmons and
Bertram C. Bruce

Department of Computer Sciences
The University of Texas
Austin, Texas
U.S.A.

ABSTRACT

Networks can be used to represent syntactic trees of the semantic relations that hold between words in sentences. They can be alternately symbolized as association lists or conjoined sets of triples. A semantic net represents a sentence as a conjoined set of binary predicates. An algorithm is presented that converts a semantic network into predicate calculus formalism. The simpler syntax of semantic network representations in contrast to ordinary predicate logic conventions is taken as an argument for their use in computational applications.

Descriptive Terms: Semantic networks, Predicate logic, Natural language, Computational linguistics, Association lists.

I. INTRODUCTION

In approaches to natural language question answering, it is generally agreed that question and text are to be transformed to some formal language representation. After this has been accomplished, answering a question phrased in a formal language from a data base represented in the same formal language is a process of theorem proving where the data are taken as axioms and the questions as theorems to be proved.

At this point two approaches are commonly found in the literature. One represents question and data in the syntactic conventions of the predicate calculus and uses standard theorem proving techniques such as Robinson's resolution algorithm supported by heuristic selection of relevant axioms (see Green & Raphael (4), Sandewall (9), Darlington (2)). The other represents question and text as attribute value lists or semantic nets (which will shortly be shown to be equivalent) and uses a matching algorithm guided by heuristic choices of relevant data. This approach is seen in Quillian (6), Raphael (7), Colby et al (1), Schwarcz et al (11).

In the recent literature Sandewall (10) and Palme (5) have each presented more or less formal developments of semantic network representations as predicate logics. Thompson (13) shows the similarity of a linguistic case-structure analysis to predicate calculus statements.

From yet another point of view, this paper informally shows the similarity of semantic networks to predicate calculus representations and argues that the semantic net syntax is computationally simpler and therefore to be preferred.

*This research was supported by: The National Science Foundation, Grant GJ 509 X

II. THE CORRESPONDENCE AMONG TREE, NETWORK AND ATTRIBUTE VALUE REPRESENTATIONS OF DISCOURSE

Linguists are accustomed to representing the structure of natural language sentences as trees. The following sentence E1 could be represented as in Figure 1.

E1) John made chairs with tools on October 20th in Austin.

For computational convenience Figure 1 can be represented also as an attribute-value list as in Table 1.

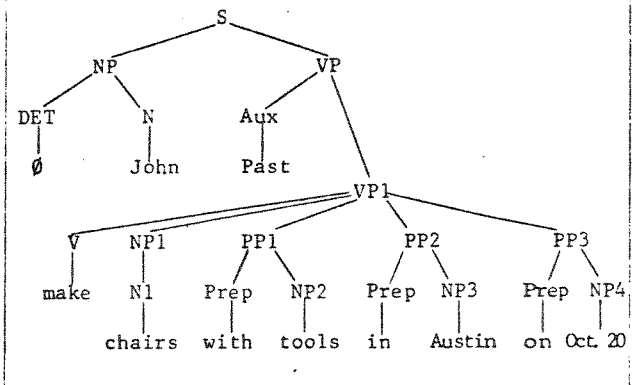


Figure 1. Syntactic Tree Representation of E1

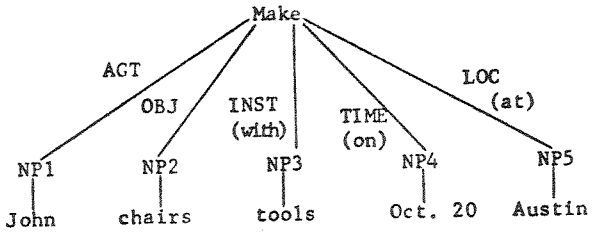
S	1st*	NP1	NP1	1st	N1
	2nd	VP	N1	Val	chairs
NP1	1st	DET	PP1	1st	Prep1
	2nd	N		2nd	NP2
DET	Val	Ø	Prep1	Val	with
N	Val	John	NP2	Val	tools
VP	1st	Aux	PP2	1st	Prep2
	2nd	VP1		2nd	NP3
Aux	Val	PAST	Prep2	Val	in
VP1	1st	V	NP3	Val	Austin
	2nd	NP1	PP3	1st	Prep3
	3rd	PP1		2nd	NP4
	4th	PP2	Prep3	Val	on
	5th	PP3	NP4	Val	Oct. 20
V	Val	Make			

*Note: 1st, 2nd, etc. is an arbitrary notation for successive branches from a node.

