Semantic Networks: Visualizations of Knowledge

Roger Hartley and John Barnden

The history of semantic networks is almost as long as that of their parent discipline, artificial intelligence. They have formed the basis of many fascinating, yet controversial discussions in conferences and in the literature, ranging from metaphysics through to complexity theory in computer science. Many excellent surveys of the field have been written, and yet it is our belief that none of them has examined the important link between their use as a formal scheme for knowledge representation and their more heuristic use as an informal tool for thinking. Considering semantic networks as computerized tools, we will discuss what we believe to be the appropriate three levels of abstraction that can help understand how semantics networks are used.

Introduction

The history of the development of semantic networks is well known. Both Sowa [1] and Lehmann [2] have expounded in excellent scholarly fashion as to their origins in the study of language. Their later development as a tool for representing knowledge is also well known [e.g. 3, 4, 5, 6], as is their role in building computerized inference systems [e.g. 7, 8, 9, 10]. Indeed, the three aspects of intelligent thought, logic and language, will never be far from our discussion. From all these sources we learn that semantic networks have three main attributes:

1. they originate in the conceptual analysis of language
2. they have equivalent expressiveness to first-order logic
3. they can support inference through an interpreter that manipulates internal representations

However, there is something missing here. The visual aspect of the semantic network idea is clearly important. As Sowa says "network notations are easy for people to read" [1] and this pragmatic aspect of the formalism cannot be ignored. According to Sowa, "graphs ... can keep all the information about an entity at a single node and show related information by arcs connected directly to that node" [1]. In contrast, in symbolic logic notations, "the scattering of information not only destroys the readability of the formula, but also obscures the semantic structure of the sentence from which the formula was derived." So the battle is joined! The visual aspects of the semantic network notation is preferred (at least by Sowa) over the arcane, but more traditional notation of symbolic logic.

We hope to show in this paper that this argument is only one component of a larger, more complex one, involving the nature of semantics, and how different notations can lead to different systems with different pragmatic uses. In doing this I will touch on many aspects of formal and informal systems: meaning, formal semantics, visualization, the limitations of the human mind, and machine representations being among them. For those readers who like a preview of the denouement before reading the exposition, here it is without reading the last page. The formal aspects of semantic networks are barely distinguishable from those of more traditional systems of logic, but their pragmatic features are (a) vitally important, (b) poorly understood, and (c) ripe for good interdisciplinary research. The executives among you can stop here, since that is your summary. Everybody else has to read on!

What is a semantic network?

Although this question ought to be easy, in fact there is much disagreement in the literature. The common elements are clear, however. A semantic network involves three aspects:

1. a way of thinking about knowledge in which there are *concepts* and *relationships* among them.
2. a diagrammatic representation comprising some combination of *boxes*, *arrows* and *labels*.
3. a computer representation that allows database-like activity and sound inferencing using algorithms that operate on these representations

Some people say that the diagrams are the semantic networks, but we think it is more accurate to say that the digrams represent the semantic network which is really a network of concepts as held by a cognitive agent. The fact that we can draw them as diagrams, and represent them in computers makes them extremely useful for supporting work in cognitive psychology, and much of artificial intelligence. As we shall see, they also have uses in more informal settings, when they are often called semantic maps.

Meaning

Rather than start with natural language, we may as well jump right into the crux of the matter. The design and use of a knowledge representation revolves around the business of meaning. Actually, one spin-off from studies in natural language provides a good start, namely, the meaning triangle of Ogden and Richards [11]. The triangle relates objects in the real world, concepts that correspond to these objects, and the symbols that languages use to refer to them. Ogden and Richards' point, echoed by many followers, was that language only acquires meaning through some cognitive agent. In the triangle, there is no strong link between symbol and object. In mathematical terms, this mapping only exists as a composition of the mappings symbol → concept and concept → object (Fig. 1a)

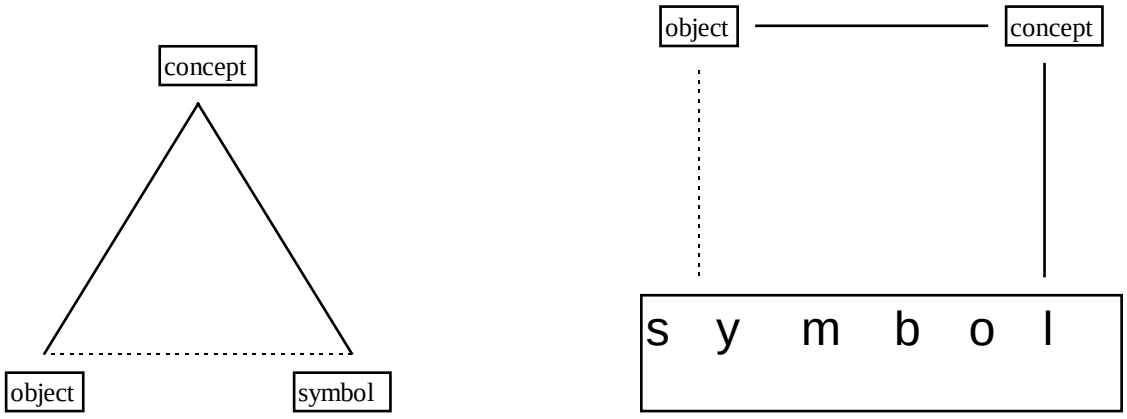

Figure 1a and 1b The meaning triangle, and the trangle distorted

It is no coincidence that the meaning triangle follows a visual metaphor. In fact, it is itself a semantic network that utilizes a number of features displayed by more comprehensive systems. It's worthwhile spending a few moments looking at the triangle, and seeing why it is a good way to convey the desired information. First, the triangle is always drawn as equilateral, thus giving equal status to the nodes. Second, it is symmetrical, giving a central role to the "concept" as intermediary between symbol and object. Third, the dashed line indicates a weaker link than the solid lines. It very neatly displays good use of diagrammatic features that are unavailable in classic symbolic forms. However, it is also clear that these features can be

