Information Technology: Research and Development (1983), *2* (187–231)

SEMANTIC NETWORKS—A GENERAL DEFINITION AND A SURVEY

G. D. RITCHIE*

Department of Computer Science, Heriot-Watt University, Edinburgh, UK

AND

F. K. HANNA

Electronics Laboratories, University of Kent, Canterbury, UK

(*Received 13 April 1983, revised 20 July 1983*)

ABSTRACT

The term 'semantic network' has been used to describe a wide variety of representation techniques within artificial intelligence. We present a general definition of this kind of structure, and use it as a basis for a survey of several of these systems. We note some trends within the reviewed material, and suggest where further formalization and clarification would be desirable.

1. INTRODUCTION

In the development of computer models for 'knowledge representation' or for 'semantic structures', the idea of a 'semantic network' has frequently been used. Although this idea is usually regarded as having been introduced to artificial intelligence as the 'semantic memory' of Quillian (1968), and has been used (in a variety of forms) for several years (*see* Scragg, 1975, for an introduction, and Findler, 1979, for a collection of original articles), there does not seem to be any clear definition of what exactly constitutes a semantic network. This means that there is no common formalism within which different versions may be compared, and there is no way of proving any results about semantic networks. It has also led to some confusion regarding the meaning of network notations (*see* Woods, 1975, for an excellent critique). We will try here to set out such a definition, and discuss the ways in which it covers various aspects of the network systems commonly proposed. Although the precise mathematical statement will be presented in the Appendix, an outline of the salient features will be given (Section 2), followed by a discussion of the general characteristics of semantic networks (Section 3). The definition will then be used as the basis for a survey of several kinds of semantic network as described in the literature, to demonstrate the generality of the definition, and to allow some comparisons between systems.

* Now at Department of Artificial Intelligence, University of Edinburgh.

0144-817X/83/03 0187-45 $03.00

There will be no treatment here of formalisms which have evolved in the field of database management (cf. Abrial, 1974; Borkin, 1979), although it seems likely that there is a great deal of overlap with the concepts of artificial intelligence (*see* Wong and Mylopoulos, 1977). A prerequisite of such a comparative study is some kind of clear characterization of the structures typically used within artificial intelligence, and this paper contributes to that foundation.

We shall use the terms 'semantic net' and 'semantic network' interchangeably, as is common practice. Notice the distinction between a particular semantic net, which is a graph structure representing a particular state of 'knowledge', and a semantic net formalism, which is a set of rules, labels, etc., for building nets. Each author (e.g., Quillian, Schank) proposes his own semantic net formalism (e.g., semantic memory, conceptual dependency), within which an infinite class of particular semantic nets may be defined. We shall be characterizing both of these ideas here, in that we give a general definition of what constitutes a semantic net formalism (i.e., what it is each author should provide to define his system) and also a general definition of the kind of structures with which these formalisms deal.

2. SEMANTIC NETWORKS

2.1 Labelled nets

The structure underlying the typical semantic network is not, in fact, a graph in the strict mathematical sense, since a graph (or directed graph) as normally defined has at most one link between each ordered pair of nodes. The structure we need is a *net* as described in (Harary *et al.*, 1965). A net consists of:

1. (a) A set of nodes, N.
   (b) A set of arcs, A.
   (c) A mapping from A to N, called 'start'.
   (d) A mapping from A to N, called 'finish'.

This achieves the appropriate degree of separation between the arcs and their associated nodes. We will use this as the basis for our development of semantic network definitions, but the actual formal definitions are given in the Appendix. The reader should be able to follow the exposition here using the informal commentary and the mnemonic names used.

We shall assume that every node and arc in a network can have at most one 'label', where a label is in fact an arbitrary item (e.g., we may want to label nodes with procedures). The occurrence of 'unlabelled' nodes or arcs can be covered by including a special 'null' marker, BLANK, in the appropriate set of labels (this not only avoids the use of a labelling function which is partial, it allows a simple characterization of the notion of an 'unlabelled node'). Some of the labels will be 'organizational'; that is, they indicate the structuring within the knowledge network in a domain-independent way. Other labels will be 'substantive' in that they denote objects, relations or other entities in the subject matter. We will assume that arcs are never unlabelled, since there seems to be no need for an 'anonymous' association between two structures (the routines for handling the network would have difficulty interpreting it).

This leads us to the following components for a semantic network S:

2. (a) A net (N, A, start, finish).
   (b) A node-labelling of the net; that is, a set NL and a function nlab from N to NL.
   (c) An arc-labelling of the net; that is, a set AL and a function alab from A to AL.

There are also the following conditions:

3. (a) The set NL can be partitioned (disjointly) into three subsets, ONL (the 'organizational' labels), SNL (the 'substantive' node labels), and PNL (the 'partition' labels).
   (b) ONL contains a special item, BLANK.
   (c) PNL may be empty. If it is not, it contains semantic nets for which the node, arc and label sets are subsets of those in S (*see* Section 3.7).

2.2 Semantic network formalisms

The previous section outlined what was necessary in the definition of a *particular* semantic net. Now we will try to characterize what is embodied in a semantic net formalism; this involves specifying what building blocks (e.g., label sets) have to be given in a semantic net system, to enable the construction of semantic nets as described in Section 2.1 above. The formalism has to specify the permissible labels (node and arc), their permissible combinations, and how these labels are to be interpreted (i.e., what operations are valid). The second of these—the syntax of semantic nets—seems to have been overlooked, or at least underpublicized, by most users of networks (*see* Section 3.2 below). The third aspect (the net operations) will also not be developed here, owing to a lack of information within the literature.

A semantic net formalism will contain the following:

4. (a) A set ONL (the organizational node labels) containing a special element BLANK.
   (b) A set OAL (the organizational arc-labels).
   (c) A set T of node-types and type-features.
   (d) A set of network operations, OP. This will formalize the interpretations of the various labels, and define the possible manipulations of the networks. (Notice that this does not involve implementation details—these 'operations' are abstract mappings, independent of any particular realization of the network structures.)
   (e) A set of network formation rules, FR. This is a statement of the ways in which the various labels can be combined.
   (f) A semantic net programming language SPL. This will be necessary in order to specify procedures (again, at a suitably abstract level), but its details are not of interest here. The only comment to be made is that the interpretation (semantics) of this language should be phrased in terms of the network operations and structures. It may seem spurious to include (even sketchily) a programming language as a component of the definition of a kind of datastructure (i.e., a semantic net), but procedural attachment (*see* Section 3.8 below), should it ever be formalized, will need some such facility. We have therefore included it for this possible future enhancement.

Notice that this omits any mention of the set of substantive labels, SNL. This is because the particular domain-specific categories for a subject area are not part of a semantic net formalism, any more than they are part of the definition of what constitutes predicate logic. Specific semantic nets are formed, within a semantic network system, using the label sets ONL and OAL together with some set of substantive categories.

3. DISCUSSION OF DEFINITION

The previous section simply outlined the form of a semantic network system, without explaining the decisions taken in designing the definitions. We now examine various aspects of semantic networks in slightly greater detail, in order to justify the approach adopted in Section 2 above. We will start by considering the net structure, and how it may be labelled, then proceed to the question of the operations that are defined on the nets.

3.1 Graphical notation

Perhaps one reason why artificial intelligence data structures are commonly regarded as 'networks' is that this viewpoint lends itself to a clear, intuitively satisfying graphical notation. The characteristics normally described in terms of networks (i.e., by talking of 'nodes' and 'arcs') could be formally described in some other, non-graphical terminology with some different mathematical basis. However, we have tried to stay as close to the accepted outlook as possible, so that our formal definition has a firm relationship to the existing literature.

Diagrams used to represent semantic networks often leave certain connections or markings out of the network, where it is obvious how these would be drawn. For example, Norman and Rumelhart *et al.* (1975) have an abbreviated notation for 'secondary' nodes (instances of relations) whereby the link to the 'primary' node (the relation) is merely indicated by having the name of the primary written inside the secondary node. Simmons (1973) has several 'properties' (e.g., Tense) marked on to nodes, without regarding these as further nodes and arcs for the purpose of drawing the diagram. There is nothing wrong with such simplifying conventions (as long as they do not lead to confusion regarding the actual data connections required), but a fuller model will have to be more uniform in its approach. (Brachman, 1977, 1978, shows how more complex diagrams result from making all connections explicit.) In our formal model, we will assume that all information is held in the form of network links—the only structures will be nodes, arcs and labels on these items, without other annotations (e.g., 'property lists' attached to nodes). This means that a 'node' will be an atomic item from the point of view of this definition—it will not be possible to have more than a single item at a node, and the item will not, in general, have any internal structure which the network functions can operate on (except in the case of 'supernodes'—*see* Sections 4.2.3, 4.2.4).

3.2 Well-formedness of networks

One aspect of semantic networks which has been almost entirely overlooked in conventional treatments is the syntax of nodes and arcs; that is, the rules which determine what counts as a valid network. This is a necessary part of a fully formal definition of semantic nets, but it is difficult to know exactly what form such rules should take, in view of the lack of examples. The relevant constructs for specifying

well-formedness are node types, node-labels and arc-labels, since we will want to include statements such as 'an ISA-labelled arc must go from an instance-type node to a concept-type node'. In the absence of evidence, we will not present here a full formalism for these rules, since it is not clear how they are likely to be used in a practical system. It is important, nevertheless, to note that such rules would be a central part of a full definition of semantic net theory.

For many systems, it will be possible to make the simplifying assumption that well-formedness can be defined on a very localized basis, in the sense that the syntax rules do not need to describe gross characteristics of large areas of the net. It might be feasible (for most systems) to adopt the convention that a rule specifies only the arcs leading in or out of a node, and the nodes immediately at the ends of these arcs (together with the labels on these nodes and arcs), no sequences of more than one arc ever being specified. The rules could then be given as specifications of what arc–node combinations are allowable from a node of a given type and a given label. For this purpose, labels could usefully be classified into categories, since we will often want a rule to refer to a whole class of labels (e.g., 'inheritance links').

There are a great many questions to be considered before a scheme can be formalized. How can we combine the various parts of a specification for a single node type? How do we specify all the various options—exactly one, at most one, at least one, none—describing allowable arcs? Are the rules to be interpreted as constraints which no node–arc–node combination may violate, or as possible combinations such that every link must satisfy at least one rule? Do we need categories of node types, so that rules can refer to whole classes of node types? It is premature to introduce 'formal notations', unless we have a better idea of how these rules are to be used. The rules cannot be given in context-free grammar, for example, with its usual interpretation, since that would result in the nets being simply trees. (It might be possible if we regarded the rules as being generalized forms of the 'node-admissibility conditions' suggested by McCawley, 1968, for syntax trees.)

To illustrate the kind of information which would have to go into these rules, we shall give informal verbal definitions of the syntax of one or two of the formalisms in the survey in Section 4.

3.3 Organizational and substantive labels

Some writers about semantic networks do not make a clear distinction between the labels which are used to structure the knowledge representation and those which are part of the substance of the knowledge being described. For example, the 'ISA' link which was commonly used to connect an instance to its generic type was treated as being similar to the 'LOVE' link which connected two items entering into the 'LOVE' relation. The former class of links ('organizational links') is a small set of modes of connection which can be used to structure a network describing any form of knowledge—instances need to be linked to their generic types whether the subject matter is a world of building blocks or a world of Chinese submarines. The latter class of links ('substantive links') is a large set, the contents of which depend on what is being talked about—the 'SUPPORT' relation of the blocks world may not be relevant to a discussion of submarines. A general theory of semantic networks (as opposed to a particular computer program using domain-dependent tricks) will have to specify a set of organizational links, and an interpretation for each such link (i.e., a set of rules concerning its use). Such a universal approach to knowledge-

structuring can then be applied to the construction of semantic networks for any subject matter. Brachman (1979) argues for a general approach like this, but he distinguishes two categories of organizational links—'logical' and 'epistemological'. Our definition will not rule out such a subdivision, but we do not need to build it into the general framework.

It is interesting to notice that a suitably clean and sophisticated semantic net will usually have all the substantive labels attached to nodes, and not to arcs, even though the early illustrative networks often included arcs labelled 'LOVES' or 'SUPPORT'. Any complex network in which relations were more than atomic labels would need to have the relation names attached to nodes, with the various slot-fillers (arguments) attached to the relation-node by suitable links (*see*, for example, Norman and Rumelhart *et al.*, 1975; Brachman, 1977, 1978, 1979). We have not built a stipulation to this effect into our definition.

3.4 Inheritance

Semantic networks have been used in many ways, and it is tempting to conclude that the notion of 'semantic network' is almost vacuous. It often seems to mean simply 'some form of labelled graph, used to represent a "meaning" or "knowledge structure"'. Since virtually anything can be regarded as a labelled graph, this seems to be a fairly trivial concept. Nevertheless, there seem to be one or two characteristic operations carried out on semantic networks, and these may embody the essential features of the 'semantic network approach'.

It has become customary, in semantic networks, to link each 'instance' of a 'generic concept' to the concept in question, and to link concepts to more general concepts ('super-concepts'). This linkage indicates that the more specific item (either instance or sub-concept) takes on some or all of the properties of the more general concept; this is usually referred to as 'inheritance', and appears to be one of the main ideas underlying semantic network systems. There are various advantages that result from this arrangement. Not only is the representation more compact (details common to several sub-concepts can be held on their superordinate concept), but simply assigning an item to a concept means that the item 'inherits' all the properties of that concept (*see* Fahlman, 1979, for a fuller discussion). Often, no distinction is made between the two kinds of inheritance (instance from concept and sub-concept from super-concept), and the inheriting connection is marked by an ambiguous label (traditionally, 'ISA'). We allow for all these possibilities in our model.

Although the traditional networks inherit only along a single link, corresponding to one of the simple ways just described, more recent versions (e.g., KRL: Bobrow and Winograd, 1977) allow several inheritance links from a single node, and Brachman (1978) allows inheritance to happen in various ways along various links. Once again, the theoretical definition will have to allow all these possibilities.