Table 1. Attribute-Value Representation of Syntactic Structure of E1

Another linguistic representation for E1 suggested by Fillmore (3) is shown in Figure 2. An attribute-value representation of this structure is also shown there. If the node values "John", "chairs", "tools", etc. are taken as symbols with unambiguous denotation, and AGT, OBJ, etc. as semantic relations, then Figure 2 is a seman-

tic network. The implied definition is that a semantic network is a system of unambiguous symbols interconnected by definable semantic relations.



Make	AGT	NP1
	OBJ	NP2
	INST (with)	NP3
	TIME (on)	NP4
	LOC (at)	NP5
NP1	TOK	John
NP2	TOK	chairs
NP3	TOK	tools
NP4	TOK	Oct. 20
NP5	TOK	Austin

Figure 2. Case structure of E1 and its Attribute-Value Representation

Looking at the attribute-value representation of Figure 2, we can see that it can also be represented as a conjoined set of triples as follows:

((make AGT NP1)
(make OBJ NP2)
(make INST NP3)
(etc.))

This is a simple and convenient representation for computational purposes. Its logic is explained later.

From this discussion it can be seen that the "node-arc-node" structure of networks is a syntax of symbolic notation that can equally well represent a syntactic tree structure or the semantic relations among elements of a sentence.* In either case, the net can be symbolized as an attribute-value list where the attributes are relations and the values are whatever is represented on the nodes. The attribute-value list, in turn, can be represented as a set of triples where each value may be another set of triples.

Before proceeding into the next section considering the logical notation for this structure, it is worth mention that Williams (14) developed a network representation for predicate calculus statements following original graphic notations she attributes to Frege. We take this as support of our eventual conclusion that semantic nets can be a fully valid representation of the underlying logic of a discourse where the nodes are unambiguous in their denotation and the relational arcs are fully defined.

*In Simmons & Slocum (12) the representation is developed for multi-sentence discourse and a network is formally defined.

III. SOME LOGICAL ASPECTS OF SEMANTIC NETWORKS

The principal value of using semantic nets for the structure underlying natural language sentences is that they are closer to both the form and the meaning of natural language than other proposed structures, such as first order predicate calculus. In this and the following sections we show that simple semantic nets can be mapped directly into ordinary first order predicate calculus, yet they have several computational advantages because of their proximity to natural language.

One particularly illuminating interpretation of semantic nets is that which considers each node as the name of a set of processes and each relational arc as a restriction on the sets named by the nodes it connects. Thus the set of all "makings" includes all events or processes in which an agent A makes an object B, with an instrument C, on time D, at location E. A, B, C, D and E must satisfy certain restrictions based on their participation in deep case relations, to the verb "make". For example, A, as an agent, must be an animate instigator. Similarly "make" itself is restricted to that subclass of all verbs which have agent, object, instrument, time and locative cases.

In a specific sentence the set of "makings" may be restricted further. For example in sentence E1 we have the small subclass of "makings":

"makings by John, of chairs, with tools, on Oct. 20, at Austin"

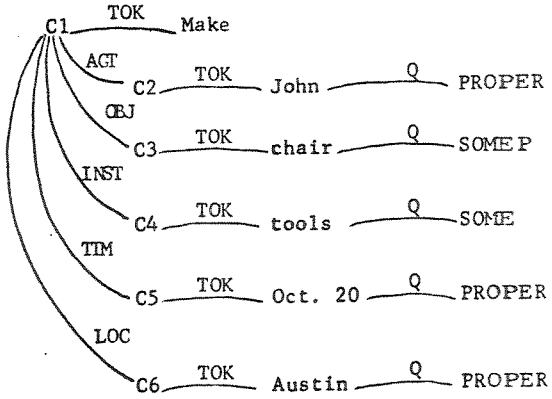


Figure 3. A Semantic Network

Referring to Figure 3 we see that each node in the semantic network is also restricted by its relations to other nodes. Thus C3 refers to a subset of processes which is contained in the set called "chairs" and in addition serves in the object relation to a certain semantically

restricted subset of "makings" represented by node C1.

