From: Proc. 6th IJCAI, Tokyo, Aug. 20-23, 1979

PROBLEMS WITH PARTS

Lenhart K. Schubert
Department of Computing Science
University of Alberta
Edmonton, Alberta T6G 2H1*

Abstract. Two problems in representing and using relationships among parts of objects are analysed, and partial solutions are proposed. The first is the problem of extracting information from overlapping partitioning hierarchies. Contrary to a common assumption, "part -of" relationships cannot be extracted from arbitrary sets of partitioning assertions by simple label-propagation methods: the problem is in general $\overline{NP}$-complete. However, if the lowest-level parts of all entities under consideration are drawn from a common pool of pairwise disjoint "ultimate" parts, then relatively simple, complete inference methods for deriving "part -of" and other relationships can be supplied. The second problem is that of property inheritance, i.e., the transfer of relationships among parts of a generic object to corresponding parts of a successor of that object in the type hierarchy. Earlier solutions are criticized and a new solution based on function tables attached to concepts is proposed.

1. Introduction

Much of our knowledge of enduring relationships (as opposed to events) concerns relationships among parts of physical and abstract objects. Yet part-whole relationships have until recently received scant attention in the AI literature, compared for example to generalization (IS-A) relationships. Interest in parts structure has been confined almost exclusively to its role in relational models for computer vision (e.g., [1-5]). However, the ability to reason about parts was an important feature of Raphael's program SIR[6]. It was able to chain together both particular and generic subpart relationships, and to combine parts inference with subset inference to answer questions about (sets of) parts of (sets of) individuals. The "slots" of Minsky's frames [7] may facilitate the representation of part-whole relationships, among others, but do not presuppose any particular theory of such relationships. To date, the most elaborate proposals for representing and reasoning about parts structure appear to be those of Philip Hayes [8,9].

The present paper addresses two problems in the use of parts knowledge which people find trivially easy but which have not been adequately mechanized: the extraction of "part-of" relationships (and disjointness relationships, etc.) from overlapping parts hierarchies, and relationship inheritance for parts of objects in a type hierarchy.

Sec. 2 introduces a convenient representation for sets of partitioning assertions, called "parts graphs". Although based on partitionings, parts graphs can represent non-exhaustive and overlapping decompositions of parts structure.

Sec. 3 notes that label-propagation methods for extracting "part-of" (and other) relationships from simple partitioning hierarchies do not generalize to arbitrary parts graphs; the problem is in general $\overline{NP}$-complete. A restricted type of parts graph, called "closed", is defined, which reduces all parts of the top-level object to a common set of pairwise disjoint ultimate (lowest-level) parts. For closed parts graphs, efficient complete inference methods can be supplied.

In Sec. 4 a relationship inheritance scheme is proposed in which corresponding parts of concepts in a type hierarchy are identifiable by the names of their Skolem functions. This overcomes a difficulty with Hayes' [8,9] property inheritance scheme, without sacrificing any of its advantages.

---

* The work reported was done while the author was on leave at Universität Karlsruhe, Institut für Informatik I, W. Germany

2. Parts graphs

The following takes for granted a notion of "part-of" which is at least a partial ordering ~~with the extension property (if all parts of x are part of y, then x is part of y)~~, and for which there exists a unique empty part Θ, a dyadic "overlap" function $\sqcap$ (the largest entity which is part of both arguments), an n-adic "merge" function $\sqcup$, n=2,3,... (the smallest entity which has all of its arguments as parts), * and a dyadic "remainder" function $\setminus$ (the part of the first argument which is not part of the second). These properties are stated formally in [10]. Bunt's [11] axiomatization of "part-of" satisfies these constraints and leads to a logically consistent generalization of Zermelo-Fraenkel set theory. Thus the theories of "part-of" and "subset-of" are closely related, and much of what is said here about the representation and use of parts knowledge also applies to the representation and use of knowledge about set inclusion relationships.

