# Using Non-Standard Connectives and Quantifiers for Representing Deduction Rules in a Semantic Network

Stuart C. Shapiro
Department of Computer Science
State University of New York at Buffalo
Amherst, New York 14226

## 1. Introduction

There have been several descriptions of logically adequate formalisms for representing quantified deduction rules in semantic networks [6,9,10,11,12,14,15,16,17,18,19,20,23]. Most of these have involved fairly straight-forward translations of the standard syntax of predicate calculus, especially with regard to the use of the common connectives, conjunction, disjunction, negation, and implication. We have been pursuing a program of investigating non-standard connectives, quantifiers and logics for use in the inference component of semantic network based understanding and question answering systems [1,2,19,20,21,22].

In this paper, we describe the set of connectives and quantifiers we are currently using, review the motivations for them and show how they are represented in a semantic network. First, in Section 2, we review our basic semantic network terminology and notation. Section 3 motivates and introduces the connectives and quantifiers. Section 4 presents the representation and gives some examples. Section 5 is a status report on the implementation of these ideas.

---

Presented at: Current Aspects of AI Research, a seminar held at the Electrotechnical Laboratory, Tokyo, Aug. 27-28, 1979.

This material is based on work supported in part by the National Science Foundation under Grant #MCS78-02274.

## 2. Semantic Network Terminology and Notation

A semantic network is a labelled directed graph in which nodes represent concepts and arcs represent non-conceptual binary relations (called structural relations, system relations or arc relations) between concepts. Each concept is represented by a unique node. Whenever an arc representing a relation $R$ goes from node $n$ to node $m$, there is an arc representing the converse relation of $R$, $R^C$, going from $m$ to $n$. Each arc relation has a unique symbol which is used to label all arcs representing the relation. There are no cycles consisting entirely of arcs with the same labels, neither are there parallel arcs (connecting the same two nodes) with the same label.

In SNePS semantic networks [11], we distinguish three kinds of arcs: descending, ascending and auxiliary. For each relation represented by descending arcs, there is a converse relation represented by ascending arcs and vice versa. Together, descending and ascending arcs are the regular semantic network arcs referred to above. Auxiliary arcs are used for hanging non-nodal information on nodes and for typing the nodes as discussed below. If a descending arc goes from node $n$ to node $m$, we say that $n$ immediately dominates $m$. If there is a path of descending arcs from node $n$ to node $m$, we say that $n$ dominates $m$. There are no cycles consisting entirely of descending or of ascending arcs, so no node dominates itself. If $R$ is an arc label and $n$ is a node, we will use the notation $R(n)$ for the set of nodes into which arcs labeled $R$ go from $n$. In what follows, we will often use the phrase "the relation $R$" when we mean "an arc labeled $R$".

There are three kinds of nodes: constant, non-constant, and auxiliary. Auxiliary nodes are connected to each other and to other nodes only by auxiliary arcs. Constant nodes represent unique semantic concepts. Nodes which dominate no other node are called atomic nodes. Atomic constants are called base nodes and atomic non-constants are called variable nodes or variables. Variables are distinguished by being in the auxiliary relation :VAR to the auxiliary node T. Non-atomic nodes are called molecular nodes. There is a set of descending relations called binding relations. A molecular node that immediately dominates one or more variables and that dominates no other variable may have at most one binding relation to an arbitrary number of its dominated variables, which are referred to as bound by that molecular node. The remaining dominated variables are referred to as free in the molecular node, which has an auxiliary arc labeled :SVAR to each of them. If a node $m$ immediately dominates a set of variable nodes $\{v_1, \ldots, v_\ell\}$ and a set of molecular nodes $\{n_1, \ldots, n_k\}$ and $V = \{v_1, \ldots, v_\ell\} \cup :\mathrm{SVAR}(n_1) \cup \ldots \cup :\mathrm{SVAR}(n_k)$ is non-empty, $m$ may have at most one binding relation, say $Q$, to one or more variables in $V$. These variables are referred to as bound by $m$. The remainder, $V - Q(m)$, are free in $m$ and have the arc :SVAR to each of them from $m$. A node $n$ such that :SVAR($n$) is non-empty is a non-constant molecular node and is called a pattern node. A molecular node $n$ for which :SVAR($n$) is empty is a molecular constant or assertion node.

Temporary molecular and variable nodes can be created. Temporary molecular nodes have no ascending arcs coming into them from the