abused, thus conveying the wrong information. Consider an alternative diagram that is topologically equivalent, but sends a distorted message (Fig 1b.) That diagrams can have a syntax is self-evident. That this syntax has a clear-cut semantic counterpart is less evident. This, I believe, hints at something that most accounts of semantic networks ignore. If, as most people believe, a semantic network is a better way to represent knowledge to a reader, whether human or machine, then where, in fact, does the improvement lie, and are there any limits to this improvement? Furthermore, are there clear-cut advantages when it comes to machine representations, or is it just "all the same" when knowledge gets digested by the machine? We hope to answer some of these questions in this paper, but, as always, there may be more questions raised during the attempt.

Levels of representation

The meaning triangle insists that meaning is a composition of two relations, from the symbols of language to the real world, through a concept-forming agent. However, this is not the only use of the term meaning. In formal semantics, symbolic structures are given meaning through interpretative mappings, or models, where the target of the mapping is some formal construct that uses mathematical, or at least, abstract ideas. The formal construct is often mant to capture the nature of the real world in some way, but it is not, in fact the real world.Typically this is done for symbolic systems such as formal logics [e.g. 12, 13, 14, 15], but also for programming languages [16, 17]. We will have more to say about these formal systems later, as we know already that semantic networks have formal logic equivalents. This idea is an important one when dealing with different, but seemingly equivalent representations. In [18], Brachman examined five levels of representation that semantic network designers commonly used, and assumed that they were all equivalent, in the sense of having the same ultimate meaning(Table 1)

Table 1 Brachman's levels of representation

LEVEL	COMPONENTS
Linguistic	Arbitrary concepts, words, expression
Conceptual	Semantics or conceptual relations (cases), primitive objects and actions
'Epistemological' (Structural)	Concept types, conceptual sub-pieces, inheritance and structuring relations
Logical	Propositions, predicates, logical operators
Implementational	Atoms, pointers

The intention was clearly to show that representations can be "high-level" or "low-level", just like computer languages. In this sense, then, logic is the assembler of representation systems, being only one step above the machine level, whereas a natural language is the furthest away from the machine. Thus, a semantic network represented at the conceptual level can be translated into the structural level, then to the logical level, and finally to machine data structures.

Whereas we could agree with the intent of the table, it is difficult to see how it fits with the reality of how knowledge representation is typically done, and, more importantly for us, with the meaning triangle. Let us discuss the first point quickly and move to the second. A knowledge representation system presents the user with two things: a basic ontology (which could be none at all) that provides semantic primitives with which to build representational structures, and mechanisms to relate these structures together. There is no careful

translation down the levels until the machine is reached; the knowledge is "compiled" into machine level structures in one go, just like a program in C or Pascal is translated into machine code in one go. Moreover, where do sorted logics fit into the table?--are they logical or epistemological? If a logic is constrained by ontological primitives, does it become a conceptual level system? The ideas in the levels are important, but the claim that there *are* clear-cut levels at all is probably bogus. More realistic, we think, is to make a simple split, as Sowa does, between external representations and internal representations, because there are important differences between human-readable forms and forms that can be stored and processed in a computer's memory.

The second point about Brachman's levels is that the meaning triangle does not distinguish among levels of symbol; in some sense, all symbolic representations, whether using linear text, such as logic, or two-dimensional forms such as semantics networks are equivalent. What is more, both logic and semantic networks (at least conceptual graphs) have equivalent forms in the other mode. Conceptual graphs have a linear, textual from, and logic has a diagrammatic form--Peirce's existential graphs. Here is Sowa's example from [1] in all four forms: (Figs. 2a-d) The sentence (another linear form) is "If a farmer owns a donkey, then he beats it". Peirce's diagrams (such as Fig. 2a) are perhaps the most opaque, but are, in fact, models of simplicity if one is willing to learn the ropes.

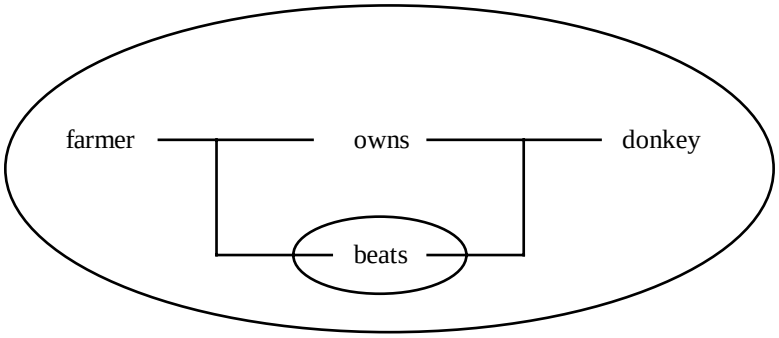

Figure 2a A diagrammatic, logical form

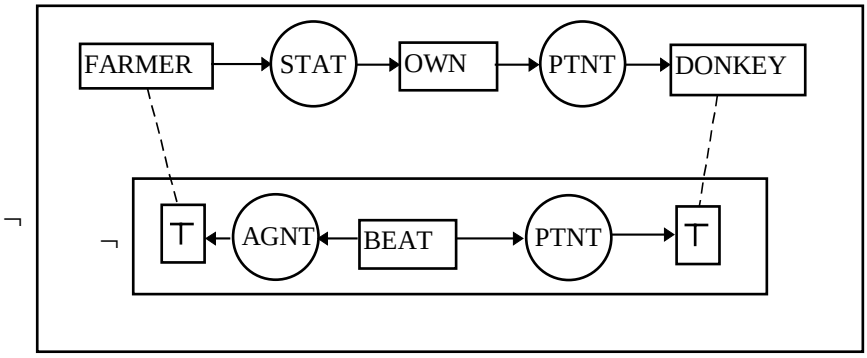

Figure 2b A conceptual graph form

$$(\forall x)(\forall y)\Big(\big(\, farmer(x) \wedge donkey(y) \wedge owns(x,y)\big) \supset beats(x,y)\Big)$$