In this scheme the relations AGT, OBJ, TIM and LOC are defined in two ways. The first is in terms of their logical import; i.e. (A AGT B) where A is a token of make and B of John, indicates that "John" is an argument of the predicate "make". The second is by the restrictions they impose on the nodes they connect. It is these restrictions as well as the interconnections of nodes which determine the semantically restricted sets.

A consequence of this viewpoint is that the statement "John made chairs with tools..." can be tested for truth value with reference to a data base (or model) by successive intersections of members from the set of binary predicates that represent it, with the set of binary predicates that make up the data base. To produce such a data base, every sentence describing an instance of a "making" is cross-referenced by its semantic relations of AGT, OBJ, INST, etc. to particular specifiers, e.g., "John", "tools", "chairs", etc. thus implying the set restrictions that have been described. And each nominal participant in a "making" is cross-referenced by -AGT, -OBJ, etc. to its token of the verb.

In the simplest case such successive intersections are performed by a simple matching algorithm. To answer a question simply find the set of triples matching each triple in the question. Methods similar to this are used in many question answering programs. A more sophisticated algorithm could incorporate various axioms for set theory and heuristics based on such information as the size of the sets being intersected or the list of semantic relations which determine the sets.

It appears that most questions can be answered efficiently by the recursive use of a simple matching algorithm. More complicated questions may require extensive use of the concept of set restrictions and intersections. Semantic nets are well suited for this kind of question answering, but they have the additional virtue of being easily transformed into more familiar logics such as first order predicate calculus. Following a more formal discussion of semantic nets and the representation of quantifiers we give an algorithm for converting simple nets into the predicate calculus.

A semantic net is a set of triples, (A R B), where A and B are nodes and R is a semantic relation. Nodes must be elements of a set of unambiguous symbols and semantic relations may be any of several defined or definable relations falling into one of the following categories:

1. Connectives such as OR, NOT, SINCE, etc.
2. Deep case relations such as AGT, OBJ, DAT, etc.
3. Attributive relations such as MOD, POSSESSIVE, HASPART, ASSOC, etc.
4. "TOK" meaning "is a token of"
5. "Q" meaning "is quantified"
6. Set relations such as SUP, SUB, EQUAL, etc.

Some of these relations are discussed elsewhere (see (12)).

Because of the special significance of the relation Q and its consequences for the translation algorithm, some discussion is warranted here. For the purposes of the algorithm which is to follow we give, in Table 2, a preliminary analysis for some of the kinds of quantification which are needed. Note that such quantifiers as "many", "most of the" and "almost all" are omitted from the table although they might be used in semantic nets.

Semantic Net Quantifier	English Language Equivalent	First Order Predicate Calculus Formula[1]
SOME	some, a	$(\exists x)(P(x)\wedge F)$
ALL	any, every, all	$(\forall x)(P(x)\supset F)$
NO	no, none of the, not any	$\{\sim(\exists x)(P(x)\wedge F)\}$ [2] $\{(\forall x)\sim(P(x)\wedge F)\}$
NOT-ALL	not all, not every	$\{(\exists x)\sim(P(x)\supset F)\}$ [2] $\{\sim(\forall x)(P(x)\supset F)\}$
SOMEP	some _____s	$(\exists x_1)(\forall x_2)(\mathrm{card}(x_1)>1 \wedge (x_2\in x_1\supset P(x_2))\wedge F)$
0,1,...,n	0,1,...,n	$(\exists x_1)(\forall x_2)(\mathrm{card}(x_2)=n \wedge (x_2\in x_1\supset P(x_2))\wedge F)$
PROPER	proper noun, example: John	$(\exists x)(P(x)\wedge F)$
SOME-INDEF	some fish	1. $(\exists x)(P(x)\wedge F)$ [3]
		2. $(\exists x_1)(\forall x_2)((\mathrm{card}(x_1)>1\wedge (x_2\in x_1\supset P(x_2)))\supset F)$
		3. $(\exists x_1)(\exists x_2)(P(x_1)\wedge \mathrm{PARTOF}(x_2,x_1)\wedge F)$
		4. $(\exists x_1)(\forall x_2)(\exists x_3)(\exists x_4)(\mathrm{card}(x_1)>1\wedge(x_2\in x_1)\supset(P(x_2)\wedge \mathrm{PARTOF}(x_3x_2)\wedge x_3\in x_4))\wedge F)$