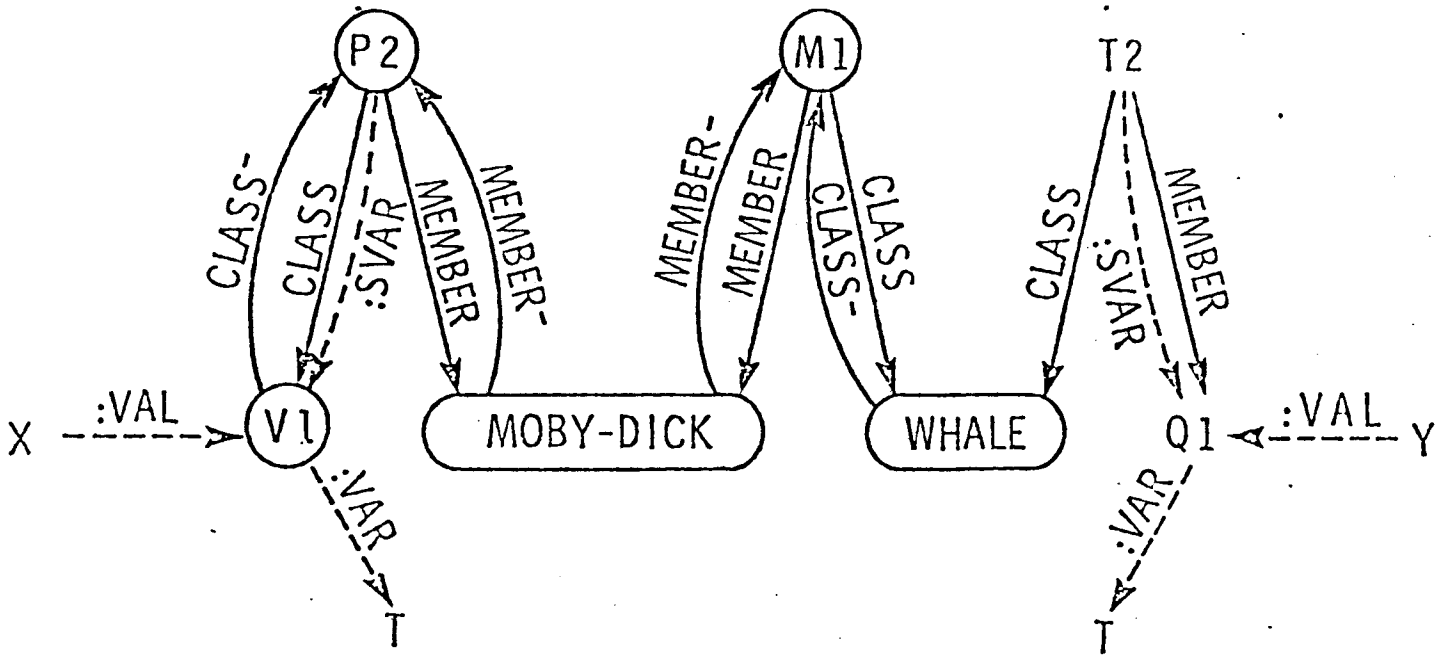


Figure 1: An example of various kinds of nodes and arcs.

| | |
|---|---|
| Descending arcs: | MEMBER, CLASS |
| Ascending arcs: | MEMBER-, CLASS- |
| Auxiliary arcs: | :VAL, :VAR, :SVAR |
| Base nodes: | MOBY-DICK, WHALE |
| Assertion node: | M1 |
| Variable node: | V1 |
| Pattern node: | P2 |
| Temporary variable: | Q1 |
| Temporary pattern: | T2 |
| Auxiliary nodes: | X,Y,T |

nodes they dominate. Temporary nodes are not placed on any permanent system list and are garbage-collected when no longer referenced. They are invisible to all the semantic network retrieval operations. We will refer to non-temporary nodes as permanent nodes.

In diagrams, a base node is drawn as an oval inside of which is an identifier meant to be suggestive of the concept it represents. A permanent assertion node is drawn as a circle inside of which is an arbitrary identifier of the form M*n*. A permanent variable node is drawn as a circle inside of which is an arbitrary identifier of the form V*n*. A permanent pattern node is drawn as a circle inside of which is an arbitrary identifier of the form P*n*. A temporary variable node is shown as an identifier of the form Q*n*, and a temporary molecular node is an identifier of the form T*n*. An auxiliary node is shown as a mnemonically suggestive identifier. Figure 1 shows a network with various kinds of nodes and arcs. In future figures, we will omit the :VAR and :SVAR arcs since they can be reconstructed from the information shown.

## 3. Non-Standard Connectives and Quantifiers

### 3.1. Motivations

In designing representations of reasoning rules one is faced with the decision of how atomic propositions or actions will be related to each other. Using connectives and quantifiers that have been studied as part of formal logical systems has the advantage that the inferential properties of these operators are clear, well known, and consistent.

This choice of logic based operators, however, need not restrict one to the standard set of connectives and quantifiers familiar to the introductory logic student. It is well known that various sets of propositional connectives are adequate in the sense that given only connectives from the set and $n$ propositional variables, it is possible to construct well formed formulas for each of the $2^{2^n}$ truth functions, for any $n$. Some adequate sets are: negation (~) plus any one of conjunction (&), disjunction (v) or implication (→); either of nor (↓) or nand (|). The actual set of connectives chosen for use in some logical system depends on factors such as ease of proving consistency and completeness, ease of proving equivalence to some other system, and ease of use for proving theorems within the system. Along with the choice of connectives come a choice of logical axioms and rules of inference. In general, if the number of connectives, axioms and rules of inference are minimized, it is easier to prove theorems about the logical system, but harder to construct proofs within the system than when a larger set is chosen.

The interests which have motivated our choice of connectives and quantifiers are in understanding natural language, representation of knowledge (specifically using semantic networks), and carrying out deductions within, rather than about, the representation formalism. Therefore, our concerns were with closeness to modes of human reasoning, structural simplicity, and expressibility rather than minimality.

The standard connectives (except for negation) are binary, but the use of binary connectives in a knowledge representation introduces needless structural complexity. Having to choose between (AvB)vC and Av(BvC) is unnecessary and unfairly distinguishes one

of the disjuncts. In a semantic network, we would have one node representing each disjunct, one node representing the disjunction, and an arc between the disjunction node and each disjunct node. Since the role played by each disjunct in the disjunction is the same, the same arc relation (we will use ARG) should be used from the disjunction node to each disjunct. Since ARG($n$), for any node $n$, is a set (unordered, with no duplicates), this representation is logically possible only because disjunction is associative, commutative and idempotent ($A \vee A \equiv A$). Since conjunction is also associative, commutative, and idempotent, it can be represented similarly.

Nor ($\downarrow$), nand ($|$), exclusive or ($\oplus$) and equivalence ($\equiv$), also being commutative binary connectives, would also seem to be candidates for being represented similary to disjunction and conjunction. Moreover, they should be represented directly because sentences which use them are not conceptually more difficult than sentences using conjunction or disjunction. In fact $\oplus$ has more right than $\vee$ to be represented directly since "or" in English almost always means $\oplus$ rather than $\vee$ (note the existence of the locution "and/or").