It does not seem to be customary to link generic concepts to all their instances, but it is not clear whether this is merely to simplify the diagrams (since the human reader can work out such details easily), or whether it represents a claim about the way knowledge should be structured. Certainly, particular implementations are liable to use this kind of connection to facilitate the network searches, but that does not mean that it is part of the abstract structure (*see* Section 3.9 below).

3.5 Node types

Although the characteristics of an arc seem to be wholly embodied in its label, there

is more that can be said about a node than simply specifying the label that it has. Norman and Rumelhart *et al.* (1975) have 'primary' nodes (relations) and 'secondary' nodes (relation instances), which are presumably treated differently by the network handling routines. Quillian (1968) had 'type' and 'token' nodes, although this distinction is not identical to that used by Norman and Rumelhart. Brachman (1978) uses various types of nodes, and adopts the graphical convention of drawing them differently (diamonds, ovals, etc.) to indicate this. Thus it appears that a semantic net does not have a simple set of nodes, but has a (disjoint) union of several sets of nodes (corresponding to the different types). Furthermore, some types have subtypes (for example, FRL: Roberts and Goldstein, 1977, has 'frame' nodes which can be classed as either generic or individual). This could be handled by decomposing each type into disjoint subtypes, but Fahlman (1979) uses 'modifiers' (of which there may be more than one on a node, as subcategorization markings), and this seems to need a set of 'type-features' (rather than the allocation of a node to a single type). This would allow the rules of a particular system to be phrased in terms of combinations of these type-features. It might seem to be more uniform to remove 'basic types' from this definition, and simply have a set of 'type-features', but it is noticeable that most written descriptions of semantic nets use the idea of a node having a 'basic type'. This leads naturally to the notion of subtypes as combinations of type-features. The fact that certain features cannot occur together, or that the presence of one feature may require the presence of another, would be part of the rules given to define the semantic net system. (The 'systems network' notation of Halliday (*see* Hudson, 1971; Winograd, 1972) would be appropriate for expressing these constraints.)

3.6 Data items as node labels

The labels in semantic networks will be of various sorts. As well as organizational arc-labels (e.g., ISA) and substantive node-labels (e.g., SUPPORT), there will be certain node-labels (both organizational and substantive) which are best regarded as data items outside the network system, in that they do not have any special network meaning (e.g., the numeral 3). In many cases, these will require the inclusion, in the label sets, of large (often infinite) sets of primitive data items (e.g., the integers, the set of alphabetic strings). Also, there will be certain special node-labels which are not part of infinite data types, but which come from small sets of special items (but which, like the integers, are atomic from the point of view of network structure). The obvious examples are labels like 'TRUE' and 'FALSE'. None of these kinds of node-labels are central enough to the notion of 'semantic network' for them to be built into the definition of what constitutes a semantic network, and we will merely observe that the semantic net system designer may have to include a wide variety of data items in his label sets.

3.7 Contexts and partitions

Some artificial intelligence systems (e.g., Conniver: McDermott, 1973) have their database organized into various 'contexts' or 'states of the world'. There are operations available for moving between contexts, and creating new contexts, so that variables may have different values in different contexts (*see* McDermott, 1975). Typically, the contexts are arranged in a tree-like structure, so that a context 'inherits' the state of the world from its 'parent' context. (This allows a relatively efficient implementation in which only *changes* to the world are recorded, with the

information from previous contexts being left 'visible' until altered.) Since each context represents an entire 'state of the world', which in a semantic net system would be an entire semantic net, it is suitable, at our present abstract level, to define context systems as maintaining trees of semantic nets. That is, there is a tree structure, totally outside the definition of the semantic net structure, such that each vertex of the tree contains a single semantic net (representing the 'state of the world' in that context). The context-moving and context-creating operators can then be defined on trees of this sort. Should it be necessary, the context configuration could be generalized from a tree to a general acyclic directed graph, although most actual systems avoid the complications which arise from this (since it needs the idea of 'merging' two contexts).

This is relatively elegant, but is not sufficient to cover the case of 'partitioned semantic nets' (Section 4.2.3 below). Hendrix (1979) groups parts of the semantic net into 'spaces' or 'partitions', and uses these larger chunks in various ways (mainly for variable-scoping). In terms of what this means, it is reminiscent of 'contexts', but the exact details of 'partitioning' prove to be more complex; in particular, this technique needs the 'partitions' to be referred to within the network (i.e., to be represented as nodes with arcs leading to them). Hendrix defines 'supernodes' for this purpose, which are different from ordinary nodes in that they indicate whole chunks of semantic net. It might be possible to form a general formalization of this scheme and the context-graph idea by allowing the set of 'supernodes' to be either part of the set of nodes (partitioning) or totally disjoint (contexts), but we have not attempted this here. Since partitioning requires an association between the supernode and an arbitrary semantic network, it can be handled by letting the node-label set for a partitioned semantic net include the set of possible semantic nets within the system.

3.8 Operations on networks

Although semantic nets take on their meaning only by virtue of the operations that can be performed on them, there is a tendency in the literature to omit descriptions of the valid operations on the proposed structures. Those descriptions that do exist show little in the way of commonality. If we attempt to generalize such ill-defined and varied systems, we will end up with the near-vacuous statement that a semantic network system allows arbitrary graph manipulations, operating on datastructures of the sorts outlined in Section 2 above. Unfortunately, it is difficult to provide a simple way of specifying semantic net operations, although various ways of defining data structure operations exist (e.g., using 'abstract data types': Guttag and Horning, 1978). We shall therefore, for simplicity, not attempt to incorporate formal definitions of operations into our characterization of semantic nets. In the survey of semantic net systems (Section 4 below), we shall describe some of the operations, and the 'meaning' of the labels, informally, as is the custom in the literature. The exposition will, we hope, be suitably clear.

This means that we shall have to gloss over, to a large extent, the question of 'procedural attachment' (i.e., the insertion of procedure definitions as executable items within a data structure). In the absence of a formal definition of the notion of operations on a semantic net, the idea of a procedure (which can give rise to a sequence of operation-applications) cannot be considered. In the survey in Section 4, we shall simply note, where necessary, that certain formalisms allow a node-label which is a procedure.

3.9 Nature of links

One very hazy aspect of the semantic network literature is what a link between two nodes indicates, in terms of structure-accessing. (There is also, as Woods, 1975, points out, some obscurity about what the links represent in terms of semantic relations—*see* 3.11 below.) Although early net programs may have identified semantic net links with simple pointers in the actual implementation, this is not their import in later systems. However, a link must have some effect on the ease with which the interpreter can move from one structure to another, otherwise the traditional ISA link would not contribute to inference. The net may be held in some uniformly indexed fashion (involving hash-coding, for example), but that level of machine retrieval is not part of the knowledge structuring. On the other hand, we must be careful not to assume that the direction of an arc necessarily indicates a one-directional access path. Although an arc must have direction in order to distinguish which of the two end-nodes is which (i.e., the order of arguments in the 'relation' given by the arc-label), that does not necessarily mean that arcs are one-way access paths. It might be tempting to adopt the assumption that all arcs are implicitly bi-directional; that is, that access paths in the network are possible along either direction of each arc. Careful consideration reveals that this would lead to some counterintuitive results. Remember that the purpose of the net is to impose some (theoretically interesting) structure upon the knowledge, not merely to reflect the full range of possible associations between all items. Therefore, we want to be able to state, as part of a semantic net description of some area of knowledge, which pieces of knowledge are accessible from which others. It is not necessarily the case, for example, that there should be an access path from every predicate or relation node to all the 'assertions' involving it; although that might be useful for predicates or relations which represent substantive concepts from the topic being described (e.g., 'SUPPORT'), it is not very plausible for the 'disjunction' relation (cf. Schubert *et al.*, 1979), since it is not relevant for the system to be able to retrieve 'all the disjunctions within the system'. Hence a diagram like Figure 1, although it has arcs from the central ('proposition') node to the operator (to indicate their connection), does not necessarily define an access path from the operator to the arguments.

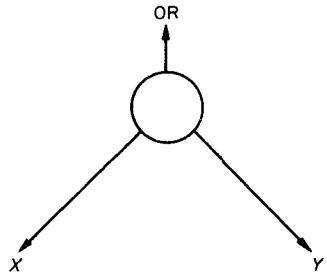

FIG. 1

The solution seems to be to class some arcs as 'two-way', and others as 'one-way'. We could represent two-way links by inserting another arc explicitly in the net, but this would move our description further from the diagrams commonly used, and

require the cumbersome use of inverse labels for these extra arcs. Using 'one-way' and 'two-way' allows us to distinguish a 'path', in the usual graph-theoretic sense (following the directions of all the arcs involved) from an 'access path' (where either direction of a two-way arc may be used). In some cases (e.g., following transitive chains of ISA links), there may be a need for a traditional path; in other cases, the more specialized notion of access path may be needed. Authors usually do not specify which of their arcs are two-way, which is a remarkable omission in view of the fact that one of the main purposes of a semantic net is to describe connections between items.

3.10 Semantic nets and predicate logic

The relationship between logic and semantic networks is a source of constant confusion, largely because of disagreement about what these terms mean. As Bundy (1979) observes, the syntactic aspect of predicate logic can be represented in network notation, either in traditional (quantified) notation (e.g., Schubert *et al.*, 1979) or in clausal form (e.g., Deliyanni and Kowalski, 1979). Of course, this depends on a suitable choice of node types and labels—not every semantic net formalism is an alternative syntactic representation of predicate logic, although it is straightforward to design one which is. The converse question is equally confusing—can any semantic net be written as a logic expression? It would be possible, as Bundy points out, to replace a network arc labelled 'P', from node X to node Y, with a term 'P(X,Y)'. This is not enough, however, to show that every semantic net formalism is simply a re-formulation of predicate logic, since we must then consider the interpretation of these structures, and the operations to be performed on them. Logic allows the X and Y in 'P(X,Y)' to be variables (bound or free) or constants. Will it always be the case that this reflects the intended use of the semantic net arc which the predicate-application is replacing? Study of the descriptions given in Section 4 should be enough to demonstrate that this is not always the case. For example, consider the DSUPERC link in Figure 9 (Section 4.3.2 below). If we write this as

DSUPERC (List-command, Printing-command)

does this have the meaning that there are objects List-command and Printing-command, with a relation DSUPERC between them? This would be the case only if we uniformly identify nodes with objects and arcs with relations, which is merely a re-expression of the network structure in other terms—it says nothing about the *meaning* of the net. An alternative argument might be to claim that the meaning of the DSUPERC link can be expressed as

$$(\forall x)\,(\text{List-command}(x) = > \text{Printing-command}(x))$$

Even if we leave aside the question of whether this is the exact meaning intended by the net-designer, all that this shows is that predicate logic can be used to describe the meaning of (parts of) semantic nets; that is, it illustrates the use of predicate calculus as a meta-language.

Hayes (1977) discusses the question of providing a 'meaning' for a notation or representation, and remarks that some interpretation is needed for semantic nets. He goes on to argue that if this meaning is specified in terms of objects and relations

(as in model theoretic semantics for logic) then semantic nets are merely another form of logic. This line of argument seems to depend on two assertions—that the only valid form of interpretation that can be provided is a model-theoretic one (cf. McDermott, 1978), and that any system with model-theoretic semantics is a logic. Under these rather strong assumptions, any representational system can be claimed to be a logic, so it is hardly surprising that semantic nets are thus classified. A logical system has a syntax and a semantics, and the mapping between the two is an essential aspect of its definition. A particular semantic net formalism may mimic the syntax of some logical system and it may also be defined to have an 'interpretation' as in traditional Tarski logic, but unless the correspondence between the two components (syntax and semantics) is effectively the same as that for some logic, it is hard to claim that the entire system is 'equivalent' to that logic.

Hayes raises many other interesting questions, including the fact that logic does not include processing rules; inference *strategies* can be imposed upon a logical representation, but they are not essential to logic. This gives another reason why care must be taken in comparing semantic networks and logic (other than Bundy's observation that nets by themselves are purely a syntactic form). A particular semantic net formalism includes not only a syntactic notation, but a set of rules (operations). Even if these rules are compatible with a logical interpretation (as in Hendrix, 1979), they can be seen as a superstructure with respect to the definitions of the representation and its interpretation. Hence, a full semantic network formalism, including the various operating rules, may not be exactly comparable to pure first-order predicate logic, since these operations correspond to inference strategies. It might, of course, be comparable to some predicate logic based program, such as a theorem-prover.

If the relevant notion of 'equivalence' or 'power' is based on the range of 'statements' that can be expressed in the formalism, predicate logic is probably more powerful than most semantic net formalisms (although those versions which include some kind of quantification may equal the expressive power of predicate logic). However, is this the relevant notion of 'equivalence'? The argument is analogous to that concerning the equivalence between various computational mechanisms. Although there are many devices which have the same computing power as Turing machines (rewrite grammars, register machines, transformational grammars), they are not equally suitable for all purposes (consider trying to write a grammar of English using a Turing machine). For example, if we are trying to devise a representational system which contains constructs and operations suitable for describing the meanings of English sentences (which seems to be the motivation behind most semantic net work) then it is not necessarily true that first-order predicate logic is the best system.

Some question-answering programs use a semantic network to represent all the information which is 'true' in their world; any assertion not present in the network is assumed to be 'false', and there is no distinction between 'unknown' and 'false' statements. We have not regarded a network as having any implicit claims as to truth or falsity—such aspects of statements can be represented explicitly in the network if required (cf. Kay, 1973). This does not mean that our definition will not cover those networks in which this implicit assumption is used; it is perfectly allowable, within our general theory, for a particular designer to interpret the presence of items in the network as 'truth', and to build operational procedures which are consistent with this idea. Nevertheless this approach is not an essential part of the notion of a semantic network.

3.11 Semantic confusions in networks

Particularly in earlier systems, there is a tendency for semantic networks to be designed without too much consideration of the exact meaning of the node and arc configurations. Woods (1975) discusses these deficiencies at some length, and it is not feasible to recapitulate his arguments in full, but we will outline briefly some of the issues.