Notes:
1. In the predicate calculus formula, above, F represents the portion of the formula which does not contain x as a free variable, while P(x) is the portion which does.
2. Equivalent predicate calculus expressions.
3. Alternate logical meanings of "SOME-INDEF".

Table 2. Examples of Quantifiers in Semantic Nets

We do not need a complete logical definition for a concept in order to use it in the semantic net. As an example of the complexity which may be avoided, consider the phrase "some fish". This may mean some one fish, several fish, or parts of one or more fish. In a semantic net we use the quantifier "SOME-INDEF" which allows any of these meanings. In predicate calculus we would be forced to select one of the formulas given in Table 2 or perhaps a complicated disjunction of all the possible formulas. The con-

sequence for question answering of such vagueness in the semantic net is of course a reduced precision in the correspondence between question and answer. Thus answering the question

"Did John eat a fish?"

from

"John ate some fish."

leads to an answer of "maybe" reflecting the imprecision of the match.

IV. THE TRANSLATION ALGORITHM

The algorithm given here is designed to convert one representation of semantic nets to predicate calculus. Several restrictions are made to simplify the presentation. No higher order predications are handled by the algorithm (such as would normally be required to handle "he moves slowly"). It is assumed that connectives can be treated like verbs, while, in fact, certain connectives may require complex translation algorithms in themselves. Also, relative clauses are not allowed. In sum, we are dealing with a subset of semantic nets which could not serve as a semantic structure for natural language.

Nevertheless, the algorithm does show that simple semantic nets have a sound logical structure, i.e., first order predicate calculus. An algorithm for more complex semantic nets appears to be dependent only upon our understanding of the concepts we wish to allow.

The algorithm handles, in turn, connectives, verbs, nouns, and noun modifiers. It uses pushdown stacks in the customary way. We start with pushdown stacks for the variables G, J, J', F and F', all empty. Capital letter variables (A) denote specific formulas, nodes or arcs under consideration, while small letter variables (a) indicate any node or arc.

Algorithm for converting semantic nets to predicate calculus:

1. Start with a conjoined list of triples representing the semantic net of a discourse. Call this list, G. Go to step 2.
2. Set J to nil. Go to step 3.
3. If there are any triples in G of the form (A TOK B) such that B is a connective, and such that there are no triples (c r A) where (c TOK d) is in G and d is a connective, then select the first such triple, call it H, and go to step 4. Otherwise go to step 12.
4. For each connective (or verb) there is a prescribed ordering for its arguments. For example, the verb "give" has the ordering AGT, OBJ, DAT. This ordering, specified in the lexicon, is called the "case argument description for the connective (or verb)". Collect and order (by case argument description) all triples of the form (A r c) such that A is the first member of H. Form an n-tuple (B c_1 c_2...c_{n-1}) such that B is the third member of H and the triple (A r_i c_i is the ith triple in the prescribed ordering. Call this n-tuple, J. Set J' equal to J with the first member removed. Remove from G and all values of G on its stack all triples whose first member is A. Go to step 5.
5. If J' has no members then set G=G$\wedge$J and go to step 6. Otherwise go to step 7.
6. If the pushdown stack for G is empty then go to step 2. Otherwise set J and J' to equal the top members of their respective stacks. Substitute G, in J, for the first element of J'. Remove the first member of J'. Set G to equal the top member of its pushdown stack. Remove the top member of the stacks G, J and J'. Go to step 5.
7. If there is a triple in G of the form (a TOK b) such that a is the first element of J' and b is a verb then call this triple K and go to step 8. Otherwise select the triple H=(a TOK b) such that a is the first element of J' and b is a connective. Push G, J and J' onto their respective stacks. Set G to nil and go to step 4.
8. Paralleling the procedure used for connectives in step 4, form an n-tuple (B c_1 c_2 ...c_{n-1}) such that B is the third member of K and c_i is the third member of the ith triple in the ordering prescribed by the case argument description of the verb. Call this n-tuple, F. Set F' equal to F with its first member removed. Remove from G and all values of G on its stack every triple whose first member is the first member of K. Go to step 9.
9. If F' has no members then go to step 11. Otherwise go to step 10.
10. Call the last element of F', D. Form a conjoined list, call it P', of all triples in G of the form (D r a) except the triples for which r=TOK or r=Q. Convert the triples in P' to prefix notation, (r D a). Set P=P'$\wedge$(E D) where (D TOK E) is in G. Locate the form specified for the X of (D Q X) in the table of quantifiers (Table 2). Let the value of D be the bound variable in that form and substitute P and F as they have been constructed. Call the result, F. Remove all triples whose first member is D, from G. Remove D from F'. Go to step 9.
11. If J is nil, set G=G$\wedge$F and go to step 12. Otherwise substitute F, in J, for the first element of J'. Remove the first member of J' and go to step 5.
12. If there are any triples in G of the form (A TOK b) such that b is a verb then select the first such triple, call it K, and go to step 8. Otherwise, stop.

As an example application of the algorithm we take the sentence:

"The old man give a book to John."

The numbering below corresponds to steps in the algorithm. The value of each variable is written out when it changes.

1. G = (C1 TOK GIVE)$_\wedge$(C1 AGT C2)$_\wedge$(C1 OBJ C3) $_\wedge$(C1 DAT C4)$_\wedge$(C2 TOK MAN)$_\wedge$(C2 MOD OLD) $_\wedge$(C2 Q SOME)$_\wedge$(C3 TOK BOOK)$_\wedge$(C3 Q SOME) $_\wedge$(C4 TOK JOHN)$_\wedge$(C4 Q PROPER)

2. J = nil

3. no connectives

12. K = (C1 TOK GIVE)

8. B = GIVE
   F = (GIVE C2 C3 C4)
   F' = (C2 C3 C4)
   G = (C2 TOK MAN)$_\wedge$(C2 MOD OLD)$_\wedge$(C2 Q SOME) $_\wedge$(C3 TOK BOOK)$_\wedge$(C4 TOK JOHN) $_\wedge$(C4 Q PROPER)$_\wedge$(C3 Q SOME)

9. F' is non-empty

10. D = C4
    P' = ( )
    P = (JOHN C4)
    F = ($\exists$C4)((JOHN C4)$_\wedge$(GIVE C2 C3 C4))
    G = (C2 TOK MAN)$_\wedge$(C2 MOD OLD)$_\wedge$(C2 Q SOME) $_\wedge$(C3 TOK BOOK)$_\wedge$(C3 Q SOME)
    F' = (C2 C3)

9. F' is non-empty

10. D = C3
    P' = ( )
    P = (BOOK C3)
    F = ($\exists$C3)((BOOK C3)$_\wedge$($\exists$C4)((JOHN C4) $_\wedge$(GIVE C2 C3 C4)))
    G = (C2 TOK MAN)$_\wedge$(C2 MOD OLD)$_\wedge$(C2 Q SOME)
    F' = (C2)