Figure 2c A linear, logical form

```
~[[[OWN -
     (PTNT) -> [DONKEY:*D],
     (STAT) <- [FARMER:*F]]
   ~[[BEAT] -
      (AGNT) -> [T:*F],
      (PTNT) -> [T:*D]]]
```

Figure 2d A linear conceptual graph form

If we attempt to reconcile these forms in the context of the meaning triangle, we get the following diagram for the two systems of CGs (conceptual graphs) and FOL (first order logic), each with their two symbolic forms (Fig 3a):

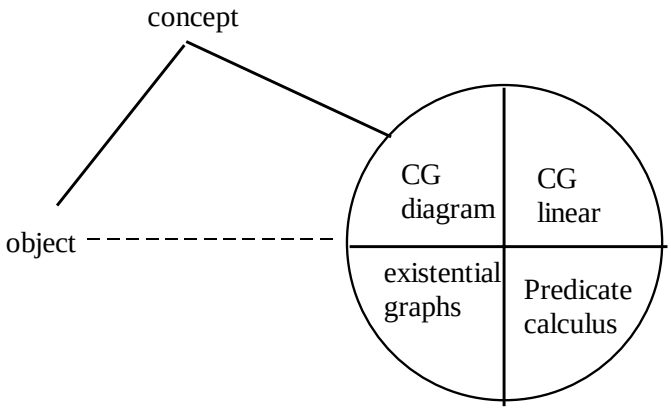

Figure 3a The meaning triangle with four alternative symbolic systems

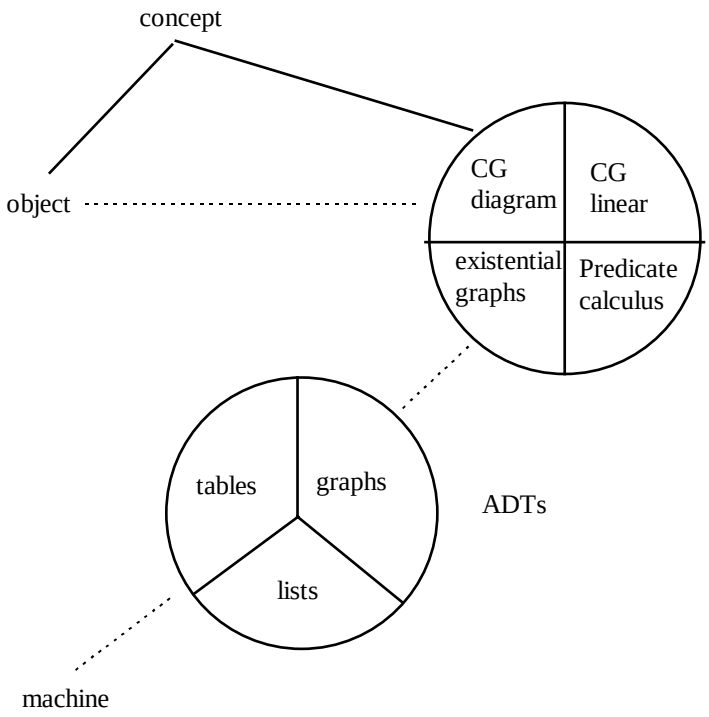

Figure 3b The ADT level added to the triangle

Thus the symbol systems all have the same meaning (although the method of mapping through the concept node is different in each case, hinting at a similar split in the concept node).

The representations in the circle are alternate external forms. We can add the distinction between external and internal representations adds to this picture. Brachman's lowest level has the machine concepts of "atoms and pointers", betraying his preference for implementations built using Lisp. Of course, any symbol system can be implemented using any data structure with adequate features. Traditional implementation of systems based on logic have been in Lisp because Lisp's list structures are perfect for representing linear expressions. What is more, pointers (hidden in the source code, but present in the programming model) can provide co-references, just like the arrows in a diagrammatic form. Prolog's internals contain similar hidden pointers. However, other structures can be used. Lendaris uses a hash-table form to represent graphs [19], and object-based implementations are also available. There is nothing special about atoms and pointers, other than ease of implementation.

A computer program is a symbol system, just like FOL or CG notation, but before adding these internal representations to the meaning triangle, I believe it is instructive to add a level between Brachman's lowest (machine) level and all the external symbolic forms. A programming language is used as an intermediary between concept and machine. The object-oriented languages support abstractions corresponding to Brachman's epistemological level, i.e. types, inheritance, containment, etc. (see [20] for an excellent exposition of these ideas). What one does in designing a program is to use these abstractions to support perceived relations and structure in the real world. However, programming these relations and structures directly is too hard (we did machine code programming in the 50s and we don't want to do it again!) so we use a programming language. The better ones (i.e. object-oriented) support our abstractions directly and the compiler can then translate these into machine form. It is natural to use a pointer/node abstraction to support a diagrammatic semantic network , but is this the most appropriate form for linear expressions, or is there even a better way to form these abstract ions, as in Lendaris's hash-tables? We clearly have choices,

governed by considerations of efficiency, size, speed, etc.--the usual computer science concerns. Thus, the internal representations must be mediated, though necessity, by an abstract data type; one which is supported by our chosen language. This diagram shows these situations: (Fig. 3b)

One way to interpret these additional levels is that we are trying to forge a link between symbol and object by using the machine representation as a substitute for the real world. In other words, the relation symbol → object, which previously could only be formed through a cognitive agent, can now be made explicit, even formalized, through the ADT level. The ADT then mediates the relation symbol → machine. The ADT level can thus be seen as a formal abstraction of the conceptual agent. Of course we end up with a different kind of 'object', but at least it is not a complete abstraction; the machine is a real object with real behavior. It is just that we have to interpret its behavior as if it were real-world behavior, and we may of course be wrong in doing so since our programming may not be adequate, or correct.