Most of the difficulties spring from disregard for certain distinctions of meaning —general/specific, intension/extension, assertion/description and attributive/referential.

General/specific: In a machine representation of knowledge about the world, there is a need to represent general statements (e.g., 'all swans are white') as well as specific facts about the current state of the world (e.g., 'a white swan is sitting by the lake'). (In a conventional database system, the former might well be part of the data dictionary or data schemata, defining the possible configurations of data, whereas the latter would be in the actual database of facts.)

Intension/extension: A distinction can be made between the meaning of a descriptive term (e.g., 'a white dog') and the objects in the world of which that description is true (e.g., the set of white dogs). (Cf. Frege's sense/reference distinction: Geach and Black, 1960).

Assertion/description: A predicate (e.g., 'red') can be combined with an argument (e.g., 'ball') to make an assertion (that a particular ball is red) or to form a description ('the ball which is red').

Attributive/referential: In certain contexts, descriptive expressions can have two possible interpretations. For example (5a) can be roughly synonymous with (5b) or with (5c) depending on the interpretation of 'a Norwegian'.

5. (a) Mary wants to marry a Norwegian.
   (b) There is a particular person that Mary wants to marry, and who is a Norwegian.
   (c) Mary has decided that whoever she marries must be Norwegian.

Early semantic networks did not take sufficient account of all these distinctions, with the result that the same node-arc configuration would be used to represent some or all of the following:

6. (a) One of the defining properties of being a swan is being white.
   (b) All swans in the current world happen to be white.
   (c) That subset of swans defined by having the property white.
   (d) A subset of swans, all of which happen to be white.
   (e) The fact that a particular subset of swans are white.

4. A REVIEW OF SEMANTIC NET SYSTEMS

Having outlined the general definition of semantic nets, we will now use that framework to give descriptions of various semantic network systems which have been proposed over the past 15 years. Of course, the written descriptions from which we are working are likely to contain incomplete specifications, or to contain minor inconsistencies of notation. We will try to remedy such omissions or irregularities when they occur, in ways which seem to preserve the essential characteristics of the

systems and which do not decrease their elegance. The formalisms described here form a fairly representative cross-section of the literature on the topic, and include the most widely-cited examples as well as some lesser known ones.

On the basis of the systems surveyed below (and a few others which we have looked at, such as Shapiro, 1979), there seem to be three styles of semantic representation system within the broad class of semantic networks—uniform semantic networks, predicate logic-based systems, and frame systems—so we have organized the survey under these loose categorizations.

4.1 Uniform semantic networks

Historically, the uniform networks are the earlier examples, and are typified by the systems of Quillian, Rumelhart and Norman, Rieger, Simmons, Schank. They have (typically, but not always), the following characteristics:

7. (a) They may contain 'example' nodes which are linked to 'generic' nodes. These 'examples' are, in general, instances of arbitrary *n*-ary relations, and the 'generic' node contains any information about that relation.
   (b) There is not a great deal of 'generic' information about relations (in comparison with the 'frame systems' described below); very often there are just type-restrictions on the argument-places.
   (c) Sometimes (but not often) the 'generic' nodes are arranged into some kind of hierarchy, so that each concept can be linked to its generalizations (up the hierarchy) and its specializations (down the hierarchy).
   (d) In the cases where the hierarchy mentioned in (c) is built in, this is used for checking compatibility of semantic classifications (e.g., whether two sets, which some element is supposed to belong to, are disjoint, or whether some predicate can be validly applied to a given node).
   (e) The transitivity of classification represented by these hierarchies of generic concepts ((c) above) are not usually used to allow 'inheritance' of arbitrary properties.

Hence, such systems tend not to have much structure or groupings (as is found in frame systems), and do not make use of standard logical semantics (e.g., depending on resolution-based inference procedures).

4.1.1 Semantic memory

Although Quillian (1968) did not use the term 'semantic net', his rudimentary computer model of memory, based on linked nodes, is widely cited as the source of the idea of semantic networks within artificial intelligence. In fact, his system differed in many respects from later systems, in that it dealt only with word definitions (that is, it was an elaborate form of dictionary). There was no representation for specific uses of words (e.g., in particular sentences or dialogues), except insofar as they appeared in the definition of other words, and there was no 'inference' which made use of the arc labels (since it was not intended as a general 'knowledge representation' scheme). Also, his use of the node-and-arc graphical notation was partly an expository device, in the sense that he did not try to represent every association between items as a link; there were various labelling conventions to cover other connections. Here, we will try to tidy up this notation slightly, by making all such connections uniform and explicit. We shall, for example, have to

introduce 'conjunction' and 'disjunction' nodes (Fig. 3), in order to describe Quillian's configurations shown in Figure 2.

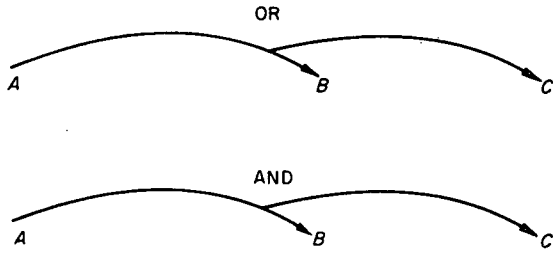

FIG. 2

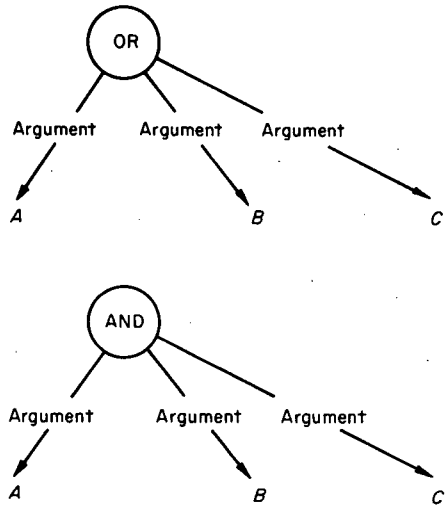

FIG. 3

The structural aspects of Quillian's system can be summarized in the following terms.

Types of nodes

Type: this (confusingly named) kind of node is the centre of a definition.
Token: marks the presence of one word definition (type node) in another definition net.
Disjunction: groups two or more nodes which form a disjunctive set.
Conjunction: groups two or more nodes which form a conjunction.

Organizational arc labels

Token-to-type: indicates definition being referred to from within another definition.
Subclass: indicates that the type (initial node) is a subclass of the class indicated by the final node.

Modifies: indicates that the final node modifies the initial node.
Subject: indicates that the final node is the 'subject' of the relation labelled on the initial node.
Object: indicates that the final node is the 'object' of the relation labelled on the initial node.
Argument: indicates that the final node is one of the arguments of the disjunction or conjunction represented by the initial node.

Substantive labels

The only substantive labels in the system are word-names, labelled on 'type' nodes.

Syntactic rules

Substantive node-labels appear only on type nodes; conjunction and disjunction nodes are unlabelled.
A token-to-type arc must go from a token node to a type node.
A subclass arc must go from a type node to a token node.
An arc labelled modifies, subject or object, must go from a token node to a token node.
An arc labelled argument must go from either a conjunction node or a disjunction node, to a token node.
No arcs except argument arcs may leave a conjunction or disjunction node.
Every conjunction or disjunction node must have two or more outgoing argument arcs.
Every token node must have exactly one token-to-type arc leaving it.

Before proceeding to the operations and processes available in Quillian's system, it is useful to make a few general observations. Although Quillian (1968: 239) mentions the notion of 'inheritance', there seems to be no implementation of it in his system. Clearly the 'token-to-type' and 'subclass' arcs are the forerunners of the later variants on 'ISA' links, but Quillian gives no specific interpretation for them. All Quillian's arcs are one-way (p. 239), although he comments that some form of 'recognition procedure' working on these memory structures might require back-pointers (to trace from attributes to definition).

The above description has eliminated some of the information which, in Quillian's diagrams, appears as extra labels on the nodes. In particular, the diagram for each definition is strictly tree-shaped, with no internal pointers from token nodes to other nodes nearer to the root. However, there are cases where such links are needed, and Quillian incorporates these by having two special labels '=A' and '=B' which can be used to set up pointers to higher nodes in the current definitional tree. This appears to be an implementation detail which has crept into the diagrammatic representation, and can be eliminated by simply allowing the necessary arcs to be drawn in directly. (Quillian also comments that there is a special 'indirect addressing' technique to allow these extra pointers to connect to nodes *outside* the current definition, but gives no details; these could be dealt with similarly by using direct arcs.) Although the additional markings =A and =B are useful for the diagrammatic representation, they do not alter the content of the net, and so they are unnecessary at this abstract level of definition.

Other annotations which appear in Quillian's diagrams are the labels 'S', 'D', and 'M', which are supposed to indicate the locations in the definition net for the insertion of items corresponding to the 'subject' 'direct object' and 'modifier' of the

word being defined. This facility is a rather ad hoc form of 'case grammar' markings (Fillmore, 1968), and is somewhat out of place in a knowledge-representation network, since it provides the linguistic information about how to map verb structures (in some canonical sentential form, presumably) into deep meaning representations. Although these labels are used by Quillian's program in trying to map inputs on to memory graphs, it could be argued (and we will assume here) that these rudimentary 'case-placement rules' are not properly part of the semantic net at all, but are really connections from the memory structures to a different level of items (linguistic rules of some sort). Coupled to these case-labels (S, D and M) are certain 'clue-words' which the program can use to work out which of the linguistic constituents in the input are to be allocated to the D and M roles. Few details of these are given in the article cited here, but the above remarks about the case-labels seem to apply equally to the clue-words; that is, they are surface syntactic rules which happen to be linked to the semantic net, but which are not strictly part of it.

Let us now look at the operations which can be carried out on a Quillian net. The main process in this memory model is 'spreading activation'. That is, the network interpreter works outward from some specified type node, 'tagging' each node that it encounters. This gives rise to a (gradually increasing) subgraph of tagged nodes around the original node (the 'patriarch', in Quillian's terminology). The program can use this technique to compare or relate two arbitrarily chosen words, by producing activation subgraphs from the two type nodes corresponding to these words and finding where they intersect.

The exact nature of this 'activation' process is as follows. When a node is tagged, two pieces of information are marked on it:

8. (a) What 'patriarch' is the origin of the spread of activation.
   (b) From which node the activation directly came (i.e., the immediate neighbour from which the interpreter moved to this node); this allows a path to be traced from any active node back to its 'patriarch'.

The set of tagged nodes for a particular patriarch can be regarded as a set of triples, containing three nodes—the one tagged, its patriarch, and its preceding neighbour in the activation process. The set of tagged nodes in a network (at any given moment) will then be a family of tagged-node sets (one for each patriarch activated). Notice that the semantic network system does not need any new constructs or definitions to allow description of tagging—it is all describable as the addition of nodes (or tuples involving nodes) to some set of tagged nodes (or sets of differently tagged nodes, in the case of Fahlman's system (*see* 4.3.1 below)).

4.1.2 Simmons' semantic networks

The networks outlined by Simmons (1973) are slightly different from those of Quillian, in several ways:

9. (a) Each network depicts the meaning of a specific sentence, rather than the relationships between general word meanings.
   (b) A node in a diagram is not necessarily a simple atom or pointer, but can be a list of attributes with values (e.g., attribute-value pairs such as (TENSE: Past) and (ESSENCE: Indeterminate)).
   (c) The idea of 'token' is altered subtly. A node can have a TOK link to the

concept of which it is a token, but whereas Quillian's 'token-to-type' link was merely a pointer indicating that another concept was involved in the current definition, Simmons' TOK link runs from an *example* of the concept to the defining node. It is hard to make this intuitive distinction precise, and this may not be necessary, since all that we are concerned with here are the formal interpretations imposed by the various network operations in the two systems.

(d) Simmons does not use semantic networks to encode general knowledge about concepts, in the way that Quillian does. All generalities are represented in other ways (paraphrase rules, etc.) which act on networks but which are not themselves in network form.

(e) The networks for individual sentences are strict trees, corresponding fairly closely to the syntactic structure of the original sentence. In fact, Simmons gives a formal syntax for his networks in a form of context-free grammar, which emphasizes the tree-structuring (and the linguistic syntactic influence).

Like Quillian, Simmons uses links which are not strictly binary, to handle conjunctions, although he does not discuss them at length or offer a graphical illustration:

> Our representation of these important terms in semantic structures is to form a TOKen structure followed by a list of arguments, as illustrated below.
>
> C1 TOK (any conjunction), ARGS C2, C3, C4 . . . (Simmons, 1973: 75).

This is a similar suggestion to the notational convention used in 4.1.1 above for Quillian's conjunction nodes, except that Simmons is allowing an open class of conjunctions (i.e., they now become substantive labels).

Simmons also uses 'case-frame' like labels—MODAL, LOC, THEME—to connect the argument nodes to a relational node. It is not at once clear how to classify these kinds of labels, in terms of the dichotomy between 'organizational' and 'substantive'. Notions like 'THEME' and 'LOCUS' certainly seem to denote aspects of the knowledge being described, rather than structural relationships between the meaning structures; thus they seem to be substantive. However, this may be a misleading impression given by the mnemonic label-names, since what is really relevant is how the network interpreter treats these arcs. If these links are merely arbitrarily-labelled arcs to join the argument nodes to the relational node, and could function just as well if they were labelled 'ARG1', 'ARG2', etc., then they are definitely organizational labels. (*See* Charniak, 1975; Wilks, 1976; Ritchie, 1980, for related arguments about the use of 'cases' in artificial intelligence representational systems.) The answer to this depends on the content of Simmons' various rules. Syntactically, the five 'case-relations' appear in the syntactic rules, and are therefore part of the built-in network system (i.e., they do not vary with subject matter). The matter is somewhat complicated by Simmons' inconsistent notation for these matters—although he has a single case-relation 'CAUSALACTANT', he allows two causal actants in a single 'proposition', labelling them CA1 and CA2 to distinguish them. Similarly, he allows two THEMEs, labelled T1 and T2 as arguments in the same structure.