9. F' is non-empty

10. D = C2
    P' = (MOD C2 OLD)
    P = (MOD C2 OLD)$_\wedge$(MAN C2)
    F = ($\exists$C2)((MAN C2)$_\wedge$(MOD C2 OLD)$_\wedge$($\exists$C3) ((BOOK C3)$_\wedge$($\exists$C4)((JOHN C4) $_\wedge$(GIVE C2 C3 C4))))
    G = ( )
    F' = ( )

9. F' is empty

11. G = ($\exists$C2) ((MAN C2)$_\wedge$(MOD C2 OLD)$_\wedge$($\exists$C3) ((BOOK C3)$_\wedge$($\exists$C4)((JOHN C4) $_\wedge$(GIVE C2 C3 C4))))

12. stop

The final result of the translation algorithm is given in step 11. A simple transformation of the given formula can be made to conform to the particular conventions one wishes to follow, such as infix vs. prefix, the method for handling proper names, modifications, and other language features.

V. DISCUSSION AND CONCLUSIONS

We have mentioned some of the reasons for using semantic nets as the struction underlying natural language discourse. These may be summarized as follows:

1. Vague or partially undefined concepts may be used in a semantic net by giving rules governing their operation. Many questions will not require the full specification of functions used in predicate calculus. For example, the dialogue:

"The old man ate some fish."
"Did the old man eat some fish?"
"Yes."

does not presume a precise understanding of the meaning of "some", but we would expect a computer question answerer to be capable of it. The earlier similar example showed that a "maybe" answer is the consequence of vagueness in critical aspects of the meaning represented in the semantic net.

2. Simple questions will be answered by a simple matching procedure which is easily performed in a semantic net. More complex questions will take advantage of the set restriction principles of semantic nets. It is also possible to convert some parts of the net to first order predicate calculus and use some form of Robinson's (8) resolution algorithm.

3. A semantic net seems closer than predicate calculus to natural language form and meaning. For example the nesting of natural language expressions such as

"The man who caught the fish ate it."

is easily handled in semantic nets (see (12)).

Lest, however, we be too happy at this situation, it must be mentioned that there are alternate specifications for processes, such as "the 20th day of the 10th month" for "October 20" and "a lathe, a saw, a plane and a chisel" for "tools", or "furniture" for "chairs". Managing the possible logical, lexical and syntactic paraphrases of a statement in testing its truth value adds a significant transformational and implicational complexity that must be accommodated if the semantic nets are to be useful in language processing.

These complexities are provided for by additional relations in a lexicon and grammar. The lexicon shows, for example, that the process named "tools" includes as subsets the processes named "planes", "lathes", "hammers", "saws", etc.; that "chairs" are a subset of "furniture"; and that there is a transformation that maps "October 20" into "20th day of the 10th month" and the converse. If this additional information is present, it is possible but difficult, to discover the approximate paraphrastic equivalence of E1 to E3 in the following two examples:

E1) "John made chairs with tools on October 20."

E3) "John made furniture with a lathe, a saw, a plane and a chisel on the 20th day of the 10th month."

E1

```
1  ACT(make John)
2  OBJ(make X)
3  INST(make Y)
4  TIME(make Z)
5  TOK(X chairs)
6  TOK(Y tools)
7  TOK(Z Oct. 20)
8  SUP(X furniture)
9  SUB(Y,(lathe, saw, plane, etc.))
10 IMPLY(20th Oct., (ORDER(day 20)∧ORDER(month 10))
11 (TOK(X,Y)∧SUB(X,V)⊃TOK(X,V))
12 (TOK(X.Y)∧SUP(X,V)⊃TOK(X,V))
13 (TOK(X,Y)∧IMPLY(Y,V)⊃V)
```

Figure 4. Augmented Representation of Example E1

In Figure 4, by augmenting the (partial) semantic representation of E1 (lines 1-7), with the relevant lexical information of lines 8-10 and the inference rules of lines 11-13, it is possible to find a correspondence with the representation of E3. But the selection of the appropriate augmentation of E1 with respect to E3 introduces the need for heuristics to guide the process, and algorithms to accomplish the indicated transformations as a part of the comparison procedure.