Conceptual Schemata and Formal Worlds

Although we have argued that every symbol system is, in some sense, equivalent, it still remains that alternative formulations have different starting points. To illustrate this, we will take logic as our starting point, although it is really semantic networks that we need to discuss. It is undeniable that different cognitive agents have different ways of looking at the world. This may or may not be due to the Whorfian hypothesis, that our language system colors our conceptual apparatus. Let us simply take this as a phenomenological fact, and call the way of looking at the world a conceptual schema. If the schema might be described as "logical" (cf. Mr. Spock on Star Trek) then any instantiation of this schema would be also logical in nature. This is what we call a theory in logic--a particular collection of propositions, all assumed to be true, and consistent one with another. The external manifestation of the theory might be in predicate calculus (PC), a symbolic language. . Fig 4a shows the meaning triangle in two levels. The lower level is an instantiation of the upper, schematic, level. The question mark is the abstract counterpart to the real world; the schema of which our real world is an instance.

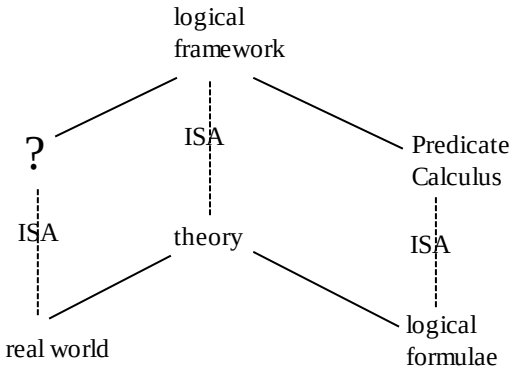

Figure 4a The schema level in the triangle

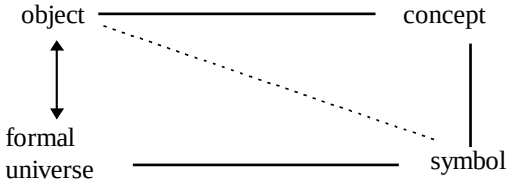

Figure 4b Formal semantics and the meaning triangle

Of course, this is the stuff of ontology, that branch of philosophy that asks the question "of what does the world consist?". For logic, the ontology is fairly simple. There are objects, only differentiated by name, and there are various kinds of relations that can relate the objects together. The world is thus seen as a static arrangement of objects. The semantic network alternative comes from substantially the same view. It is no wonder that the visualization of the network as nodes and arrows fits this picture exactly. The presence of a cognitive agent in the picture is not allowed to change the picture, since the ontology is simple and fixed.

This is not the only ontology that is possible. In fact, 'ontological engineering' is becoming a big part of knowledge representation systems as we try to pin down domains as to what they contain, and how to handle knowledge about these domains in useful ways [21].

Semantic networks systems have usually been built with either informal semantics, represented by program behavior (what the program does defines the semantics) or with incomplete semantics. Those that rely on the equivalence to logic through external representations usually draw also on the well-defined formal semantics of logic. However, many of the visual aspects of semantic networks we have discussed do not play a part in this since they have no formal counterpart in logic. Thus, proximity of nodes, placement of nodes and link weights have no place in logic.

The usual way of supplying a formal semantics is by defining an abstract *stand-in* for the real world in the meaning triangle. Fig. 4b shows how, for instance, a formal model can be a real-world substitute to bypass the usual attachment of meaning to a symbol though a cognitive agent. The fourth node is assumed to be an formal abstraction from the real world, suitable for the techniques of denotational semantics and model theory. The double-headed arrow in the diagram shows an assumed relationship between the elements of the formal model and the real world, although sometimes there are abstract things in the formal universe that have no obvious real-world counterpart. One example is the functional terms that form part of the Herbrand model[1] that only loosely relate to operations on real objects.

How do computers change things?

We have discussed the meaning triangle, which really relates, as we have seen, an ontology with a conceptual schema and an (external) symbol system. However, the computer scientists, through the field of artificial intelligence have added a fourth leg--the internal representation. The advantage of using these internal representations are:

1. They can make the linguistic representations *computable*. What this boils down to is that the rules of a formal system can be applied to the representations to explore complex possibilities of inference, and even proof. Without the computer, logic would be too hard for describing complex systems. In fact, Prolog, which uses a restricted, but useful, form of first-order logic, can make logic accessible even to people unversed in the arcana of the formal calculus.

2. They make persistent databases of knowledge possible. We can store, in a permanent fashion, efficient external representations of human memory that permit complex queries and infallible processing. Again, without them, these applications would not be possible.

3. They can make the diagrammatic representations useful over and above their use in small motivational examples. Human-computer interface technology helps here. This particular aspect of visualization has been, we believe, neglected when it comes to discussing semantic networks. *How* useful diagrams can be and what their limitations are needs thorough investigation.

[1] The Herbrand model is a simple formal model that is often used to give meaning to logic programs.