Simmons' grammar for his net structure is:

S -> Modality + Proposition
Modality -> Tense, Aspect, Form, Mood, Modal, Manner, Time
Proposition -> Vb + (CASEARGUMENT)*
Vb -> run, walk, break, etc.,
CASEARGUMENT -> CASERELATION + [NP ¦ S]
NP -> (prep) + (DET) × (ADJ)* + (N) + N + (S ¦ NP)
CASERELATION -> CAUSALACTANT, THEME, LOCUS, SOURCE, GOAL.

(The (N) in the sixth rule is in a different typeface from the second 'N' so presumably they are different categories.)

These rules cannot be interpreted too strictly, since Simmons' examples do not conform to them exactly. Nevertheless, they are a handy expository device for conveying the possible combinations of categories. We will now try to convert them into our standard form of node-types and label-categories.

Types of nodes

Sentential: called 'S' by Simmons; corresponds to a tensed sentence.
Propositional: corresponds to a filled case-frame.
Modality: groups various information such as tense, mood, etc.
Verbal: corresponds to a verb (with its own case-frame).
Term: corresponds to a simple noun phrase or an embedded sentence.
Atomic: miscellaneous terminal nodes, given values by their labellings, could be subclassified further, into determiner, tense, etc., but this seems redundant, since the labelling of the nodes and their connected arcs will provide this information.
Connective: conjunctions, etc., with several attached arguments.

Organizational arc labels

MODALPART: for linking sentential node to modality node.
PROP: for linking sentential node to proposition node.
TOK: for linking propositions (filled case-frames) to their verbs.
TENSE, ASPECT, FORM, MOOD, MANNER, TIME: all these (which we will categorize as 'modal' labels) link modality node to a value.
CAUSALACTANT, THEME, LOCUS, SOURCE, GOAL, CAUSALACTANT′, THEME′, LOCUS′, SOURCE′, GOAL′: all these (which we will categorize as 'case' labels) link a propositional node to a term node. Including duplicates of each allows at most two of each case in each case-frame, which should be sufficient.
DET, ADJ, NBR, NOM, PREP, MODIF: all these (which we will categorize as 'modifier' labels) link a term node to another node.
ARG*i* (*i* = 1, 2, 3, etc.): arguments of relations.

Substantive node label categories

(Some of these may be organizational).
Connectives: OR, SINCE, BUT, etc.
Attributives: MOD, POSSESSIVE, ASSOC, etc.
Set-relations: SUP, SUB, EQ, etc.

Syntax rules

As observed in Section 3.2, we do not yet have a formal way of stating syntactic rules. The following informal descriptions serve merely to illustrate the information that might go into such rules.
A sentential node must have one outgoing MOD arc and one outgoing PROP arc.
A MOD arc can appear only between a sentential node and a modality node.
A PROP arc can appear only between a sentential node and a propositional node.
The only labels allowed on outgoing arcs from a modality node are labels of category MODAL.
Arcs with labels from category MODAL must start at a modality node and finish at an atomic node.
The only outgoing arcs from a propositional node are those labelled with a case-label or an arc labelled 'TOK'.
Each propositional node must have exactly one TOK link; this must run to a node of type verbal.
Each connective node must have exactly one TOK link; this must run to a node labelled with a substantive connective.

4.1.3 The LNR system

Norman and Rumelhart *et al.* (1975) describe a semantic network system which is intended to be used for various purposes, including language understanding and problem-solving. They use a 'case-frame' style of structure, in which each relation can have several arguments, and they distinguish several relations as being 'primitive'. Each node representing an instance of a relation (a 'secondary' node) is connected to the relation node (the 'primary' node), and each relation node (primary node) is linked to a structure defining it in terms of primitive relations. Each primitive primary node has attached to it a procedure which acts as its definition. Hence non-primitive relations gain their definition by being linked to a secondary node with a primitive relation linked to it, and primitive relations gain their definitions, in turn, from their attached procedures. (The written description states that all relations have procedural definitions, but it seems that the non-primitive relations have procedural definitions which simply rephrase in terms of the configuration of primitives (i.e., the patterns attached, in the diagrams, by ISWHEN links to the non-primitive nodes).) To incorporate all this in a uniform node-arc notation, we will use 'procedure' nodes, attached to the relevant primary node by a 'DEFINITION' link. The links available for connecting a secondary node to a primary node are referred to as 'TYPE' arcs, but this seems to be a category of labels (which includes 'ISA' and 'ACT'), rather than a label itself.

Inheritance is not very prominent in this system. 'Passing of properties' happens from a primary to a secondary node (i.e., down an ISA or other TYPE link), in that information such as selectional restrictions is available to each instance (secondary) of a relation (primary). Similarly, certain properties may be passed (presumably) from a defining configuration of primitives along an ISWHEN link.

'Case' labels—AGENT, OBJECT, LOC—are used to connect the argument nodes to a relational node. Similar remarks apply here as in the case of Simmons' argument-labels—it is not clear to what extent these labels are merely arbitrary mnemonics. The answer to this depends on the content of the various rules. There is further confusion about the import of these case-labels, since Norman and Rumelhart say that the labels refer to selectional restrictions (i.e., constraints on

what items may be arguments for the relation) as well as naming arguments, and some of the examples use two case-fillers with the same label (e.g., LOC). For the moment, we will include these as a special subset of the organizational arc labels.

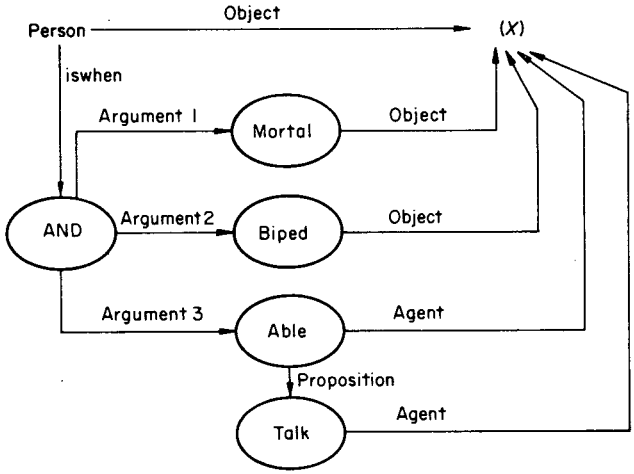

FIG. 4

Types of node

Primary: a relation.
Secondary: a relation-instance.
Atomic: a basic item (e.g., a number or character string).
Variable: used to connect argument positions in a defining configuration to corresponding positions in the relation being defined.
Procedure: carries procedural definition of a primitive relation.

Organizational arc labels

ISA: links a secondary to its primary (a label of category TYPE).
ACT: links a secondary to its primary (a label of category TYPE).
ISWHEN: links a primary to the root node of its defining configuration.
DEFINITION: links a primary to the procedural definition.
ARGUMENT*i* ($i=1$, 2, 3, etc.): links a relation-instance with several arguments (e.g., AND) to an argument node.
AGENT, OBJECT, etc.: case-labels (*see* above).

Substantive label categories

Primitive: a basic relation, marked on primary nodes (e.g., AND).
Non-primitive: a defined relation, marked on primary nodes (e.g., PERSON).
Procedural: a procedure, labelled on nodes of type procedure.
Basic: a data item.

All the arcs are stated to be two-way, although it is not clear what path-tracing processes use this fact.

4.1.4 Conceptual dependency

Schank *et al.* (1975a, b) have presented a general system of 'conceptual' representation which, despite its different terminology, seems to fall firmly into the category of being a semantic network. They emphasize that their system uses only a few 'primitive' elements for a wide range of meanings, but this covers only his representation of *actions*; for other meanings (e.g., referring to physical objects), conceptual dependency uses a wide range of substantive labels.

There is a fairly clear statement (Schank *et al.*, 1975a) of the elements of conceptual dependency notation, although the syntax rules given in that article merely describe the composition of the *LISP* expressions used in a particular implementation, rather than the abstract syntax of the knowledge representation scheme.

Types of nodes

PP: corresponds to a (physical) object.
ACT: marked with one of the primitive actions (or states?).
TIME: a time-structure.
CAUSAL: a connection between two related conceptualizations.
STATE: example of a state.
ACTION: example of an act.
STATE-CHANGE: example of a state-change.
BASIC: miscellaneous values (e.g., +2 in a state-change, *A* in a reference description).
CAUSALs, STATEs, ACTIONs and STATE-CHANGEs can be classed as 'BONDS' (to borrow terminology from Rieger—*see* Section 4.1.5).

Organizational node label categories

Causals: there are several kinds of 'causation' relation.
Special tags: miscellaneous markers.

Organizational arc labels

(These are exactly the MAINROLE and MODROLE lists of Schank *et al.*, 1975a.) For typographical convenience we have replaced the triple arrows of that description with single arrows (<-> and <-) in both the 'state' and 'causal' links).
ACTOR: the ACTOR which performed some act.
OBJECT: the object of one of the physical acts.
MOBJECT: object of mental act.
TO: recipient of act.
FROM: source of movement in act.
<=>: link to the primitive ACT involved.
<->: attribute and value of a state relation.
<->F: the initial state in a state change.
<->T: the terminal state in a state change.
SO: the object of a state or state change, where a PP is involved.
SC: the object of a state or state change, where a conceptualization is involved.
ANT: antecedent of a causal relation.
<-: Causal connection.
<-C: Possible causal connection.

TIME: links act, state or state change to its time-structure.
VAL: the value part of a state.
INC: for state changes, the amount of change along some scale.
PART: a body part relation.
REF: links an item to some information about its reference.
REL: links an item to a related item.

There are also three links which could be classed as 'TIMERELS'—SAME, BEFORE, AFTER. (Schank *et al.* do not list these as primitives, mainroles or modroles, but embed them in the syntax of the implementation; for uniformity and cleanliness, they should be put in a suitable category.)

Substantive label categories

Act: primitive actions; there are 11 (e.g., PTRANS, PROPEL, INGEST, etc; *see* Schank *et al.*, 1975a.)
State-names: e.g., JOY.
Things: e.g., John, LTM.
Time-values: e.g., T000034.
Data: e.g., integers.
There may be other categories.

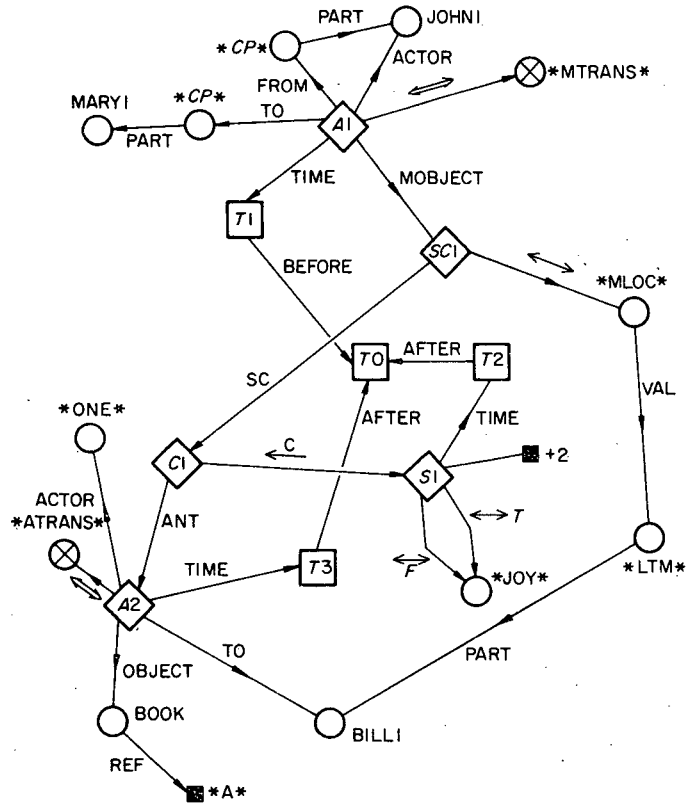

FIG. 5

There is, in the implementations of conceptual dependency, other miscellaneous information, but it is not always clear whether this is properly part of the conceptual dependency system, or simply part of the language-processing routines which operate to analyse or generate sentences (e.g., 'Thus ATRANS is a primitive ACT that requires an ACTOR, an OBJECT, and a RECIPIENT made up of a SOURCE and a GOAL. The SOURCE and GOAL must be animate and the object physical. This information is contained in the analyser together with a dictionary of what words combine into what configurations of ATRANS' (Schank *et al.*, 1975a: 311).

Figure 5 shows the example from (Schank *et al.*, 1975a) ('John told Mary that Bill wanted a book') in terms of the categories given above. Diamond shapes are 'bond' nodes, squares are time-nodes, circles are PP nodes, circles with crosses are act nodes, and black squares are basic nodes.

4.1.5 Rieger's version of conceptual dependency

Rieger (1975) describes the inference component of the MARGIE program, which uses conceptual dependency (CD) as its representational system. However, he does not give a clear implementation-independent characterization of the knowledge representation used for making inferences. The informal description makes it clear what the implementation structures are (*LISP* atoms and their property lists), but it is quite hard to work out what higher-level knowledge structures are supposed to be represented by these primitive programming constructs. The main problems (in understanding the system) are that the obvious interconnections (as given by the property lists) do not correspond directly to those of the usual CD diagrams, that miscellaneous substantive labels (e.g., WANT, SEX, etc.) appear without explanation in the examples, apparently with the same status as labels which appear in 'primitive' CD examples, and that there are almost no restrictions on what operations can be applied to what structures. We will attempt to impose some organization on this, but much of it will be arbitrary. For example, the program operates largely with lists of the form (<keyword> <list of arguments>) (e.g., (CAUSE C0061 C0054), (POSCHANGE #MARY #JOY), (WANT #BILL #C0045)) and attempting to subcategorize the various keywords used may be a false superstructure; perhaps the program makes no systematic distinction between keywords like WANT, LOC, POSCHANGE, TIME, BEFORE. It is worth attempting to categorize these labels, if only for comparison with other systems, but it may be the case that the actual program is a fairly uniform semantic net which makes no clear distinctions between any of its label types (least of all the CD primitive ACTs). This would then raise the question of how closely it was actually related to the CD representations proposed in articles by Schank.