However, $\downarrow$, $|$, $\oplus$ and $\equiv$ cannot be extended to taking arbitrarily large sets of arguments as $\vee$ and $\&$ can, because $\downarrow$ and $|$ are not associative ($(true \downarrow false) \downarrow false \equiv true$, but $true \downarrow (false \downarrow false) \equiv false$; $(true | false) | false \equiv true$, but $true | (false | false) \equiv false$), and none of the four are idempotent ($A \downarrow A \equiv \sim A$; $A | A \equiv \sim A$; $A \oplus A \equiv false$; $(A \equiv A) \equiv true$). Also, though $\oplus$ and $\equiv$ are associative and commutative, when applied

to more than two arguments, they are not true, as we would want, when exactly one argument is true and when all arguments are equivalent, respectively, but when an odd number of arguments are true and when an odd number of arguments are false.

Connectives that take sets of propositions as arguments and that mean "none are true", "not all are true", "exactly one is true", "all are equivalent" would be useful, but, as we have seen, the standard connectives are not appropriate, so we must define new connectives for the purpose.

## 3.2 Non-Standard Connectives

A principle of good programming is to prefer a general solution over a collection of special purpose solutions, especially when the general solution is not much more complicated than any one of the special purpose solutions. Following this principle, we use a single, parameterized connective which generalizes &, ∨, ⊕, ↓ and | properly to the set oriented connectives we desire. We call the connective AND-OR and symbolize it as ${}_n\mathbb{W}_i^j$. This takes as arguments a set of $n$ propositions, the formula ${}_n\mathbb{W}_i^j(A_1,\ldots,A_n)$ being true just in case at least $i$ and at most $j$ of the arguments are true. That AND-OR is the generalization we claim may be noted by the following equivalences:

$$ {}_2\mathbb{W}_2^2(A,B) \equiv A\&B $$

$$ {}_2\mathbb{W}_1^2(A,B) \equiv A \vee B $$

$$ {}_2\mathbb{W}_1^1(A,B) \equiv A \oplus B $$

$$_2\mathbb{W}_0^0(A,B) \equiv A \downarrow B$$

$$_2\mathbb{W}_0^1(A,B) \equiv A|B$$

$_n\mathbb{W}_0^0(A_1,\ldots,A_n)$ means "none of $A_1,\ldots,A_n$ are true"

$_n\mathbb{W}_0^{n-1}(A_1,\ldots,A_n)$ means "not all of $A_1,\ldots,A_n$ are true"

$_n\mathbb{W}_1^1(A_1,\ldots,A_n)$ means "exactly one of $A_1,\ldots,A_n$ is true"

Moreover:

$$_1\mathbb{W}_1^1(A) \equiv A$$

$$_1\mathbb{W}_0^0(A) \equiv \sim A$$

From the above equivalences, it is clear that AND-OR is adequate in the sense mentioned above. However, using $_n\mathbb{W}_i^j$ to express $\equiv$ and $\rightarrow$ requires nesting, and we prefer structural simplicity to minimality, so we use additional connectives. The connective we call THRESH (symbolized $_n\theta_i$) generalizes $\equiv$ to a set of arguments. The formula $_n\theta_i(A_1,\ldots,A_n)$ is true if either fewer than $i$ of the arguments are true or they all are. Note that $_2\theta_1(A,B)$ is equivalent to $A \equiv B$, and that $_n\theta_1(A_1,\ldots,A_n)$ means that the arguments are either all true or all false.

Since $\rightarrow$ is not symmetric, we cannot generalize it to a connective that takes a single set of arguments, but we can generalize it to one that takes two sets, a set of antecedents and a set of

consequents. We have used two versions, or-entailment ($\vee\!\!\rightarrow$) and and-entailment ($\&\!\!\rightarrow$). The formula $(A_1,\ldots,A_n)\vee\!\!\rightarrow(C_1,\ldots,C_m)$ means that $A_i\rightarrow C_j$, $1\leq i\leq n$, $1\leq j\leq m$, or the disjunction of the antecedents implies the conjunction of the consequents. The formula $(A_1,\ldots A_m)\&\!\!\rightarrow(C_1,\ldots,C_m)$ means that the conjunction of the antecedents implies the conjunction of the consequents. These two entailments may be combined into one. The formula $(A_1,\ldots,A_n)\overset{i}{\Leftarrow}(C_1,\ldots,C_m)$ means that if any $i$ of the antecedents are true, so are all of the consequents. Thus, ${}_n\Theta_i(A_1,\ldots,A_n)$ is equivalent to $(A_1,\ldots,A_n)\overset{i}{\Leftarrow}(A_1,\ldots,A_n)$.

The above non-standard connectives are all truth functions, and a set of axioms and rules of inference may be devised for them to yield a logic equivalent to any standard propositional calculus. There is another useful connective that is not truth functional and whose introduction brings us to a logic with a strange property. It is the non-derivable operator $\nvdash$ [8,13] (the THNOT of PLANNER). The formula $\nvdash A$ means that A is not derivable in the current data base. The strange property $\nvdash$ introduces is that of non-extensionality. The extension principle is that if a formula A is derivable from a set of formulae $\Phi$, it is also derivable from any superset of $\Phi$. However, A is derivable from $\{(\nvdash B)\rightarrow A\}$, but not from $\{(\nvdash B)\rightarrow A,\ B\}$. Rules such as $(\nvdash B)\rightarrow A$ would never be used in a standard logical system or theorem prover, but they are often useful in common sense reasoning programs. Since B might be asserted after $(\nvdash B)\rightarrow A$ is used, plans have to be made to recognize and recover from such contradictions. Doyle's "truth maintenance" mechanism [4,5] is one possible approach.