All of these uses rely on efficient internal representations fit only for machine consumption. The computer must be programmed, however, to cater for the limitations of the human mind. Two areas of concern are information retrieval and presentation. If too much information is thrown at the user too fast, the mind can easily be overloaded. A database can be programmed to dump itself on the screen, but there is little point, since no-one can read it. The manner in which information is presented is also crucial. Only certain modes yield good results. A visual display that is too complex also produces cognitive fatigue. These issues are acknowledged, but there are no easy solutions, and old-wives tales are common. For instance, "a picture is worth a thousand words" is true but it does depend on what kind of picture, and how it is presented. Fig 5 shows a complex semantic network in a typical problem-solving system. I defy anyone to "get the picture" from this graph without a lot of help in terms of descriptive support.

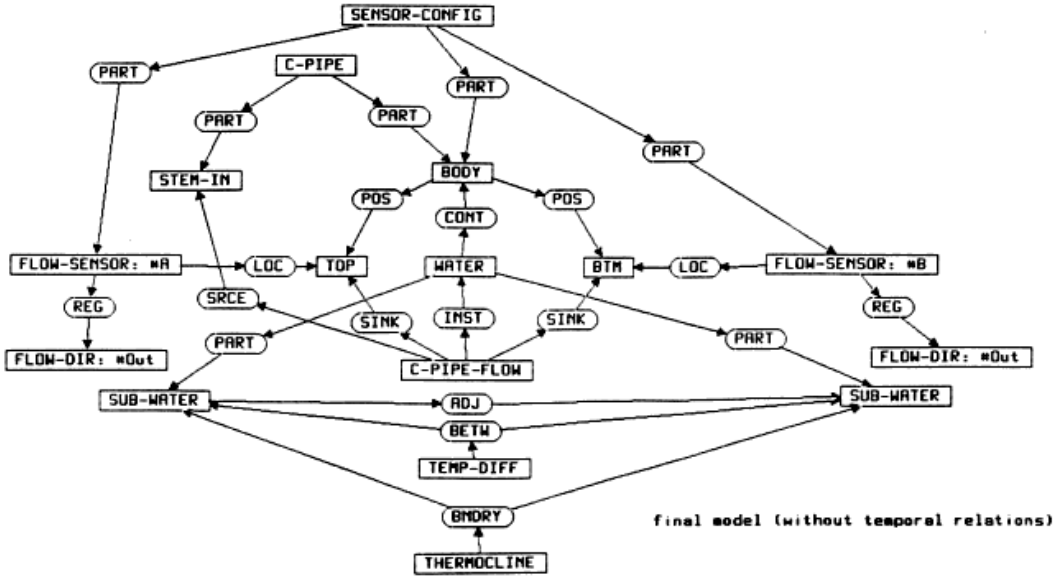

Figure 5 A complex semantic network

The question of whether semantic networks convey *computational* advantages over other representational approaches such as logic is somewhat vexed. In large measure, the *visual* advantage (if any) of semantic networks for human beings rests on the property of locality, as Sowa points out in [1]: for a given real-world item, such as the person John, that is represented in the network by a node, all information that is specifically about John is attached to the node. For example, if John is represented as loving Mary, then the network substructure that represents that state of affairs includes the John node. And why should the attachment of the John node be important? It is because the human eye can follow the links easily and efficiently.

Now, the typical computer implementation methods for semantic networks preserve this conception of locality to some important degree, converting it into a type of computational locality. In fact, there is a virtually universal *tacit assumption*, on the part of semantic networks community, that an implementation does preserve the locality. A neutral and general way to put the assumption is as follows:

> *when the system (e.g., person or AI program) is attending to a given node N, it is quick for the system to transfer attention to the node or nodes that are connected to N by a single link of a given type.*

For instance, if the system is attending to WATER node in Fig. 5 then it is quick for it to transfer attention to the BODY node, through the CONT(ains) link. (We do not mean that every type of link has to be

susceptible to rapid traversal. Most importantly, a link might be rapidly traversable in one direction but not the other.)

In sum, much as the human eye can follow links efficiently in a diagram of the net, the computer is assumed to be able to do so in its internal implementation of the net. Thus the visual benefit is (partially) transformed into a computational benefit.

All this provides a contrast with formal logic, which has no comparable implementational assumptions, tacit or otherwise. But we must resist the temptation to say that semantic networks therefore have a corresponding computational advantage. In practice, when we implement a logical representation scheme in a computer, for reasons of efficiency we need to include indexing schemes in the implementation. This is to allow related pieces of information to be brought together efficiently. For example, one indexing scheme is for each individual constant symbol in the logic, such as the constant JOHN, to be an efficient indexing key to all formulae that involve that symbol. Thus, information that is specifically about the person John should be quickly accessible from the JOHN constant symbol. (See [27] for a useful textbook introduction to indexing in logic implementations.) We therefore do consciously impose a parallel to the computational locality that is assumed to hold for semantic networks.

This is not to say that if one were to take a semantic network, convert what it says somehow into a logical formalism, and then implement that framework one would necessarily end up with the same types and degrees of efficiency. One would have to specially engineer the logic implementation in order for that equivalence to hold. Moreover, as we have already pointed out, the choice of which ADT to use will govern the end result.

A full comparison of logic and semantic networks would need to address the way in which semantic networks facilitate inheritance-based reasoning and allow for exceptions to inherited information. Again, there is no real advantage over logic, because logics can be implemented or designed in such a way to have those advantages. One specific recent movement is to use description logics (again, [27]), which are specifically designed to capture some advantages of semantic networks.