In the Rieger system, certain relationships between items need to carry special annotations (such as when they were last used in an inference or what other links gave rise to the relationship). This provides one fairly firm criterion for classing certain items within the general framework, since Rieger states explicitly that these markings can occur only on two kinds of item ('bonds' and 'concepts'). This can lead to surprising results. For example, the relation 'IDENTIFIES' between two items (meaning that they are equivalent) can be annotated (with, for example, its 'reasons'), and hence must be a bond. On the other hand, the relationship of 'REASON' (which links an item to those which gave rise to it) cannot itself be annotated, and thus is of a different data type. Intuitively, both IDENTIFIES and REASON seem to be the same kind of 'meta-information'.

The following description seems to capture most of Rieger's system.

Types of node

Concept: These fall into two subtypes—general and token—which correspond to the 'definition' and 'instance' types. The concept token nodes seem to occur only for 'natural kinds'—i.e., classes like 'NOSE' and 'HAND'.
Bonds: These are items often drawn as arrow-headed links in CD diagrams—they are generalized in Rieger's system to be virtually any predicate or function applied to a set of arguments.
System values: These do not normally appear in CD diagrams, but are the basic atomic data items (e.g., system clock values).

Organizational arc labels

These fall into three classes:
(A) *Management links*—these connect a bond or concept node to information about it (e.g., its reasons for creation), thus maintaining various annotations for the inference rules.
RECENCY: connects to a value giving a (real) time (i.e., not a 'TIME' in the semantics of the matter being represented, but a time during the system's processing of the memory structures).
TOUCHED: connects to a real-time value.
SEARCH-TAG: another annotation used by the inference system.
(The above three can be marked on concept nodes—of both subtypes; the following links can be joined only to bond nodes.)
TRUTH: system's assignment of truth to this relationship.
STRENGTH: the validity (value between 0 and 1) of the relationship.
REASON: the other bond nodes which gave rise to this bond.
OFFSPRING: the other bond nodes to which this one gave rise (by inference).

Also, a general concept node has a link XFORM connecting it to 'a template specifying the conceptual dependency structure which will express in CD format any memory structure which involves' it.
(B) *Slot labels*—these do not appear explicitly in Rieger's description, since he uses position as a notational convention to link a relation-instance to its arguments. We have assimilated these into our general definition by the usual technique of labelling argument places with names.
ACTOR, OBJECT, FROM, TO: These are the argument places for primitive ACTs, as commonly shown in CD diagrams.
BONDPRED: This links a bond (i.e., an instance of an action, state, or causal) to the particular predicate involved (in CD, the predicate symbol is normally drawn in amongst the arguments).
TIME: The status of this link (from a bond to a description of its time of occurrence) is confused in the CD articles. It seems to fit here.
TS: Similarly, there is often a link to the start-time of an item.
ARG*i* ($i = 1, 2, 3$, etc.): This is necessitated by our use of explicit labelling of arguments for arbitrary predicates. Rieger uses positional notation (on property-list entries) for the arguments of predicates like 'CANNOT'.
(C) *Semantic links*—These are miscellaneous connectors which seem to carry a wide variety of imports.
ISA: connects a concept node to another concept node.
NAME: links an item to its name.
BEFORE: links a time-description to another time-description.
AFTER: links a time-description to another time-description.

Substantive node label categories

There are (at least) six classes:
(A) *Primitive ACTs*—ATRANS, PTRANS, MTRANS, etc.
The above 'primitives' can be classed into categories (e.g., MTRANS is an ACTIONPRED), and this is represented explicitly in the network using ISA links. Thus we need some substantive node labels for concept nodes.
(B) *Conceptual categories*—ACTIONPRED, CAUSAL.
(C) *Causals*—There are several types of causation, including at least these:

CAUSE
CAN-CAUSE
ENABLE-CAUSE
RESULT-CAUSE

These are marked on general concept nodes which define relations.
(D) *States*—A state is normally described (in conceptual dependency articles) as being a connection between an item and an attribute-value pair, but Rieger's examples do not seem to include clear illustrations of this general structure. There are various predicates, however, which might plausibly be called 'states' (e.g., physical contact, location, 'part-of'), and these take arbitrary sets of arguments.

Also, Rieger uses illustrative examples full of labels like 'COLOUR', 'RESIDENCE', 'WANT', but it is far from clear how these fit into his system. He appears to be using them as general labels for network links without regard for conceptual dependency. They might fit under the all-purpose heading of 'state', emphasizing that it is a very broad category.
(E) *Natural kinds*—These are 'classes' of real world objects. Clearly the set is extremely large. The fact that these are used in Rieger's system is demonstrated by the presence of items like 'NECK' and 'NOSE' in his examples. Presumably these are labelled on general concept nodes, and then concept token nodes are connected to them by ISA links.
(F) *Logical connectives*—There are several items in Rieger's examples of the form (CANNOT . . .) and Schank's diagrams sometimes have 'AND' symbols. This seems to require a set of logical predicates.

CANNOT
AND
NOT

Operations on these structures

The main flow of inference is driven by the creation of certain types of data item in the memory. The nodes classed above as bonds and token concepts are the relevant ones; when one of these is created, it may set off inference rules. The system works continually through those inference rules (which are linked to the newly created items), executing these rules. Each rule, or 'inference molecule', is an arbitrary *LISP* program which can test any part of the memory structure and can build or delete arbitrary structure.

4.2 Predicate logic-based systems

The frameworks of Hendrix *et al.*, and of McSkimin and Minker, combine the hierarchical classification found in uniform semantic networks with the formal operations of the first-order predicate calculus, thereby using the semantic classes as a way to improve the efficiency of a well-defined, logically complete system.

These systems can be seen in two components:

10. (a) A predicate calculus unit. This stores logical statements (either in conventional form or in clausal form) and operates on them in ways consistent with their interpretation within traditional logic (e.g., resolution).
    (b) A semantic network. This is primarily a classificatory mechanism, in which classes and subclasses are linked hierarchically, and elements are linked to the classes to which they belong. The operations in the logical unit can use this information to improve the process of searching through the stored statements, in various ways. The classification of the domain into different 'sets' or 'types' of entity is similar to using a sorted logic (Enderton, 1972).

Janas and Schwind also describe a network system which is closely related to the ideas of conventional logic in its interpretation.

4.2.1 Semantic graphs

McSkimin and Minker (1979) outline a system which is interesting in a number of ways. They combine the mathematically precise mechanisms of predicate logic with a system of semantic classification, in such a way that the latter guides the operation of the logic-based clause manipulations. The system as a whole is, they state, equivalent to the first-order predicate calculus in power. The main data structures which their system uses are of two varieties—logic clauses and semantic classifications, the latter being represented in a 'semantic graph'. This hierarchical taxonomy is used for a complex form of 'type-checking' by the theorem prover, thus allowing an (arbitrarily detailed) amount of domain-specific knowledge to be incorporated into the logical system. The simplicity of structure of the semantic graph, and the limited use to which it is put, allows a very compact and efficient implementation, which McSkimin and Minker describe. (Roughly, the graph-theoretic notion of an incidence matrix is generalized so that entries contain arc-labels rather than merely 0 or 1.)

It would be fairly straightforward to represent logical clauses in network notation, but this is not of particular interest (except to show the universality of the network notation—*see* Section 3.10 above). The 'semantic graph', which represents a hierarchical taxonomy of items into sets, is more obviously within the semantic net tradition. It can be described using the following categories.

Types of node

Set: can be connected to any other node by any arc type.
Primitive set: can be connected up to a set node by a subset arc. (The primitive sets are regarded as the bottom of the hierarchy, and are assumed to be all disjoint.)

Organizational arc labels

Subset: joins set (or primitive set) to a set node, representing set inclusion.
Equal: joins two set nodes, indicating they are alternative descriptions of the same set.
Disjoint: joins two set nodes, indicating that they do not overlap.

Substantive node label categories

Set-name: marked on any node.

4.2.2 Predicate logic nets

Schubert *et al.* (1979) describe a semantic net system which is closely related to conventional predicate logic (*see also* Schubert, 1976). Their system has nodes corresponding to propositions, connectives, variables and functions, allowing any logical expressions (including quantified ones) to be written in node-and-arc form.

They discuss inheritance, and propose that the usual ISA hierarchy of concepts be represented by universally quantified statements containing nested sets of implications. For example, implications about OWLS would be contained within the scope of the universal statements about BIRDS. Although (1979: 153) they talk of the need to 'move up one step in the generalization hierarchy to the elephant node' when answering a question about a particular elephant, they do not describe exactly how the levels in the hierarchy are connected.

They also provide an indexing mechanism for propositions, in which all statements are classified into a 'topic hierarchy', covering such categories as 'generalizations', 'social', 'appearance', 'kinship'. Every pair of a proposition and an argument fits under (at least) one of these headings. This structure is separate from the main semantic net, although it helps to organize the processing of the main net.

Types of nodes

Operator: subtypes—relation, function, logical.
Result: for instances of items held on operator nodes.
Constant: fixed items (including Skolem functions) in domain of discourse.
Variable: universally quantified variables.

Organizational node label categories

Relational names: includes the function names (e.g., 'C' the credibility operator) and logical operators and connectives (e.g., =>, NOT).

Organizational arc labels

A, B, C, etc.: arbitrary argument labels.
PRED: connects a result node to a relation-operator node.
FUNC: connects a result node to a function-operator node.
OP: connects a result node to a logical-operator node.
Scope1: connects a variable node to a Skolem function depending on it.
Scope2: connects a proposition node (i.e., a result node connected by a PRED link to a relation node) to a quantified variable which lies within its scope.

Substantive node label categories

Constant names: e.g., 'John'.
Relational names: includes predicate names ('likes') and function names (e.g., 'age').
Figure 6 shows one of Schubert's examples, expanded to show all nodes and arcs, depicting the meaning of 'There is always someone there' (1976: 177). A broken circle is a universally quantified variable, a solid circle is a constant, ovals represent result nodes and rectangles are operator nodes.

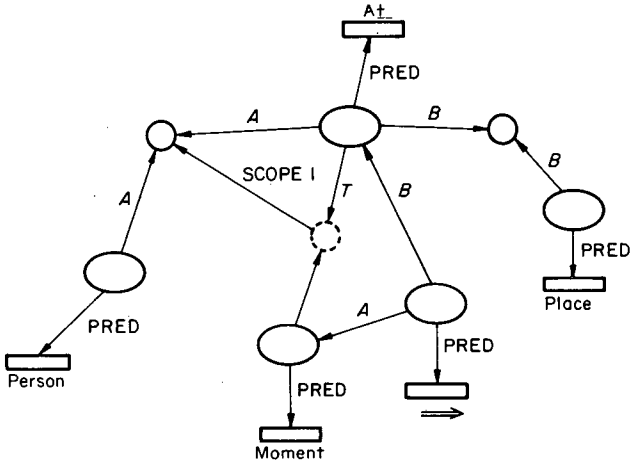

FIG. 6

4.2.3 Partitioned semantic networks

The system described by Fikes and Hendrix (1977) and by Hendrix (1979) is interesting, since it contains a range of advanced features (*see* Figure 7). The networks are subdivided into 'spaces', where a space is an arbitrary network, and spaces may overlap or be contained within each other. Some of the nodes in the network are classed as 'supernodes', where a supernode stands for an entire space. Hence an arc can connect an ordinary node to some partition (space), thus indicating a relation between that node and what is represented by the entire subgraph. Predicate logic statements are represented in the network by the use of nodes labelled as 'implications', and by various conventions regarding variables.

Partitions (spaces) are used in various ways:

11. (a) The logical operators have spaces as their 'scope'.
    (b) A space can simply represent a substructure to be treated as a chunk (e.g., during the gradual building up of a semantic structure in sentence analysis).
    (c) Spaces can represent 'contexts'; that is, subsets of the net which are accessible under certain conditions. For this purpose, spaces can be linked into 'vistas', which are ordered lists of spaces which together make up the context.
    (d) A space can group together all the things 'believed' or 'known' by some entity in the world, thus giving a notation for describing belief systems.

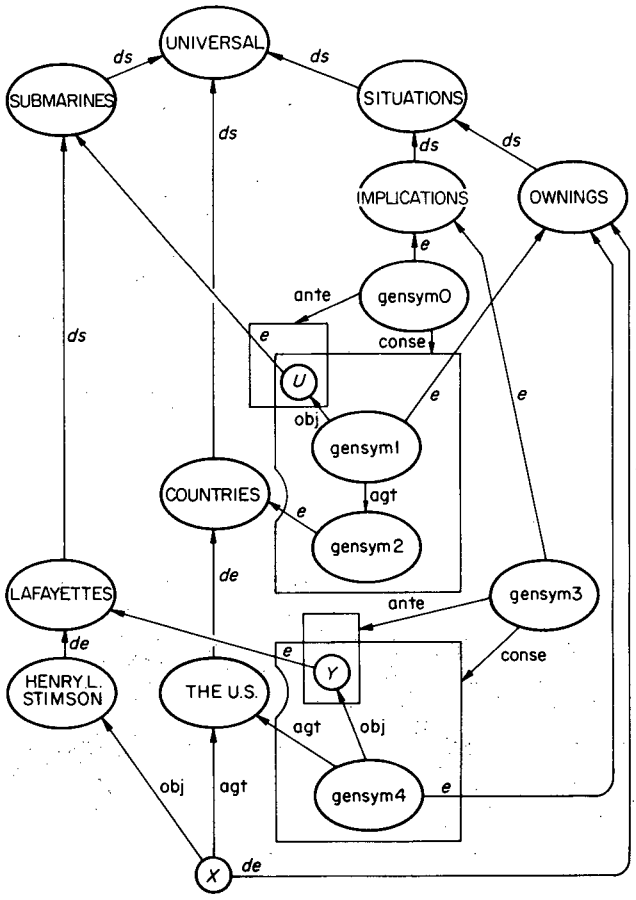

FIG. 7

The logical aspect of the net is encoded by having set-element nodes which are 'implications', 'disjunctions', 'negations', etc., with links to the supernodes which they modify. 'Implications' are linked (by argument names 'ante' and 'conse') to two overlapping spaces; any node in the overlap is the bound variable (universally quantified). 'Disjunctions' are connected (by 'de' links—*see* below) to any number of spaces, and 'negations' are connected (by a single 'e' arc) to a space whose contents are all to be negated. Conjunctions are not given a special node, since a space is regarded as conveying the conjunction of all the propositions represented in it. Similarly, all nodes are assumed, where relevant, to be existentially quantified, if they are not bound in an implication. Fikes and Hendrix outline some procedures which operate on these structures to give the effect of conventional theorem-proving.