One common use of $\nvdash$ is in default reasoning: $(\nvdash \sim A) \rightarrow A$, "assume A is true unless it is provably false." Because of the frequency of such rules in common sense reasoning, we use a default operator, $\Delta$, the schema $\Delta A$ being an abbreviation of $(\nvdash \sim A) \rightarrow A$, but having a structural simplicity commensurate with its common use.

The non-standard connectives we have discussed, ${}_n\Psi^j_i$, ${}_n\Theta_i$, $\overset{\vee}{\rightarrow}$, $\overset{i}{\leftarrow}$, $\overset{i}{\rightarrow}$, $\nvdash$, and $\Delta$ have the properties that they represent with minimal complexity operators used in common sense reasoning, or are straight-forward generalizations of such operators, that their semantics and inferential properties are clear, and that they are adequate for representing any propositional formula from any standard propositional calculus.

## 3.3 Quantifiers

As with connectives, we want a set of quantifiers that are adequate, that capture or generalize modes of common sense reasoning without undue representational complexity, and that have clear semantics. For adequacy, we use the universal ($\forall$) and existential ($\exists$) quantifiers. These are generalized to bind sets of variables instead of single variables, e.g. $\forall(x,y)P(x,y)$ and $\exists(x,y)P(x,y)$ instead of $\forall x \forall y P(x,y)$ and $\exists x \exists y P(x,y)$. This is proper since the order of successive quantifiers of the same type is logically irrelevant ($\forall x \forall y P(x,y) \equiv \forall y \forall x P(x,y)$ and $\exists x \exists y P(x,y) \equiv \exists y \exists x P(x,y)$).

Simple numerical quantifiers are very useful in common sense reasoning, e.g. "Every elephant has exactly one trunk," "Every animal has exactly two parents." We have generalized this to the general numerical quantifier schema

${}_n\exists^j_i\bar{x}(P_1(\bar{x}),\ldots,P_k(\bar{x}):Q(\bar{x}))$, where $\bar{x}$ is a sequence of variables, [21]. The meaning of this is that there are $n$ sequences of constants each of which when substituted for $\bar{x}$ will make $P_1(\bar{x})\&\ldots\&P_k(\bar{x})$ true, and that at least $i$ and at most $j$ of these will also satisfy $Q(\bar{x})$. Just as $\forall xP(x)$ is equivalent to $P(a_1)\&P(a_2)\&\ldots$, for all constants $a_i$ in the universe, and similarly $\exists xP(x)$ is equivalent to $P(a_1)\vee P(a_2)\vee\ldots$, so is ${}_n\exists^j_i(x)P(x)$ equivalent to ${}_n\exists^j_i(P(a_1),\ldots,P(a_n))$. Also, ${}_n\exists^j_i$ generalizes $\forall$ and $\exists$ just as ${}_n\exists^j_i$ generalizes $\&$ and $\vee$, ${}_n\exists^n_n$ being equivalent to $\forall$ and ${}_n\exists^n_1$ being equivalent to $\exists$. The numerical quantifier, however, is used differently than the universal and existential quantifiers. The universal quantifier is used for storing propositions which are true for all constants in some domain. The existential quantifier is used to justify discussing individuals who have not been explicitly introduced ("Everyone has a mother" justifies discussing John's mother). Numerical quantifiers are used for reasoning by exclusion--if everyone has exactly one mother, then Mary can't be John's mother if it is already known that Jane is John's mother. Because of the different uses, we retain $\forall$ and $\exists$ even after introducing ${}_n\exists^j_i$.

## 4. Representation of Connectives and Quantifiers

### 4.1 Representation of Connectives

Since in semantic networks, nodes are used to represent all information that can be discussed, nodes are used to represent both atomic and non-atomic propositions. (from now on we will refer to nodes which represent non-atomic propositions as rule nodes). The issue

of representing connectives consists of two questions: how should rule nodes be connected to atomic proposition nodes; how should the different connectives be represented.

We answer the first question by extension of the technique of using case relations in the representation of atomic propositions [3]. A case relation is a binary system relation between a node representing a proposition, event, or act and a node representing an individual, which indicates the semantic role of the latter in the former. A single case relation may be used in different kinds of propositions and events, but always indicates approximately the same kind of role. (The role may also depend on the event, especially in the case of the "object" case.) In accordance with the set orientation of our connectives, when several propositions play the same role in a rule, their proposition nodes will be connected to the rule node by the same system relation. Below, we list the labels of arcs going from a rule node *r* to a proposition node *n* and indicate the role played by *n* in *r*:

CQ — If *r* is true, it may be used to prove *n* true (*n* plays a consequent role in *r*).

DCQ — *n* is a default consequent of *r*. It can be asserted using *r* only if it is not already provably false.

ANT — If *r* is true, the truth of *n* helps determine the truth of the consequents of *r* (*n* plays an antecedent role in *r*).

&ANT — *n* plays an antecedent role in *r* but only in conjunction with other antecedents of *r*.

ARG $n$ plays both an antecedent and a consequent role in $r$ depending on how $r$ is being used and/or on whether $n$ is derivable independently of $r$.

One way of representing the different connectives is to have an operator arc from the rule node to a node representing the connective. This is the technique we used in [17,18], but we no longer see the need for storing additional information about the connectives, nor for retrieving all rule nodes that use a particular connective, so we have eliminated connective nodes and their accompanying ascending arcs. Two alternatives remain. First, the connective may be determined by the set of arcs emanating from the rule node. We use this for $\vee\!\!\rightarrow$, which is the connective assumed whenever a rule node has ANT and CQ arcs. The second alternative is to use auxiliary arcs to type the rule node with its main connective. This is especially useful for $\underset{n\,i}{\forall^{j}}$ $\underset{n\,i}{\theta}$, and $\overset{i}{\rightarrow}$ which have numerical parameters. The representation of each of our connectives is shown in Figures 2-8.