Pictorial forms

Most semantic networks are drawn first, and given a linear, textual form later. This is particularly true of the major systems, KL-ONE, conceptual graphs and SNePS. Actually, however, they are not pure pictorial forms. They are text/picture hybrids (diagrams), where words or abbreviations label the nodes, whether drawn in boxes or not, and arrows, curved or not, connect them. Sometimes the arrows are labeled, sometimes not. Schank realized this when he used the phrase "picture producer" in his particular brand of semantic network, the conceptual dependency graph [22]. The labels in his graphs were external representations of concepts held by a cognitive agent, where each concept was assumed to produce a picture of the corresponding object, or at least of an archetypal object. Thus the word "CAT" was assumed to represent the concept of a cat, a concept being something that the mind could envision. A purely pictorial semantic network representation might use pictures to do the same job. Fig. 6a shows a conceptual graph of a cat sitting on a car, and Fig. 6b a pure picture form. Of course, we could disagree about whether my car looks like your idea of a car, but this happened with the word 'CAR' anyway. Notice the use of a solid line for the 'agent' relation, whereas there is a picture for the 'supports' relation. Perhaps better would be to use the characters of the Asian languages, which are both pictures *and* words. Figure 6c shows a Japanese version of the same thing.

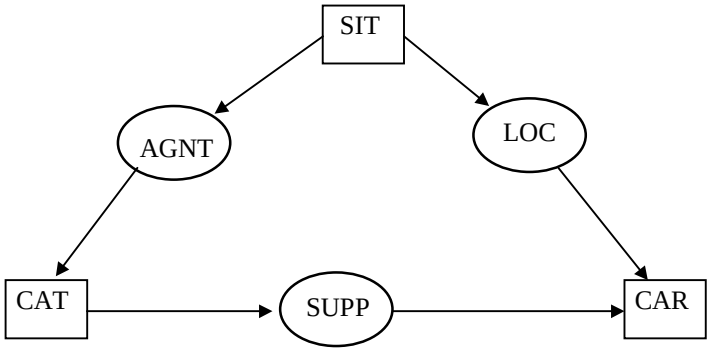

Figure 6a A conceptual graph for "A cat sitting on a car"

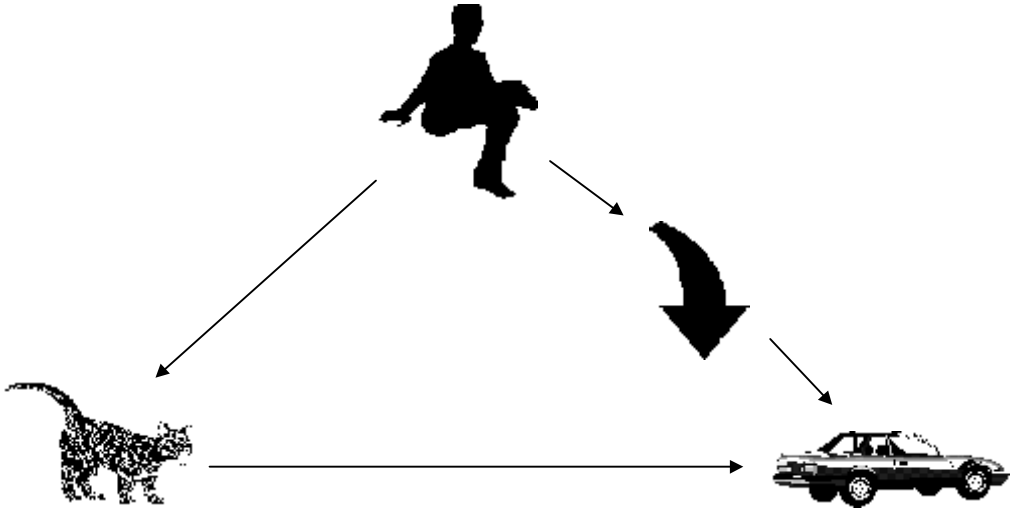

Figure 6b A pure picture form of a conceptual graph

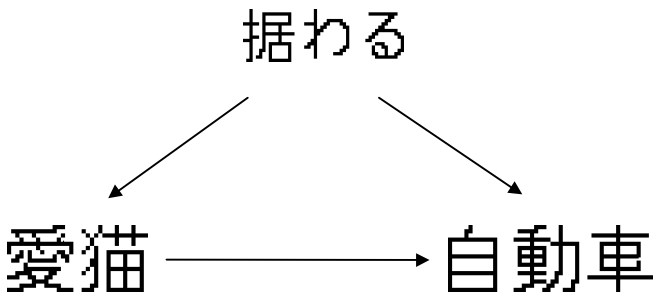

Figure 6c A Japanese language version[1]

Diagrammatic forms like these have been advocated often as a useful brain-storming tool (e.g. [23]). Our children are sometimes taught these techniques using paper and pencil. Conceptual mapping is a recognized way of getting ideas down on paper so that their relationships can be seen rather than contemplated abstractly. In this sense, the properties of a graphical network become important. Placement of the nodes on

[1] We claim no real knowledge of Japanese, but the principle of using pictograms should be clear.

the page is important; the more centrally placed a node is, the more 'central' the associated concept is in the cognitive structure. More importantly, the length of the links signifies similarity in concept. This property has been exploited in Pathfinder networks [24] where a clever algorithm determined the best placement of nodes determined by their cognitive 'distance' as reported by a human agent. As mentioned above, this aspect of the diagrammatic form is absent from many descriptions of semantic networks, as if it was somehow irrelevant. At least Sowa attempts in [1] to say why the diagram is better than the corresponding FOL form, but then proceeds to give a linear form in his conceptual graph notation that is clearly as good as the diagram![2] The two are virtually interchangeable (Figs. 7a, b). Perhaps the example is a poor one., since there *must* be advantages to the diagrammatic form, it's just that no-one can quite put their finger on it. How nearness and connectedness figure into the equation is never spelt out.