It is important to notice the distinction between 'spaces' (subgraphs), 'vistas' (ordered lists of subgraphs) and 'supernodes' (items in the network representing spaces). Of the four uses of spaces given in (11) above, only (a) and (d) require connections inside the network to whole spaces (i.e., only these use 'supernodes'). The other two cases are situations where some external mechanism (the linguistic

rules, or a context handler) needs to refer to some subgraph of the net. This is emphasized by the fact that in Hendrix' illustrative diagrams (for context-vistas and for sentence-analysis) the connections between the partitions are unlabelled. That is, they are qualitatively different from the semantic network arcs, being part of an external system which operates on semantic nets.

This suggests that it might be worth investigating the use of 'spaces' by the logical part of the system, in case 'supernodes' are unnecessary here also. If that were the case, the logical aspects of the representation might be separable into another component which operates on structures within the net of set relationships (cf. Section 4.2.1). Such an elimination does not appear to be the case, although many of the simpler illustrative examples (in Hendrix, 1979) encourage the idea. The logical connective nodes are linked by 'e' ('element of') nodes, to general nodes for those connectives, which in turn are linked to broader classes of item (e.g., 'situations'). This does not in itself show that the logical structure is inextricable from the network, since it is not clear that all these links (e.g., the 's' link, for 'subset of', to 'situations') are actually used in a way compatible with the interpretation of the links elsewhere. For example, a negated space is linked to the 'Negations' node by an 'e' link—is the meaning of this arc ('element-of') used here, or is it just an arbitrary way of connecting a space to a negation node? The attachment of set-relationship arcs to logical nodes may be a (slightly misleading) notational device, which does not necessarily indicate an entanglement of the two aspects of the net (set hierarchy and logic).

The next relevant point (to the separation of the logical information) is that of nested statements. The 'antecedent' of an implication may itself be (or contain) an implication, a disjunction or a negation, for example. Since the arguments of the logical operators are spaces, the inner constituents now have network links inside them pointing to further spaces. Even this does not enmesh the logical operators with the semantic net as much as it might seem. It would be quite feasible to have two components, a semantic net (embodying predicate-argument structures and set relationships) and a logic system. The only pointers across from the latter to the former would be where (positive) literals appeared in logical statements. All the other structures (nesting, application of connectives, etc.) could occur in the logic system, with no pointers back from the net to the logical operators.

This suggestion relies on the tree-shaped structure of a logical statement—spaces are necessary only where several nodes have to be indicated rather than the root of a tree. In partitioned semantic networks, the spaces also serve to indicate the scope of variables; in particular, any node occurring in both the antecedent and the consequent is the bound variable of the implication. This information (i.e., the sharing of a node) is already present without 'spaces'—the node is present in both expressions since it can be reached from the root node of each. The implementation using supernodes (described in Hendrix, 1979) may be an efficient way of handling this information, but that does not mean that it is an essential part of the formalism.

Types of nodes

Set-element: an element of a set.
Set: a set (either of entities in the subject matter or of nodes within the network).
Basic: terminal nodes.
Supernodes: *see* discussion above.

Organizational arc labels

Set-relationships: these provide the usual subset hierarchy, using four labels—'e' (linking set-element to set), 's' (linking set to set), 'de' (linking a set-element to a set, where all the 'de' arcs coming into a set node come from nodes 'known to represent distinct objects'), 'ds' (linking set to set, where all the 'ds' arcs into a set node come from sets 'known to be disjoint').
Argument-names: there are many argument names, but it seems that they could be uniformly renamed as 'ARG1', 'ARG2', etc., without disrupting the system. They link set-elements to other nodes.
Delineation: the only example, in the articles cited, of an arc which is neither set-relational nor an argument-name is the 'delineation' link, which links a set-node to a node which describes elements of that set in more detail (cf. Norman and Rumelhart's 'ISWHEN' in Section 4.1.3, and Brachman's 'structural description' in Section 4.3.1).

Substantive node label categories

Set-names: marked on set nodes.

4.2.4 Extensional semantic networks

Janas and Schwind (1979) describe a network system which is unusual in various ways. It is similar in some ways to Norman and Rumelhart's semantic nets in that it represents an instance of a relation by a case-frame configuration containing a node labelled with a relation (e.g., 'love') connected by case-like links (e.g., 'SUBJ', 'OBJ') to nodes labelled with individual names (e.g., 'John'). It is reminiscent of Schubert's logic network, since it has nodes labelled with quantifiers which are linked to the nodes which they bind. As in partitioned semantic nets, an arc may connect to a whole subgraph, in order to express the scope of a quantification. For such a self-contained subnet, one node within it is designated the 'entry' node.

Janas and Schwind provide a careful mathematical definition of a 'value' function to allow these 'supernodes' to be included, but this function is essentially the node-labelling function, so it is possible to incorporate this into our general definition. That is, one of these 'supernodes' can be regarded as a node which is labelled with an entire subgraph. Such nodes will have no outgoing arcs (all Janas and Schwind's networks are tree-like, and so have 'terminal nodes').

The 'entry node' for such a subnet is used to indicate which part of the substructure is being referred to by any arcs which point to the subnet as a whole. For example, the sentence 'The milk the mother is drinking is hot' has the diagram in Figure 8, where rectangular boxes indicate nodes and the contents of the boxes indicate their values; hence a semantic network is a single node, whose value is the net structure.

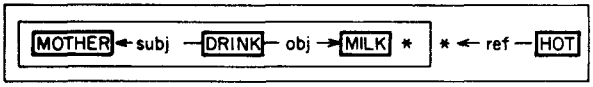

FIG. 8

(We are not concerned here with the linguistic or logical adequacy of the notation.)
The idea of a 'head' for a structure may be linguistically useful, but it does not

seem necessary to introduce the concept of an 'entry node' into our general definition of a semantic net. Instead, we can define a class of 'entry-marked semantic nets', and use this class to define the range of the labelling function for an ESN ('extensional semantic network').

Although the subgraphs labelled on to nodes provide a useful scoping device for quantifiers (as in 'partitioned semantic networks'), the use of entry nodes becomes inconvenient when Janas and Schwind introduce a notation for logical conjuncts and disjuncts. They therefore amend their definition to allow subgraphs *without* entry nodes to be labelled on to nodes. This generalization is useful from the point of view of our uniform general definition, as it brings ESNs closer to 'partitioned semantic networks', in which entry nodes are not used.

Logical formulae are expressible in ESNs by the use of negation nodes, universal quantifier nodes and existential quantifier nodes, as well as conjuncts and disjuncts.

The word 'extensional' occurs in the title of ESNs because Janas and Schwind include a definition of how the (intensional) structure in their nets can refer or apply to sets of items (in a way similar to the Tarski semantics for first-order logic). Thus, if we assume that truth-conditions in terms of objects and relations constitute an 'interpretation' for a knowledge representation system, then Janas and Schwind have given a semantic interpretation for ESNs (or at least a partial one—some aspects are not explained).

Hence we can describe ESNs as 'multilevel semantic networks' (*see* Appendix) with the categories given below. The distinction between substantive and organizational links is unclear in their framework, since they comment:

> Many linguists have argued . . . that meaning is closely related to subject areas . . . Consequently, a certain subject area is covered by a set of concepts and a set of relations defined on these concepts, and these sets are usually different for different areas. For example, we do not need a verb–subject relation for covering mathematics (Janas and Schwind, 1979: 270).

Nevertheless, the relations they propose seem to be organizational—SUP (concept superordinate to another), REF (concept may/does refer to another), SYN (synonymous), ANT (antonymous), etc. Also, their illustrative examples (which use these relations) are drawn from various subject areas. This question is further confused by the fact that many of their relations and concepts seem to be based on English grammatical inter-constituent relations (e.g., prepositional adjunct, object). It seems best to interpret the above quotation as meaning that certain applications of their network theory may not need to use some of the vocabulary of organizational links, rather than taking it to mean that different subject areas use completely different link-types. With this assumption, ESNs have no substantive arc labels—all substantive labels occur on nodes.

Types of nodes

Atomic: three subtypes—terminal (basic atoms), relation (names of verb-like relations such as 'drink', or adjectival relations such as 'hot'), and logical (bearing quantifiers, connectives, etc).
Supernodes: labelled with subnets.

Organizational node labels

EXIST: quantifier, on a logical node.
FORALL: quantifier, on a logical node.
NOT: negation operator, on a logical node.
AND: conjunction, on a logical node.
OR: disjunction, on a logical node.

Organizational arc labels

Janas and Schwind include some unlabelled arcs in their diagrams, connecting logical nodes to the items they affect, although neither their definition nor ours permits unlabelled arcs. We will include extra arc labels to cover these cases.
SUBJ: verb–subject relation.
OBJ: verb–object relation.
PREP: prepositional adjunct.
OWN: general association between concepts.
REF: concept applies to another.
VAR: connects quantifier to its bound node.
ARG: connects negation node to the item it covers.
SUPO: connects concept node to a superordinate one (cf. ISA).
ANT: connects two antonymous concepts.
SYN: connects two synonymous concepts.

Substantive label categories

These node-labels fall into two broad classes:
Atomic: (marked on Atomic nodes), with sub-classes—verbal, adverbial, relational, sentential, adjectival (marked on relation nodes), and nominal (marked on terminal nodes).
Subnet: (marked on supernodes).

4.3 Frame systems

The systems of Brachman, of Fahlman, and of Roberts and Goldstein exemplify representational schemes in which information is more clustered than in uniform semantic networks. The typical characteristics are:

12. (a) As in uniform semantic nets, there is a distinction between 'generic' definitions and 'instances' of these definitions.
    (b) The amount of information associated with the generic items (and inherited by the instances) is large.
    (c) The simple 'argument places' used in uniform semantic nets are replaced by 'slots' which contain large amounts of information about the aspect of the concept being described. Usually there is a 'default' value, a 'restriction' on what may be associated with the slot, and 'procedures' which have to be executed under certain conditions.

These systems can be seen as derived from the 'frames' of Minsky (1975).

4.3.1 NETL

Fahlman (1979) outlines a semantic net scheme which is designed to make use of parallel hardware. The essential idea is that many of the computationally intensive parts of net-searching (e.g., finding intersections between sets) can be done very simply and quickly if an appropriate (special-purpose) machine is used. His network representation is well within the mainstream of knowledge representation, but has certain special features to take advantage of the special parallel operations (which use 'tagging' operations as described in Section 4.1.1. above). Many of the details of his net constructs have not been fully developed, and it is sometimes hard to separate the abstract definitions from descriptions of implementation techniques.

One of the slightly confusing aspects of Fahlman's account is his use of the word 'link'. A 'link' in NETL is not a simple 'arc' in the sense adopted in this article, but is a complex structure in its own right, which can have further connections to other items in the net. It is, in fact, much more like a 'node' in our terminology. The word 'link' has been used because these items represent relations between other items (i.e., they are like Norman and Rumelhart's 'secondary nodes'), and this can be seen as a kind of 'connection'. The confusion is increased by the fact that Fahlman also has simple, atomic connections, called 'wires', which are much more like the 'arcs' of our definition, and he states that certain semantic connections may sometimes be represented by a 'link' (i.e., a relational node) and sometimes by a 'wire'.

Each 'element' (i.e., 'node' or 'link') can have 'modifiers' which are regarded as flag-bits giving further details of the type of element. These are what were referred to as 'type features' in Section 3.5 above, since several of them may be present, and they subcategorize the node for the benefit of the interpreter (i.e., they are 'organizational' rather than 'substantive' information). It is difficult to establish exactly what are the constraints between these features, so we have not included a full specification of these here. Fahlman regards the node types as having certain basic types (denoted by mnemonics starting with '*'), each subcategorized by modifiers (denoted by mnemonics starting with '**'), but the possible combinations of modifiers within each basic type are not explicitly listed.

In addition to the modifiers, there are 'markers' on each element, which the fast parallel hardware (which is spread throughout the net) can use to propagate information across the net. Although these would be stored directly on the element in an implementation, they are (at an implementation-independent level) a form of 'tagging' as discussed in Section 4.1.1, and so need not be included in the specification of the net structures. In this particular case, such inclusion would be possible, since the scheme uses a small fixed set of markers (M1, M2, etc.), which could be regarded as a predefined set of 'tag-sets' into which nodes may be inserted or removed.