People tend to conceptualize objects in terms of pairwise disjoint, jointly exhaustive parts, i.e., in terms of partitionings. For example, the top level of a human parts hierarchy might reasonably divide the human body x into a head h, neck n, torso t, arms a&a', and legs l&l'. Thus it is natural to base computer representations of parts structure on a partioning relation P,[1] where [x P $x_1 \ldots x_m$] iff

$$\bigwedge_{1 \le i < j \le m} [(\sqcap x_i\ x_j) = \Theta] \ \& \ [(\sqcup x_1 \ldots x_m) = x].$$ [2]

Thus $(\forall x)$ (∃hntaa'll')[[x human]→ [[x P h n...l']&[h head]&[n neck]&...&[l' leg].

It is important to notice that "part-of" is expressible in terms of P. For example, [b part-of a] can be rewritten as [a P b c], where c is a previously unused constant denoting $(\setminus a\ b)$, i.e., the (possibly empty) remainder of a with respect to b. In general, arbitrary sets of "part-of" assertions and disjointness assertions can be converted to sets of partitioning assertions. However, the conversion may introduce parts wich are possibly empty, even if all of the original parts are known to be nonempty.

---

[1] cf. the partitionings of Grossman [12] and "complete splits" of Fahlman [13] for concept taxonomies.

[2] In the form of predicate calculus used here sentential formulas are in infix form and delimited by square brackets (so that the second list element is the predicator or operator) and functional expressions are in prefix form and delimited by round brackets; e.g., (∀xy)[[x =(mother-of y)] → [x female]].

*where $\sqcup$ and $\sqcap$ are mutually distributive

Sets of partitioning assertions can be represented as "parts graphs" as illustrated in Fig.1.

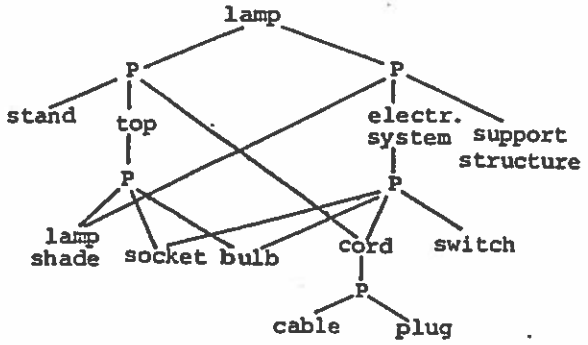

Fig.1. Parts graph for a desk lamp. The nodes marked "P" represent partitioning assertions while the named nodes represent parts.

This shows two overlapping parts hierarchies for a particular (as opposed to generic) lamp, with a "structural view" on the left and a "functional view" on the right.

In the graphical representation, partitioning assertions and particular parts are represented as assertion nodes and constant nodes respectively. Assertion nodes are marked "P" and have an incoming edge from the first argument of the assertion and outgoing edges to the remaining arguments. Such parts graphs are readily represented as semantic nets in the network formalism of [14].

Hierarchies are distinguished from unrestricted parts graphs by having exactly one downward edge from each nonleaf parts node and a unique path from the root to each parts node. As Fig.1 shows, the available knowledge about an object need not form a single hierarchy. "Tangled" hierarchies result when there are multiple views of the same object, known overlap regions such as shoulders, waist, or knees, or parts with unknown mutual overlap.

3. Extracting information from parts graphs

A partitioning hierarchy has the important advantage of allowing efficient inference of part-of and disjointness relationships. In fact, if a and b are any two nonempty parts appearing in a hierarchy, then b is part of a iff a is an ancestor of b, and a and b are disjoint iff neither is an ancestor of the other. [3]

For parts within a common subhierarchy of a

---

[3] Incidentally, these conditions can be checked in constant time using a method based on left-to-right numbering of the leaves, whereas the worst-case complexity of label-propagation methods is linear in the number of edges.

non-hierarchic parts graph the same methods can be used. For example, in Fig.1 the plug must be part of the electrical system since the latter is an ancestor of the former in the "functional view" of the lamp; the cord is disjoint from the bulb since neither is an ancestor of the other in the "structural view".