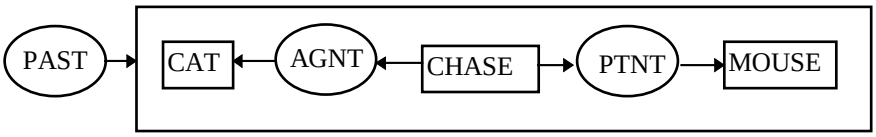

Figure 7a Sowa's 'good' conceptual graph

```
(PAST) -> [[CAT] <- (AGNT) <- [CHASE] -> (PTNT) -> [MOUSE]]
```

Figure 7b The linear form of Fig 7a.

Visualization in concept maps

Perhaps if we turn to the ideas in concept mapping, we might find an answer. A concept map is a visualization technique used for brainstorming, for exploration of ideas, and discussing complex systems among the members of a group. [25] shows excellent examples of kinds of concept maps, and their use. For instance, Fig. 8a shows their 'Systems' concept map, which they say is 'similar to a flowchart'. However, there is no discussion of why the diagrams look like they do., or why they do indeed convey the required information. [26] shows a concept map for water that looks suspiciously like a semantic network. Indeed, the only real difference between the two as presented there seems to be a pragmatic one, and perhaps more formality in the semantic network's choice of relations. Concept maps are used for knowledge acquisition and analysis, and have a useful heuristic nature, whereas semantic networks are more used for representation in systems, especially computer systems. It is clear from all the examples of concept maps, that their usefulness degrades as the size increases. As Fig. 5 shows, these diagrammatic forms become useless as complexity gets too high, and the linear form can serve just as well (or as badly, since the linear form is *always* difficult to read). Perhaps a graph such as Fig. 8b might show the relative usefulness of the alternate external representations as complexity increases. Diagrams are better than the linear form for reasonable size chunks of knowledge, by ultimately they both fail because of the limitations of human cognition.

[2] Actually even the linear form in the article is 'diagrammatic' since it contains non-ASCII characters--the arrows.