Types of node

*INDV: Individual entity.
*TYPE: Typical member of set.
*MAP: Copied version of some inherited role.
*TMAP: Copied version of an inherited *TYPE node.
*IST: Instance of relation or predicate.
*OTHER: arbitrarily-chosen individual of a set, different from the *TYPE node for that set.
*EVERY: Universal statement about a set. (Fahlman describes this as a '*TYPE

node modified', so perhaps this is a type-feature (modifier, not a basic type).
*INT: Represents the intersection of two sets ('Like an *EVERY node . . .' according to Fahlman's specification).
(The above node-types are what Fahlman calls 'nodes'; the next eight are those which he calls 'links').
*VC: inheritance link.
*EQ: two nodes connected to this are equivalent.
*CANCEL: explicit deletion marker from a structure to the 'deleted' item.
*CANVC: explicit statement of non-inheritance between two items.
*SPLIT: used to link items known to be distinct to their containing set.
*EXFOR: asserts that for every copy of one item, there must be a version of the other.
*EXIN: asserts that every copy of the first item intrinsically contains a version of the other.
*SCOPE: one item is valid within the scope of another.
Sub-types (Modifiers):
**PART: marks part of some structure; on *TYPE or *TMAP nodes.
**EXIN: marks whether an item is an intrinsic (defining) part of its containing structure, or merely an attached (related) item; on *INDV nodes.
**SPLIT: marks whether item is distinct from others similarly connected to the *TYPE node; on *INDV nodes.
**SPEC: marks nodes which are in 'specification' part of *EVERY structure; on link-nodes, *IST and *INDV nodes.
**EXTERN: marks an externally-expressible (printable) item; on *INDV nodes.
**RSPLIT: marks whether this item can fulfil the same role in several items or only in one; on *INDV, *EXIN, *EXFOR nodes.
**UNK: marks unknown item; on *IST nodes.
**NOT: negation marker; on *IST and *EVERY nodes.
**LIKE: marks a connection as not being inheritance, but a form of similarity; on *VC nodes.

Organizational arc labels

These are what Fahlman calls 'wires'. Some pieces of information may be represented by a 'link-node' (e.g., *SPLIT) or by one of the 'wires' together with a modifier on one of the nodes thus connected (e.g., 'existence wire' and **SPLIT modifier). Hence the appearance of similar ideas in different sections.
PARENT: connects a node to another node defining what kind of entity the former represents.
SCOPE: connects node to description of its area of validity.
SPEC: connects node to the *EVERY node in which it is a specification.
A and B: these connect to any two related items; their meaning depends on the node-types involved.

4.3.2 Structured inheritance networks

Brachman (1977, 1978, 1979) sets out a very detailed, general and rigorous system of semantic nets, with the aim of eliminating the confusion caused by the multiplicity of ill-defined and inconsistent proposals previously made on the topic. To a large extent, this purpose is achieved. Although Brachman's 'structured inheritance networks' do not solve every problem of knowledge representation, they serve as an

extremely useful notation for representing many distinctions which have been overlooked elsewhere.

The main mechanism is, as in Fahlman's NETL (*see* Section 4.3.1), inheritance from a generic 'concept' node of a complex interconnected set of information, including a set of 'roles' which may be filled in that concept. Each concept is linked to a 'structural description' (cf. the ISWHEN of Norman and Rumelhart), specifying in detail what the meaning of that concept is, in terms of connections between its roles, etc. Brachman uses a large number of links to express the various subtleties between different forms of inheritance or 'instantiation', and distinguishes several variations on the notion of an 'event', depending on how it is being described. The various nodes used in these event-descriptions are still part of the general scheme of 'concepts', and are not classed as a new type of item. Logical connectives (e.g., AND, OR) are also incorporated as concepts (seemingly the only built-in (organizational) concepts in the system), with the arguments being represented as role-fillers in a uniform fashion.

Types of node

Concept—has subtypes: generic, individual, paraindividual, logical. A paraindividual is a version of a generic concept whose roles are linked to those of another concept (Brachman, 1979: 38–89).
Role—a cluster of information about a part or attribute of a concept.
Structural description—the root of a subnet describing a concept.
Basic—any atomic symbol (e.g., an integer or character string). Can hold the modality tag for a role.

Organizational node labels

Logicals: AND, OR, etc.
Modalities: Optional, etc.

Organizational arc labels

Structure: from concept to structural-description.
RoleD: from concept to role (also known as 'Dattrs').
The next five all link role nodes to details of the role (facets):
Modality: optional, derived, etc.
Value-restriction: constraint on filler.
Val: the value.
Role: name of role, or other role which this represents.
Number: how many in the set of fillers.
Paraindividuates: from paraindividual to the generic concept of which it is an instance.
Individuates: from individual to generic concept.
Satisfies: from role to the role of which it is an instance.
Corefvalue: from role in structural description to role for which it stands.
SDfacet: from structural description to one of its parts.
Corefsatisfies: from role in structural description to corresponding role in generic concept being defined.
Dinsts: Individual (or paraindividual) to a role in a generic.
Dsuperc: from sub-concept to super-concept.
Diffs: from sub-concept to roles which differentiate roles in super-concept.

Dmods: from sub-concept to roles which add specification to roles in super-concept.
Dbrotherc: from sub-concept to 'similar' sub-concept.
Dfactive: between event-node and 'fact-of-happening' node.
Dresult: from event-node and thing-produced node.
Dactivity-process: from event-node to description as a process.
Dactivity-complaction: from event-node to description as a completed action.
Dgen: special kind of generic event.
Drole: concept derived from a role.

Substantive node label categories

There are only two varieties of item needed:

1. Concept-names.
2. Data (character-strings, integers, etc.).

Figure 9 shows a sample SI-net, using the conventions that quoted items are data (on basic nodes), ovals are concept nodes, squares are role nodes, and black squares are role nodes within an individual concept.

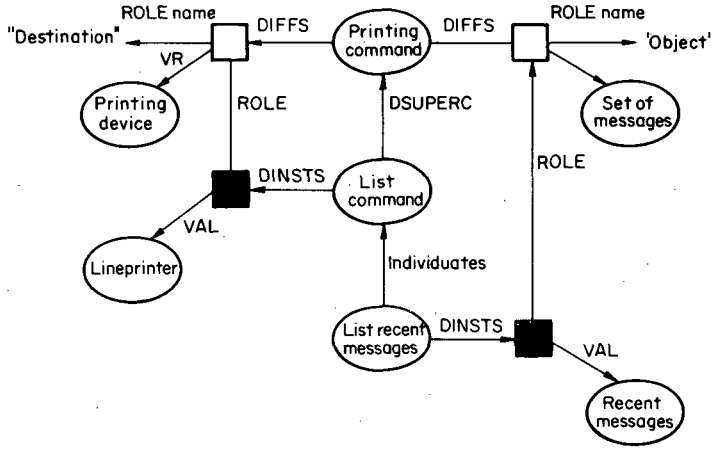

FIG. 9

4.3.3 FRL

The *FRL* system ('Frames Representation Language') of Roberts and Goldstein (1977) can be seen as a language which operates on a form of semantic net, although it is not described in normal network terminology.

The main structure in *FRL* is a 'frame' (cf. Minsky, 1975), which contains certain 'slots', each with a number of 'facets'. Although these structures can be seen as a form of 'attribute-value' lists, as used in *LISP* programs over recent years (and are implemented in that way) they can also be seen as configurations of nodes. The structures in *FRL* are similar in some ways to the 'SI-networks' of Brachman (*see* Section 4.3.2).

A 'frame' represents either a general concept or a particular instance of a concept.

Its 'slots' contain further information about it (e.g., subparts or attributes), with each 'facet' of a slot containing some specific extra information about the item filling that slot (e.g., a restriction on what items may fill it, or a default value to be used if none is explicitly entered). Frames normally have a slot labelled 'AKO' ('A Kind Of') which contains a pointer to another (generic) frame, thus incorporating the traditional 'ISA-link'. The operations which act on the frames take special account of this slot, in order to 'inherit' information from frames further up the chain of AKO links. In a way, the AKO slot is not a slot in the same sense that other slots in the frame are, since it appears on all frames and does not represent a part or attribute of the particular item described. The fact that a frame is implemented as a *LISP* list, and the AKO entry is on this list, blurs the distinction between the special meaning of the AKO link and the other slots. There is also a special slot 'INSTANCES', which is the inverse of AKO.

There is some further structure within the entries on the 'facets' of the slots. Each 'datum' (i.e., facet entry) can have attached to it a list of 'comments', where each comment consists of some label or keyword followed by a list of 'messages' (arbitrary strings). There are a fixed set of comment-keywords that the *FRL* system can use, although presumably the *FRL* programmer could define further comment-types, since *FRL* is embedded in *LISP*. Similarly, there are six standard facet-names used by the *FRL* interpreter, but a *LISP* programmer using *FRL* could probably extend this set. To do so would be to extend *FRL*, rather than merely to use it, since it would entail changing the interpreter (which implicitly defines the language).

Values (e.g., integers, atoms) appear in the system solely as node-labels, and there is no need for a special class of nodes to support them. These primitive data items can appear, for example, as the labels on 'datum' nodes.

It seems necessary to allow substantive categories as labels on arcs, in order to capture the structure of *FRL* slots, since the slot-names are used within a frame to distinguish the different slots, and also have global significance (in that some of the inheritance procedures search up the inheritance links to find a slot of a given name). If we wished to eliminate the use of substantive arc labels, we would have to use a (less obvious) representation in which each slot was linked to a facet holding its slot-name (cf. Section 4.3.2 above).

FRL allows 'procedural attachment', since some of the 'facets' (e.g., IFADDED) are procedures to be executed under particular circumstances. We can incorporate this by allowing the labels for 'datum' nodes to be of type 'procedure', but it is not possible (within the limited framework of this article) to give any more precise definition.

Types of node

Frame: these can be of two sub-types, generic or individual.
Slot: this indicates a cluster of information attached to a frame.
Facet: one of these is a particular part of the information in a slot.
Datum: an entry in a facet.
Message: part of a 'comment', attached to a datum.

Organizational arc labels

Although slot names will in general be substantive, there are two which are built in to the system:
AKO: joins a frame to a generic frame.

INSTANCE: joins a frame to its (list of ?) instances.
There are six facet-names. These link a 'slot' node to a 'facet' node, and the following are the informal meanings of the items to which they connect:
VALUE: the item in the slot.
DEFAULT: the value to be used if none is actually in the slot.
IFADDED: a procedure to be executed when a value is inserted in the slot (i.e., a VALUE link is made to some item).
IFREMOVED: a procedure to be executed when the VALUE link is removed.
IFNEEDED: a procedure to be executed in order to compute an item to fill the slot.
REQUIRE: a predicate which delimits the permitted VALUE items.
There are three built-in comment types, which link a 'datum' node to a 'message' node. They are used by miscellaneous *FRL* system routines.

IN
FINHERIT
TYPE

Substantive node label categories

Frame-names (marked on frame nodes).
Data (marked on datum nodes).
Procedures (marked on datum nodes).
Strings (marked on message nodes).

Substantive arc categories

Slot-names—connect frame nodes to slot nodes.
Figure 10 shows one of Roberts and Goldstein's examples drawn as a net; the original is given in (13).

(13)
(THING
(INSTANCE (IF-ADDED ((ADD-INSTANCE)))
(IF-REMOVED ((REMOVE-INSTANCE))))
(AKO (IF-ADDED ((ADD-AKO))
(IF-REMOVED ((REMOVE-AKO)))))

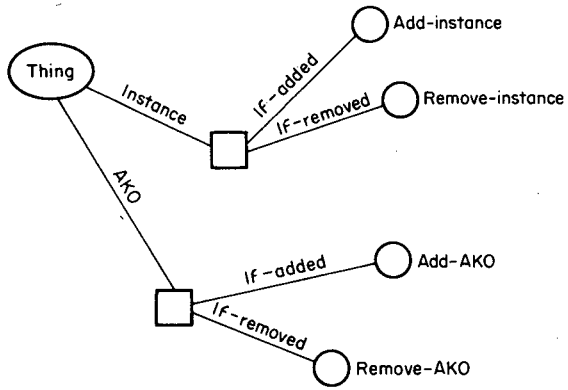

FIG. 10

5. FURTHER DEVELOPMENT

Despite the length and breadth of this paper, we have laid the shallowest of foundations for a precise, rigorous approach to defining semantic networks. The main areas for elaboration are:

Syntactic rules. As suggested in Section 3.2, there is a need for some way of specifying the form of semantic networks. Some augmented version of the grammars discussed by Shaw (1970) might be suitable, for example.

Operations. It is not very useful to list labels, node-types, etc., without any reference to the operations which are defined on them. We have had to follow this custom, since the published accounts of semantic nets generally rely heavily on the reader's intuitions in this area. For any particular semantic net formalism, the operations could be stated in any suitably precise way, for example by defining the net as an 'abstract data type' (Wegner, 1980).

Procedures. As well as defining the basic network operations, we need (at least for those systems using procedural attachment) some notion of a 'procedure' which is a data-item, within the network, which can give rise to some sequence of operations being applied to the net.

Interpretation. It is rare for the 'meaning' of a semantic net to be given in a written description (*see* comments in 3.8, 3.10 above, and in Hayes, 1977, McDermott, 1978). Janas and Schwind (1979) give such an interpretation, and some authors rely on the similarity of their notation to predicate logic, but most use totally uninterpreted notations.

One possible obstacle to developing these aspects of semantic nets is the fact that semantic networks fulfil different purposes for different authors. Quillian originally proposed a network as a natural data structure for the kind of processing he was using (spreading activation). Thus his networks were not a formal system of representation comparable to, say, predicate logic. The visually appealing graph structure was then adopted by others (e.g., Schank, Norman and Rumelhart) who were not using spreading activation models, but who saw the relational structure expressed in the net itself as a form of knowledge representation. In some later systems, the network takes on a less central role. For McSkimin and Minker, and for Hendrix, it describes a hierarchy of sets which can be used to guide logical deduction. For Brachman, it provides an abstract level for describing how a knowledge representation language (KL-one) operates. Fahlman, on the other hand, returns to the original idea of tagging elements in the net, albeit with more complex structures and operations than those of Quillian (*see also* Charniak, 1981; Alshawi, 1982).

It follows that to describe a system as 'using a semantic network' says very little about it. Only if details are given of where the network fits into the overall system (e.g., set hierarchy to guide theorem prover) and what operations can be performed on the network (e.g., tagging) is any real information conveyed.

It might be claimed that these issues are only of minor, pedantic interest, since the main aim is to improve the adequacy of representations, rather than worrying about the underlying formalisms. We believe that if insufficient attention is paid to the foundations, then much time and effort will be wasted on ill-defined, uncommunicable notations, and in re-inventing trivial variants of existing techniques.