It would actually be very rare that a rule of the form $\Delta A$ would be stored, for at the time $\Delta A$ is asserted, either $\sim A$ is provable or it is not. If $\sim A$ is provable, $\Delta A$ would never accomplish anything. If $\nvdash \sim A$, then A could simply be asserted. In fact, one might view $\Delta A$ as the standard form of an assertion to a system that checks all input for consistency. Such a system either complains that it already knows that $\sim A$, or it stores A. Also, $\Delta A$ would seldom appear in an antecedent role. Because the most common use of $\Delta A$ is as a consequent, we allow $\vee\!\!\rightarrow$, $\&$, and $\overset{i}{\rightarrow}$ rule nodes to have CQ arcs, DCQ

arcs or both. This saves an extra level of nesting in rules of the form $(A_1,\ldots,A_n) \overset{i}{\rightarrow} (\Delta C)$.

## 4.2 Representation of Quantifiers

As with connectives, we no longer represent quantifiers as separate nodes, as we earlier did [17,18], but as arc relations between rule nodes and variable nodes. One may view a variable node as a pattern that stands for (matches) other nodes, and similarly for pattern nodes (molecular nodes with free variables). A quantifier may be considered as specifying the role a variable plays in the formula which comprises the scope of the quantifier.

A formula of the form $\forall\bar{x}A(\bar{x})$, where $\bar{x}$ is a sequence of variables, is represented by connecting arcs labelled AVB from the rule node representing $A(\bar{x})$ to each variable of $\bar{x}$. Similarly, a formula of the form $\exists\bar{x}A(x)$ is represented by connecting arcs labelled EVB from the rule node for $A(\bar{x})$ to each variable in $\bar{x}$. There is no need for putting formulas into prenex form first, they may be stored in the network as they are naturally expressed. An extra level is required in the representation of $\forall\bar{x}A(\bar{x})$ if $A(\bar{x})$ is an atomic proposition or is of the form $\exists\bar{y}B(\bar{y})$, or in the representation of $\exists\bar{x}A(\bar{x})$ if $A(\bar{x})$ is an atomic proposition or of the form $\forall\bar{y}A(\bar{y})$. In either case, $A(\bar{x})$ is represented as if it were $\{\}\vee\rightarrow\{A(\bar{x})\}$.

A rule of the form ${}_n\exists_i^j\bar{x}(P_1(\bar{x}),\ldots,P_k(\bar{x}):Q(\bar{x}))$ is represented by a rule node with an arc labelled PEVB to each variable in $\bar{x}$, an &ANT arc to the node representing $P_i(\bar{x})$, $1\leq i\leq k$, a CQ arc to the node for $Q(\bar{x})$ and auxiliary arcs labelled ETOT, EMIN and EMAX to $n$, $i$, and $j$ respectively.

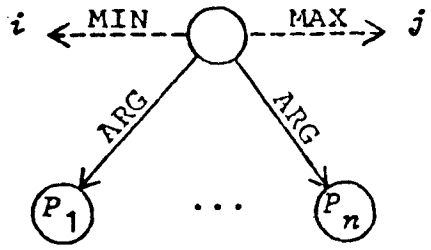


**Figure 2:** The network representation of

$${}_n\mathbb{X}_i^j \{P_1, \ldots, P_n\}$$

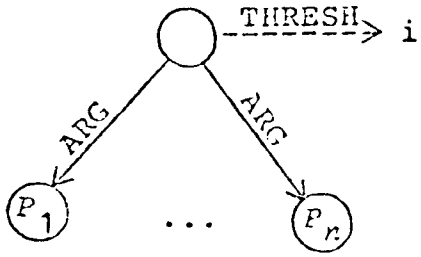


**Figure 3:** The network representation of

$${}_n\Theta_i \{P_1, \ldots, P_n\}$$

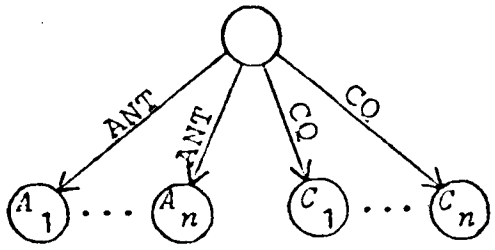


**Figure 4:** The network representation of

$$\{A_1, \ldots, A_n\} \vee\!\!\rightarrow \{C_1, \ldots, C_n\}$$

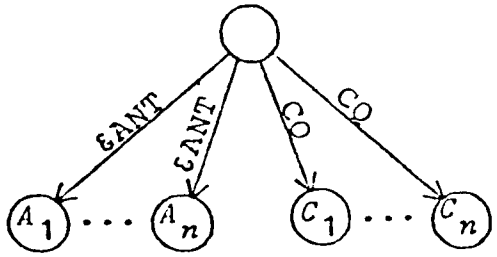


**Figure 5:** The network representation of

$$\{A_1, \ldots, A_n\} \&\!\!\rightarrow \{C_1, \ldots, C_n\}$$

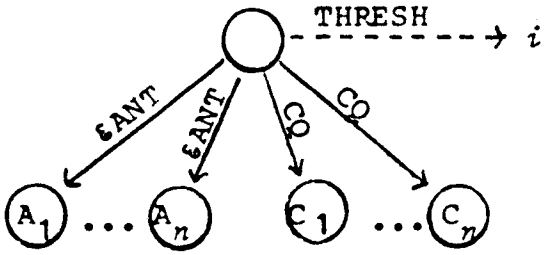


Figure 6: The network representation of $\{A_1,\ldots,A_n\} \overset{i}{\downarrow} \{C_1,\ldots,C_n\}$

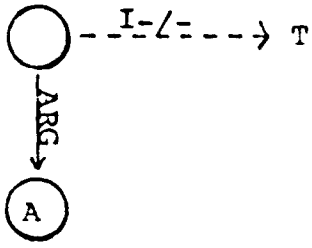


Figure 7: The network representation of $\vdash A$


Figure 8: The network representation of $\Delta A$

This discussion of the representation of connectives and quantifiers is summarized by Figures 9-13 in which several rules are exhibited.