In conclusion, this paper has shown that network formalisms can be used to represent syntactic structures of sentences or to represent the semantic relations that hold between word meanings in discourse. In the latter case the network is called a semantic network. Networks can be written as graphs, as attribute-value lists, or as lists of triples for the sake of computational convenience.

Semantic nets were defined formally and shown to have a set theoretic interpretation as a structure of relations that restricts the range of a set of events such as all "makings" to some small subset occurring at a particular place and time with a particular agent, object and instrument. An approach for handling quantification in the nets was described and shown to have some advantage over the predicate calculus for representing such vague expressions as "some fish". An algorithm was presented for translating from a subset of the network formalism to first order logic and there is no particular reason why algorithms for translating to higher order logics cannot be developed as the conventions are established for representing more complex meanings both in the networks and the higher order calculi.

The several representations of structure that have been used widely in question answering include triples, attribute-value lists, graphs and logical predicates. We have shown the relation of semantic networks to each of these formalisms to demonstrate the generality and logical adequacy of the semantic network approach to representing meanings of natural language discourse. These findings in conjunction with those of Sandewall (10) and Palme (5) support our belief that semantic network structures are a well-formed logic with computational advantages deriving from the similarity of their notational conventions to those of natural languages.

REFERENCES

1. COLBY, K.M., TESLER, L. and ENEA, H. "Experiments with a search algorithm for the data base of a human belief structure." Proc. Int. Jt. Conf. Art. Intel., Washington, D.C., 1969, pp. 649-654.

2. DARLINGTON, J. L. "Theorem proving and information retrieval." Machine Intelligence 4, Meltzer and Michie (eds.) Edinburgh U. Press, Edinburgh, 1969, Ch. 11.

3. FILLMORE, C.J. "The Case for Case." In Bach, E. and Harms, R.T., Universals in Linguistic Theory. Holt, Rinehart and Winston, Inc. Chicago, 1968.

4. GREEN, C.C. and RAPHAEL, B. "Research on Intelligent Question-Answering Systems." Proc. ACM 23rd Nat. Conf., 1968, Brandon Systems Press, Princeton, N.J., pp. 169-181.

5. PALME, JACOB. "Making Computer Understand Natural Language." Research Inst. of Nat'l Def., Op. Res. Ctr. A-10450, Stockholm 80, Sweden.

6. QUILLIAN, M.R. "The Teachable Language Comprehender." Comm. ACM 12, August 1969, pp. 459-475.

7. RAPHAEL, B. "SIR: A Computer Program for Semantic Information Retrieval." Ph.D. Th., Math. Dept., MIT, Cambridge, Mass., June 1964. In Proc. AFIPS 1964 Fall Joint Comput Conf., Vol. 26, Pt. 1, Spartan Books, New York, pp. 577-589.

8. ROBINSON, J.A. "A Machine Oriented Logic based on the Resolution Principle." JACM 12 January 1965, pp. 23-41.

9. SANDEWALL, E.J. "Concepts and Methods for Heuristic Search." Proc. Int. Jt. Conf. Art. Intel., Washington, D.C., 1969.

10. SANDEWALL, E.J. "Representing Natural-Language Information in Predicate Calculus." Stanford Univer. Comp. Sci. Dept. Report #166, Palo Alto, July, 1970.

11. SCHWARCZ, R.M., BURGER, J.F. and SIMMONS, R.F. "A Deductive Logic for Answering English Questions." Comm. ACM 13, March 1970, pp. 167-183.

12. SIMMONS, R.F. and SLOCUM, J. "Generating English Discourse From Semantic Networks."

Natural Language Research for Computer-Assisted Instruction, Tech. Report NL-3. University of Texas, Austin, November, 1970

13. THOMPSON, SANDRA A. "On Relative Clause Structure in Relation to the Nature of Sentence Complexity." Mss. from the author, UCLA Linguistics Dept., Fall 1969.

14. WILLIAMS, THYLLIS. "Case Variables and Case Description in a Reticular Logic Grammar." Preprint, Graduate Library School, Univer. of Chicago, Chicago, Ill., September, 1966.