ACKNOWLEDGEMENT

This paper was written with the support of Science Research Council grant GR/A/74760.

APPENDIX—GENERAL DEFINITIONS

Defn. 1. A *net* is a quadruple (A, N, start, finish), where A and N are sets, and start and finish are functions from A to N. The set N is called the set of *nodes*, and A is the set of *arcs*. If x is in A, and n,m are in A, with $n = \text{start}(x)$ and $m = \text{finish}(x)$, then x is said to be *from n to m*.
Comment: This allows several arcs between two nodes, and allows arcs to form a loop from one node to itself. This generality is not included in conventional directed graphs.

Defn. 2. A *path* in a net (A, N, start, finish) is a sequence of nodes and arcs $\{n(1), a(1), n(2), a(2), \ldots\ldots, a(k), n(k+1)\}$ (with $n(i)$ in N, $a(i)$ in A, for each (i), such that, for each arc $a(j)$, $\text{start}(a(j)) = n(j)$ and $\text{finish}(a(j)) = n(j+1)$. The *length* of such a path is defined to be k (i.e., the number of arcs involved in it).

Defn. 3. A *directed graph* is a net (A, N, start, finish) such that, for each pair of nodes (n,m) in $N \times N$, there is at most one arc a in A with $\text{start}(a) = n$ and $\text{finish}(a) = m$, and such that there is no x in A with $\text{start}(x) = \text{finish}(x)$.
Comment: This is equivalent to the standard definition of a directed graph, in which the arcs are identified with pairs from $N \times N$. It will not be used in describing the graph structure of semantic nets, but it is introduced here for completeness.

Defn. 4. An *acyclic directed graph* is a directed graph (A, N, start, finish) in which there are no paths of length greater than zero $\{n(1), \ldots\ldots, n(k)\}$ with $n(1) = n(k)$.

Defn. 5. A *type-system* is a pair of sets ($P1$, $P2$) such that $P1$ is a subset of $P2$ (not necessarily proper). $P1$ is known as the set of *basic types* and ($P2 - P1$) is known as the set of *subtypes*.

Defn. 6. Given a net $X = (A, N, \text{start}, \text{finish})$, a *node-typing* of X is a pair $(f,(P1, P2))$ where $(P1, P2)$ is a type system, and f is a mapping from N to the set of subsets of $P2$ such that for each n in N, $f(n)$ contains exactly one element of $P1$.

Defn. 7. Given a net $X = (A, N, \text{start}, \text{finish})$, a *node-labelling of X* is a pair (NL, *nlab*), where NL is a set (the node-labels) and *nlab* is a function from N to NL.

Defn. 8. Given a net $X = (A, N, \text{start}, \text{finish})$, an *arc-labelling of X* is a pair (AL, *alab*) where AL is a set (the arc-labels) and *alab* is a function from A to AL.

Defn. 9.

1. A *fully-labelled net* is a triple (X, NODELAB, ARCLAB), where X is a net, NODELAB is a node-labelling of X, and ARCLAB is an arc-labelling of X.
2. A *fully-labelled typed net* is a pair (Y,T) where $Y = (X, \text{NODELAB}, \text{ARCLAB})$ is a fully-labelled net, and $T = (f,(P1, P2))$ is a node-typing of X.

Defn. 10. A *labelling system* is a pair $LS = (S, F)$ where F is a family of sets and S is the union of all the sets in F.
Comment: This rather trivial concept is introduced simply in order to make later definitions less verbose. It merely refers to a set of items and a classification imposed on them.

Defn. 11. An *arc-labelling system* is a pair of labelling systems $P = ((S1, F1), (S2, F2))$, where for every pair of sets x,y with x in $F1$, y in $F2$, x and y are disjoint. (Equivalently, $S1$ and $S2$ are disjoint). $S1$ is known as the 'unidirectional labels' of P, and $S2$ is known as the 'bidirectional labels' of P.
Comment: The sets are the various categories of labels (which may overlap), and the stipulation of non-intersection is so that no label can be classed as both one-directional and two-directional.

Defn. 12. A *semantic network formalism* is a tuple $S = (ONLF, OALF, T, OP, FR, SPL)$ where:
$ONLF = (ONL, ONLC)$ is a labelling system, with ONL known as the set of *organizational node labels in S.*
$OALF = ((OAL1, OAC1), (OAL2, OAC2))$ is an arc-labelling system; the union of $OAL1$ and $OAL2$ ($= OAL$) is known as the *organizational arc labels in S.*
T is a type system.
OP is a set of network operations.
FR is a set of network formation rules.
SPL is a semantic net programming language.
Comment: As discussed in the text (Section 3.8), we will not say anything further about the last three components.

Defn. 13. Let X be a semantic net formalism $F = (ONLF, OALF, T, OP, FR, SPL)$, let SNL be a set, (the 'substantive node labels'), and let SAL be another set (the set of 'substantive arc labels'), with SNL and SAL disjoint. Let ONL be the union of all the sets in $ONLF$, and let OAL be the union of all the sets in the components of $OALF$. Then:

1. A *one-level semantic net Y within X describing SNL U SAL* is a fully-labelled typed net $Y = (((N, A, \text{start}, \text{finish}), (NL, nlab), (AL, alab)), (f, T))$ such that:
   Y is well-formed with respect to FR.
   $AL \subseteq SAL \cup OAL$
   $NL \subseteq SNL \cup ONL$
2. An *entry-marked semantic net Y within X describing* $SNL \cup SAL$ *is a pair* (Y', n') where
   Y' is either a one-level or multilevel semantic net within X describing $SNL \cup SAL$.
   n' is a set of at most one node from Y'.
3. A *multi-level semantic net Y within X describing* $SNL \cup SAL$ is a fully-labelled typed net $Y = (((N, A, \text{start}, \text{finish}), (NL, nlab), (AL, alab)), (f, T))$ such that:
   Y is well-formed with respect to FR.
   $AL = SAL \cup OAL$
   $NL = SNL \cup ONL \cup$ the set of entry-marked semantic nets within X describing $SNL \cup SAL$).

Defn. 14. Let Y be a semantic net with nodes N, arcs A, and arc-labelling function *alab*. For any x,y in N, an *access path* from x to y is a sequence of nodes and arcs: $\{n(1), a(1), n(2), \ldots\ldots, a(k), n(k+1)\}$ (where $n(i)$ in N, $a(i)$ in A) such that

1. $n(1) = x$
2. $n(k+1) = y$
3. For each i, $1 <= i <= k$,

either $a(i)$ goes from $n(i)$ to $n(i+1)$,
or $a(i)$ goes from $n(i+1)$ to $n(i)$ and $alab(a(i))$ is a bidirectional label.

Defn. 15. In a multilevel semantic network, using the notation of the Defn. 13 above, a node x such that $nlab(x)$ is a semantic net (i.e., $nlab(x)$ is not in $SNL \cup ONL$) is termed a *supernode.*
Comment: This is used for 'partitioned semantic networks'.

REFERENCES

Abrial, J. R. (1974) Data semantics. *Database management* (J. W. Klimbie and K. L. Koffeman, eds.) pp. 1–59. Amsterdam: North-Holland.

Alshawi, H. (1982) *A Clustering Technique for Semantic Network Processing.* Cambridge: Computer Laboratory, University of Cambridge (Technical Report 25).

Bobrow, D. G. and Winograd, T. (1977) An overview of KRL: a knowledge representation language. *Cognitive Science 1*, 3–46.

Borkin, S. A. (1979) *Equivalence Properties of Semantic Data Models for Database Systems.* Cambridge, Mass: MIT (Report MIT/LCS/TR-206).

Brachman, R. J. (1977) What's in a concept: structural foundations for semantic networks. *International Journal of Man-Machine Studies 9*, 127–152.

Brachman, R. J. (1978) A structural paradigm for representing knowledge. Cambridge, Mass: Bolt Beranek and Newman (BBN Report 3605).

Brachman, R. J. (1979) On the epistemological status of semantic networks. *Associative Networks* (N. V. Findler, ed.) pp. 3–50. New York: Academic Press.

Bundy, A. (1979) What's the difference? predicate calculus and semantic nets (again). *AISB Quarterly October*, 8–9.

Charniak, E. (1975) *A brief for case.* Castagnola: Institute for Semantic and Cognitive Studies (Working Paper 22).

Charniak, E. (1981) *Passing markers: A theory of contextual inference in language comprehension.* Providence: Department of Computer Science, Brown University (Technical Report CS-80).

Deliyanni, A. and Kowalski, R. A. (1979) Logic and semantic networks. *Communications of the ACM 22*, 184–192.

Enderton, H. B. (1972) *A Mathematical Introduction to Logic.* New York: Academic Press.

Fahlman, S. (1979) *NETL: A System for Representing and Using Real World Knowledge.* Cambridge, Mass: MIT Press.

Fikes, R. E. and Hendrix, G. G. (1977) A network-based knowledge representation and its natural deduction system. *Proc. Fifth International Conference on Artificial Intelligence.* Cambridge, Mass: MIT.

Fillmore, C. J. (1968) The case for case. *Universals in Linguistic Theory* (E. Bach and B. Harms, eds). New York: Holt Rinehart and Winston.

Findler, N. V. (1979) *Associative Networks.* New York: Academic Press.

Geach, P. and Black, M. (1960) *Translations from the Writings of Gottlob Frege.* Oxford: Blackwell.

Guttag, J. V. and Horning, J. J. (1978) The algebraic specification of abstract data types. *Acta Informatica 10*, 27–52.

Harary, F., Norman, R. Z. and Cartwright, D. (1965) *Structural Models: An Introduction to the Theory of Directed Graphs.* New York: Wiley.

Hayes, P. J. (1977) In defence of logic. *Proc. Fifth International Conference on Artificial Intelligence.* Cambridge, Mass: MIT.

Hendrix, G. G. (1979) Encoding knowledge in partitioned networks. *Associative Networks* (N. V. Findler, ed.) pp. 51–92. New York: Academic Press.

Hudson, R. A. (1971) *English Complex Sentences.* Amsterdam: North-Holland.

Janas, J. M. and Schwind, C. B. (1979) Extensional semantic networks: their representation, application and generation. *Associative Networks* (N. V. Findler, ed.) pp. 267–302. New York: Academic Press.

Kay, M. (1973) The MIND system. *Natural Language Processing* (R. Rustin, ed.). New York: Algorithmics Press.

McCawley, J. D. (1968) Concerning the base component of a transformational grammar. *Foundations of Language 4*, 243–269.

McDermott, D. V. (1973) Assimilation of new information by a natural language-understanding system. Cambridge, Mass: MIT (M.S. Thesis).

McDermott, D. V. (1975) Very large planner-type data bases. Cambridge, Mass: MIT (AI Memo 339).

McDermott, D. V. (1978) Tarskian semantics, or no notation without denotation! *Cognitive Science 2*, 277–282.

McSkimin, J. R. and Minker, J. (1979) A predicate calculus based semantic network for deductive searching. *Associative Networks* (N. V. Findler, ed.) pp. 205–238. New York: Academic Press.

Minsky, M. (1975) A framework for representing knowledge. *The Pyschology of Computer Vision* (P. H. Winston, ed.). New York: McGraw-Hill.

Norman, D. E. and Rumelhart, D. A. and the LNR group (1975) *Explorations in Cognition.* San Francisco: Freeman.

Quillian, M. R. (1968) Semantic memory. *Semantic information processing* (M. Minsky, ed.). Cambridge, Mass: MIT Press.

Rieger, C. J. (1975) Conceptual memory and inference. *Conceptual Information Processing* (R. C. Schank, N. M. Goldman, C. J. Rieger and C. K. Riesbeck, eds). Amsterdam: North-Holland.

Ritchie, G. D. (1980) *Computational Grammar.* Brighton, Sussex: Harvester Press.

Roberts, R. B. and Goldstein, I. P. (1977) *The FRL manual.* Cambridge, Mass: MIT (AI Memo 409).

Schank, R. C., Goldman, N. M., Rieger, C. J. and Riesbeck, C. K. (1975a) Inference and paraphrase by computer. *Journal of the ACM 22*, 309–328.

Schank, R. C., Goldman, N. M., Rieger, C. J. and Riesbeck, C. K. (1975b) *Conceptual Information Processing.* Amsterdam: North-Holland.

Schubert, L. K. (1976) Extending the expressive power of semantic networks. *Artificial Intelligence 7*, 163–198.

Schubert, L. K., Goebel, R. G. and Cercone, N. J. (1979) The structure and organisation of a semantic net for comprehension and inference. *Associative Networks* (N. V. Findler, ed.). pp. 121–175. New York: Academic Press.

Scragg, G. (1975) Semantic nets as memory models. *Computational Semantics* (E. Charniak and Y. Wilks, eds). pp. 101–127. Amsterdam: North-Holland.

Shapiro, S. (1979) The SNePS semantic network processing system. *Associative Networks* (N. V. Findler, ed.) pp. 179–203. New York: Academic Press.

Shaw, A. C. (1970) Parsing of graph-representable pictures. *Journal of the ACM 17*, 453–481.

Simmons, R. F. (1973) Semantic networks: their computation and use for understanding English sentences. *Computer Models of Thought and Language* (R. C. Schank and K. M. Colby, eds.). San Francisco: Freeman.

Wegner, P. (1980) *Research Directions in Software Technology.* Cambridge, Mass: MIT Press.

Wilks, Y. (1976) Processing case. *American Journal of Computational Linguistics 56.*

Winograd, T. (1972) *Understanding Natural Language.* Edinburgh: Edinburgh University Press.

Wong, H. K. T. and Mylopoulos, J. (1977) Two views of data semantics: A survey of data models in artificial intelligence and database management. *Infor 15*, 344–378.

Woods, W. A. (1975) What's in a link? Foundations for semantic networks. *Representation and Understanding* (D. G. Bobrow and A. Collins, eds.) pp. 35–82. New York: Academic Press.