## 5. Implementation

All the connectives and quantifiers discussed in this paper are implemented in the inference sub-system of SNePS, which is implemented in LISP on a CYBER-173 and runs in either interactive or batch mode. The system does backward inferencing and a restricted form of forward inferencing (see [20; p.195ff]). Top level rules and forms in consequent position are not restricted. However, the inference system is currently only capable of deriving atomic propositions, or formulae with ${}_n\mathbb{I}_i^j$ or $\vdash$ as main connectives, so currently these are the only allowable forms of questions and of antecedents. This is adequate for most question answering applications, but a more complete implementation is in progress.

## 6. Summary

In designing representations of common sense reasoning rules, logic-based connectives and quantifiers have the advantage that their semantics and inferential properties are clear and consistent. However, one need not be restricted to the standard, well-known connectives and quantifiers. We want such operators that enable us to express common modes of human reasoning simply, that take arbitrarily large sets of arguments, and that are no less adequate than the standard syntax of predicate calculus. The connectives we currently use are AND-OR (${}_n\mathbb{I}_i^j$), THRESH (${}_n\Theta_i$), $\underset{i}{\text{or}}$-entailment ($\vee\!\!\rightarrow$), and-entailment ($\&\!\!\rightarrow$), numerical-entailment ($\rightarrow$), non-derivable ($\not\vdash$), and default ($\Delta$). Our quantifiers are universal ($\forall$), existen-

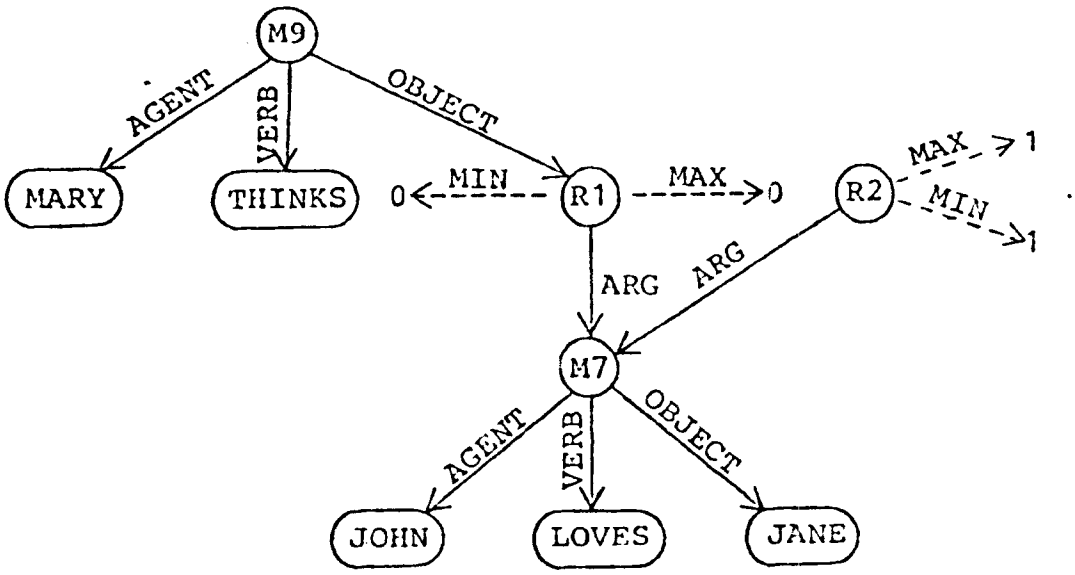


Figure 9: "Mary thinks John doesn't love Jane, but he does."
Rule nodes are labelled Rn.