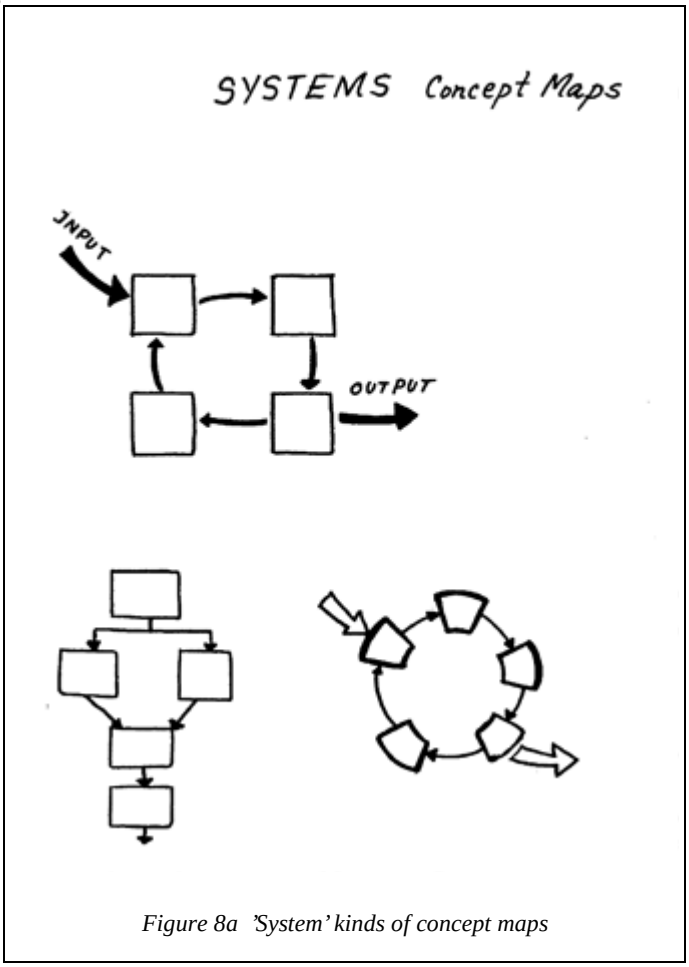

Figure 8a 'System' kinds of concept maps

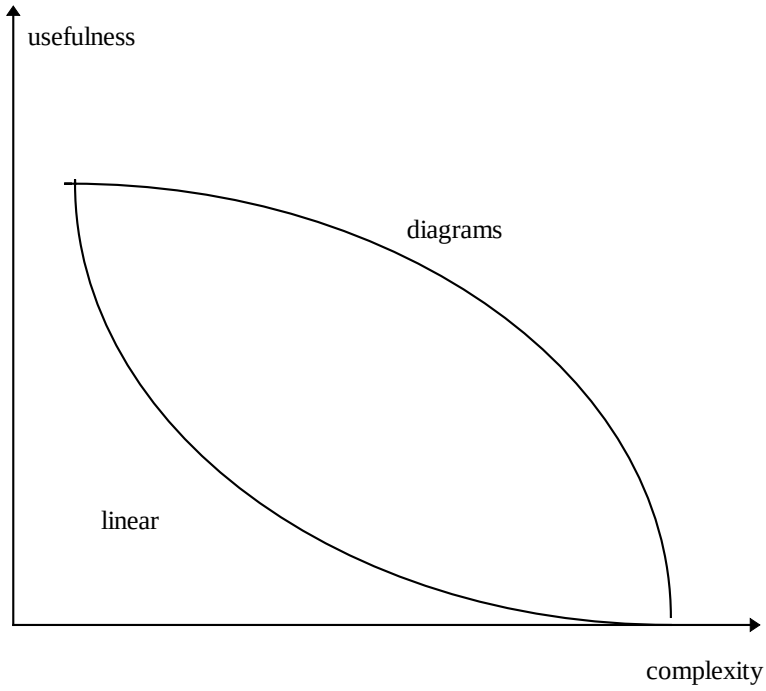

Figure 8b Showing complexity vs. usefulness for alternative representations

Conclusion

We have not discussed an important aspect of semantic networks that deserves a longer discussion than is possible here. The representation of procedural knowledge (knowing 'how', instead of knowing 'that') has been approached from several different angles, but has not yet yielded, we believe, to a definitive analysis. Bringing together the ideas of procedural attachment, the inference engine that interprets a network, and formal semantics of programming languages might be a starting point. This is only one of a number of outstanding questions concerning semantic networks that we do not have space or time to discuss:

- how can semantic networks best represent procedural knowledge?
- do semantic networks have any benefits for knowledge acquisition over text-based methods?
- does the visual aspect of semantic networks help or hinder knowledge representation?
- what is the best way to implement a knowledge representation system based on semantic networks?
- can (and should) the ties to formal logic and logical inferencing be broken?
- at what level of complexity do the visual aspects of semantic networks break down?

Semantic networks have led a dual existence, as a motivational diagrammatic form of knowledge representation, and as an internal, computerized form, suitable for a variety of computational methods. Their visual aspect, however, while acknowledged, has not been studied extensively enough. Their resurrection as tools for visualization of the structure of knowledge should be explored, and they have much to offer the field of concept mapping in terms of their formal underpinning and the experience of the artificial intelligence community that uses them

References

[1] Sowa, J.F. *Semantic Networks*, in Encyclopedia of Artificial Intelligence, S.C. Shapiro (Ed.) NewYork: Wiley and Sons (1987).

[2] Lehmann, F. Semantic Networks. Computers Math. Applic. 23(2-5), 1-50, (1992).

[3] Brachman, R.j. and Schmolze, , J.G. *An Overview of the KL-ONE Knowledge Representation System*, Cogn. Sci. 9(2), 171-216 (1985).

[4] Schank, R.C. *Conceptual Information Processing* (Ed.) Amsterdam: North Holland (1975).

[5] Shapiro, S.C. and Rapoport, W.J. *SNePS Considered as a Fully Intensional Propositional Semantic Network*, in Cercone, N. and McCalla, G. (Eds.) The Knowledge Frontier, New York: Springer Verlag, 263-315, (1987).

[6] Sowa, J.F. Conceptual Structures: Information Processing in Mind and Machine, Reading, Mass: Addison Wesley (1984).

[7] Winston, P.H., Learning structural descriptions from examples, in Winston, P.H. (Ed.) The Psychology of Computer Vision. New York: McGraw Hill (1975)

[8] Rieger, Chuck. An organization of knowledge for problem-solving and language comprehension. Artificial Intelligence, 7(2) 89-127. (1976)

[9] Brachman, R.J., Fikes, R.E. and Levesque, H.J., KRYPTON: A functional approach to knowledge representation. IEEE Computer 16(10), 67-73. (1983)

[10] Coombs, M.J and Hartley, R.T., The MGR algorithm and its application to the generation of explanations for novel events. Int. J. Man-Machine Studies 27, 679-708 (1987)

[11] Ogden, C.K. and Richards, I.A. The meaning of meaning. New York: Harcourt, Brace and World (1923).

[12] McDermott, D. and Doyle, J. Non-monotonic logic I. Artificial Intelligence 13, 41-72 (1980)

[13] Poole, D.L. A logical framework for default reasoning. Artifiicial Intelligence, 36 (1988)

[14] Przymusinski, T. Three-valued non-monotonic formalisms and semantics of logic programs. Artifiicial Intelligence, 49, 309-343 (1991)

[15] Epstein, R.L. The semantic foundations of logic: predicate logic. New York: Oxford University Press. (1994)

[16] Strachey, C., Towards a formal semantics, in Steele, T.B. (Ed.) Formal language description languages. Amsterdam: North Holland (1966)

[17] Scmidt, D.A., Denotational semantics: a methodology for language development. Dubuque, Iowa: Wm. C. Brown. (1986)

[18] Brachman, R.J. On the epistemological status of semantic networks, in Findler, N., (Ed.) Associative networks: representation and use of knowledge by computers. New York: Academic Press (1979)

[19] Lendaris, G.G., Conceptual graphs as a vehicle for improved generalization in a neural network pattern recognition task.. Proc. Fifth Annual Workshop on conceptual structures, (1990)

[20] Budd, T., An introduction to object-oriented programming. Reading, Mass.: Addison Wesley (1996)

[21] Various authors. Proc. Workshop on Ontological Engineering. AAAI Spring Symposium, Stanford U. (1997)

[22] Schank, R.C. and Rieger, C.J., Inference and computer understanding of natural language. Artificial Intelligence 5, 373-412 (1974)

[23] Gowin, B.D., and Novak, J.D. Learning How to Learn. New York: Cambridge University Press. (1984).

[24] Schvaneveldt, R.W., Durso, F.T., and Dearholt, D.W. Pathfinder: scaling with network structures. Tech. Rep. MCCS-85-9. Computing Research Lab. at New Mexico State University (1985)

[25] AIM Lab. http://classes.uiuc.edu/AIM/Discovery/Mind/c-m2.html

[26] Boyer, P. http://cs.uwp.edu/staff/boyer/100/concept.mapping.faq.html

[27] Russell, S. and Norvig, P. Artificial Intelligence: A Modern Approach. Englewood Cliffs, N.J.: Prentice-Hall. (1995)