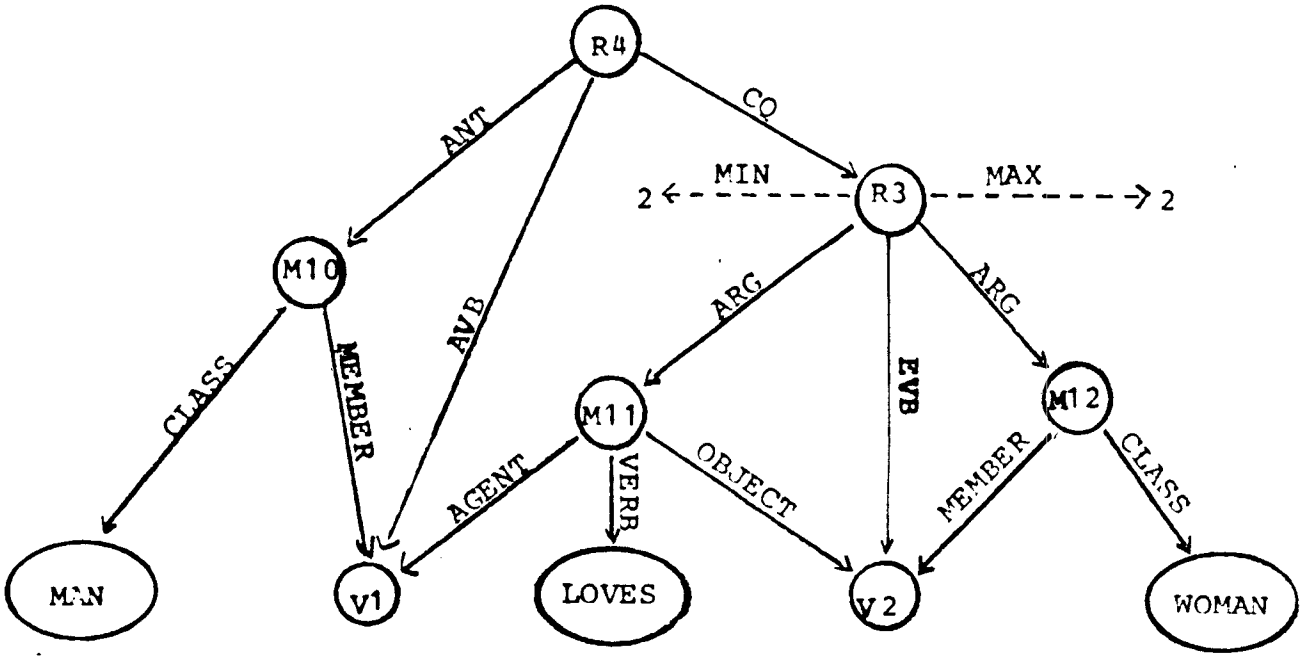


Figure 10: "Every man loves some woman."

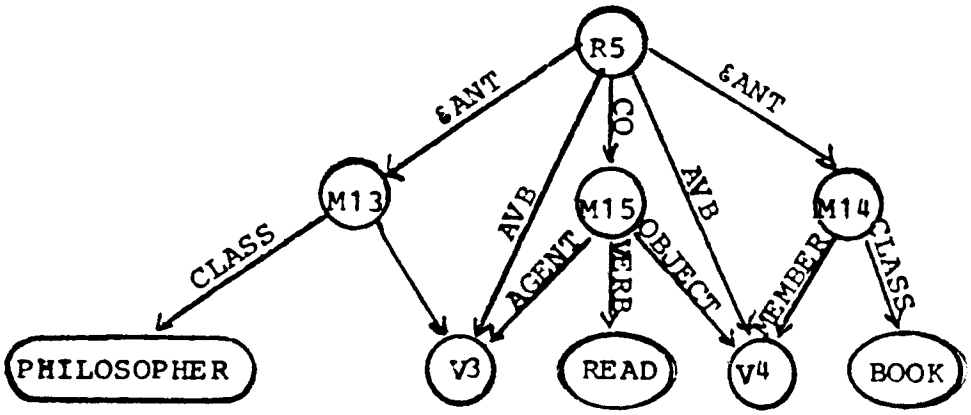


Figure 11: "All philosophers read all books."

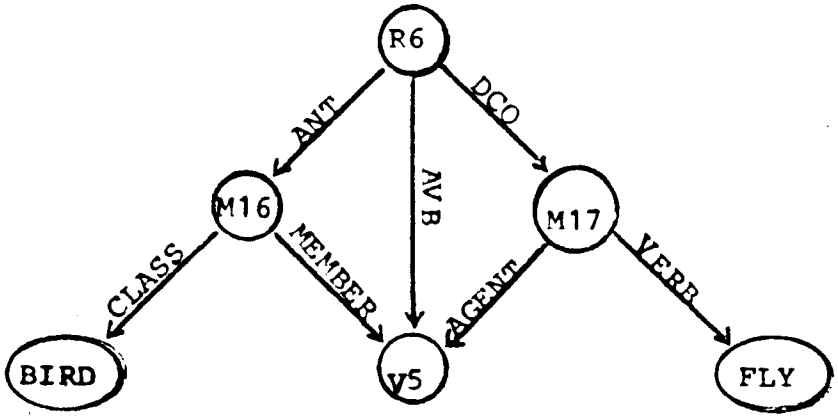


Figure 12: "Birds (in general) fly."

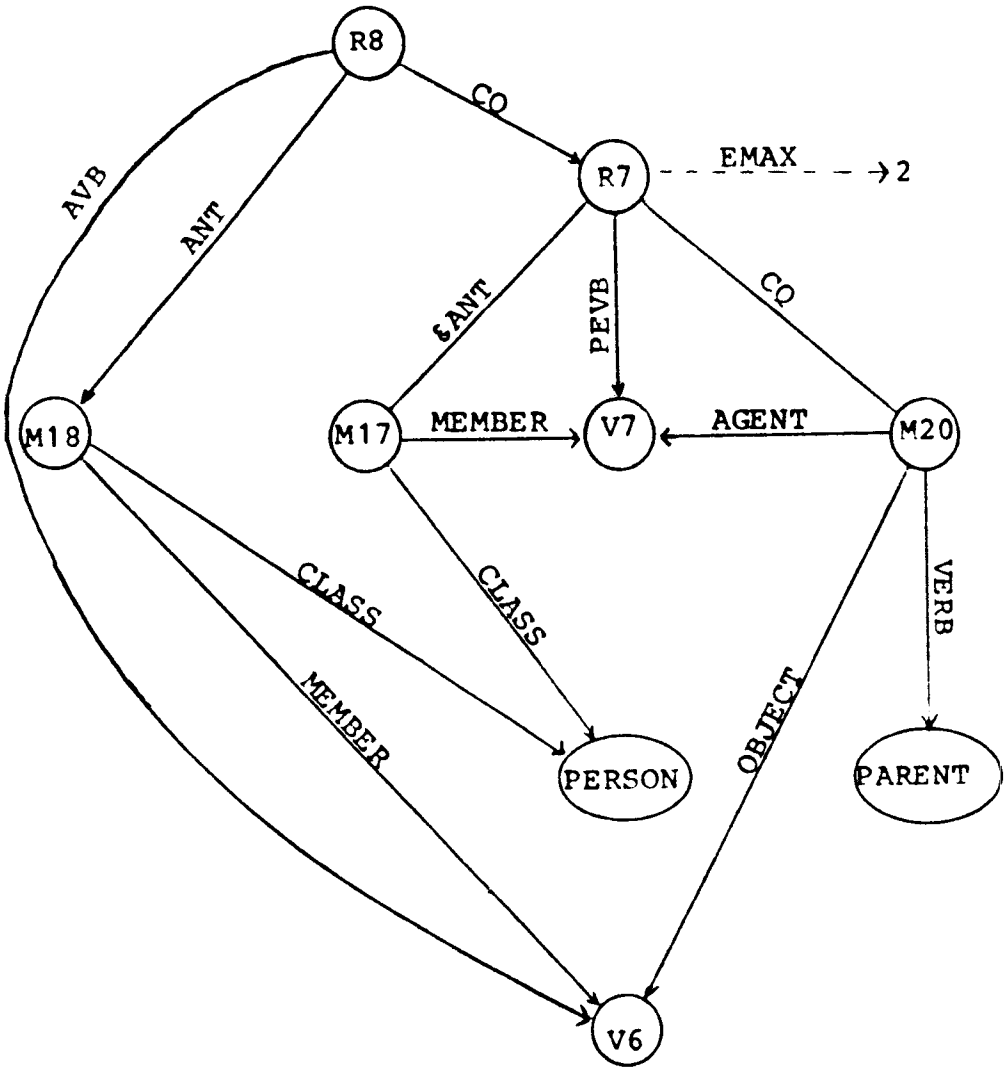


Figure 13: "Every person has at most two parents."

tial ($\exists$), and numerical ($_{n}\exists^{j}_{i}$). These connectives and quantifiers have been implemented in an operational semantic network based inference system.

References

1. Bechtel, R. J. Logic for semantic networks. M.S. Thesis. Technical Report No. 53, Computer Science Department, Indiana University, Bloomington, IN. July 1976.

2. Bechtel, R. J. and Shapiro, S. C. A logic for semantic networks. Technical Report No. 47, Computer Science Department, Indiana University, Bloomington, IN. March, 1976.

3. Bruce, B. Case systems for natural language. Artificial Intelligence 6, 4 (1975), 327-360.

4. Doyle, J. Truth maintenance systems for problem solving. MIT Artificial Intelligence Lab TR-419, 1978.

5. Doyle, J. A glimpse of truth maintenance. In P. H. Winston and R. H. Brown, eds. Artificial Intelligence: an MIT Perspective, Vol. 1, MIT Press, 1979, 119-135.

6. Fikes, R. and Hendrix, G. G. A network-based knowledge representation and its natural deduction system. Proceedings of the Fifth International Joint Conference on Artificial Intelligence, 1977, 235-246.

7. Findler, N. V., ed. Associative Networks: Representation and Use of Knowledge by Computers. Academic Press, New York, 1979.

8. Hayes, P. J. The frame problem and related problems in artificial intelligence. In A. Elithorn and D. Jones, eds. Artificial and Human Thinking, Jossey-Bass Inc., San Francisco, 1973, 45-59.

9. Hendrix, G. G. Expanding the utility of semantic networks through partitioning. Advance Papers of the Fourth International Joint Conference on Artificial Intelligence, 1975, 115-121.

10. Hendrix, G. G. Partitioned networks for the mathematical modeling of natural language semantics. Ph.D. Thesis. Technical Report NL-28, Department of Computer Sciences, The University of Texas at Austin. December 1975.

11. Hendrix, G. G. Encoding knowledge in partitioned networks. In [7], 51-92.

12. Kay, M. The MIND system. In Natural Language Processing, Rustin, R., ed. Algorithmics Press, New York, 1973, 155-188.

13. Reiter, R. On reasoning by default. In D. Waltz, ed. Theoretical Issues in Natural Language Processing-2, ACM, New York, 1978, 210-218.

14. Schubert, L. K. Extending the expressive power of semantic networks. Advance Papers of the Fourth International Joint Conference on Artificial Intelligence, 1975, 158-164.

15. Schubert, L. K. Extending the expressive power of semantic networks. Artificial Intelligence 7, 1976, 163-198.

16. Schubert, L. K., Goebel, R. G., and Cercone, N. J. The structure and organization of a semantic network for comprehension and inference. In [7], 122-175.

17. Shapiro, S. C. The MIND System: a data structure for semantic information processing. R-837-PR, The Rand Corporation, Santa Monica, California, August, 1971.

18. Shapiro, S. C. A net structure for semantic information storage, deduction and retrieval. Proc. Second International Joint Conference on Artificial Intelligence, The British Computer Society, London, England, September, 1971, 512-523.

19. Shapiro, S. C. Representing and locating deduction rules in a semantic network. Proc. Workshop Pattern-Directed Inference Systems, SIGART Newsletter 63 (1977), 14-18.

20. Shapiro, S. C. The SNePS semantic network processing system. In [7], 179-203.

21. Shapiro, S. C. Numerical quantifiers and their use in reasoning with negative information. Proceedings of the Sixth International Joint Conference on Artificial Intelligence, 1979.

22. Shapiro, S. C. and Wand, M. The relevance of relevance. TR No. 46, Computer Science Department, Indiana University, Bloomington, IN., 1976.

23. Woods, W. A. What's in a link: foundations for semantic networks. In Representation and Understanding, Bobrow, D. G. and Collins, A., eds. Academic Press, New York, 1975, 35-82.