However, not all implicit part-of and disjointness relationships can be extracted in this way. In Fig.1, the switch must be part of the stand; in fact, the switch and support structure must partition the stand, because the stand on the one hand and the switch plus support structure on the other are the only nonoverlapping parts of the two lamp hierarchies. Another point worth noting is that even without the assertion [lamp P stand top cord], it follows from the graph that "top" is part of the lamp, because all of its parts are.

One can devise additional inference methods to handle these particular cases, but such stopgap measures are almost surely futile because of the following result.

Theorem 1. The problem of confirming [b part-of a], where a and b are nodes of a parts graph, is NP-complete.

This is proved in [10] by mapping the unsatisfiability problem of the propositional calculus into a "part-of" problem for a parts graph, and vice versa.

This suggests that additional constraints should be imposed on parts graphs so as to permit efficient information extraction, while still allowing for overlap parts and multiple views of the same part. Such constraints are obtained by requiring parts graphs to be closed: G is closed iff any two of its parts nodes are projectible into a common subhierarchy. A node a is projectible into a subhierarchy H if G contains a subhierarchy rooted at a whose leaves lie in H. (Since a single parts node is a subhierarchy, any parts node is trivially projectible into any subhierarchy to which it belongs. Hence any two nodes of a subhierarchy H are projectible into a common subhierarchy, viz.,H).

For example, if the left and right views of the lamp in Fig.1 are called H_L and H_R, then "top" and "bulb" are projectible into H_L (because they belong to H_L), and "top" and "electrical system" are projectible into H_R (because the leaves {lamp shade, socket, bulb} of a subhierarchy rooted at "top", as well as "electrical system", belong to H_R). However, neither "stand" and "switch", nor "stand" and "support structure" are projectible into any common subhierarchy. Thus the graph is open, but could easily be closed by addition of the assertion [stand P switch support-structure].

The notion of a closed graph becomes clearer if parts graphs are required to be "fully consistent" in the sense that none of the parts they represent are necessarily empty. This rules out graphs, for example, in which two disjoint parts (distinct leaves of a single subhierarchy) have a common descendant, or in which one "view" of a part terminates in a proper subset of the leaves of another "view" of that part.

It is proved in [10] that all leaf nodes of a fully consistent, closed graph belong to a single (not necessarily unique) main hierarchy whose root represents the whole entity. All nodes of the graph are projectible into this main hierarchy. In such a graph, therefore, all leaves are pairwise disjoint and each parts node corresponds to a subset of the leaves. The graph of Fig.1 is now easily seen to be open, since the leaf nodes "stand" and "switch", for example, do not belong to any common subhierarchy.

Fig.2 shows a closed graph with a main hierarchy rooted at a, an overlap part b made up of two parts occurring in the main hierarchy, and two "views" of a part c.

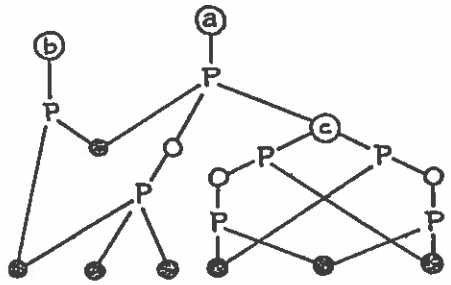

Fig. 2. A closed parts graph with an "overlap part" b and two "views" of a part c. The darkened nodes represent the ultimate parts of a.

Surprisingly, the syntactical restrictions of closed graphs do not lead to any logical restrictions:

Theorem 2. For every parts graph there is a logically equivalent closed graph.

The proof and a procedure for constructing a closed graph from a set of partitioning assertions are given in [10]. The sort of idea involved is illustrated in Fig.3. Admittedly, the size of the equivalent closed graph may be exponential in the size of a given open graph. However, this exponential growth is associated with unknown and unrestricted overlap between sets of parts, a situation which rarely, if

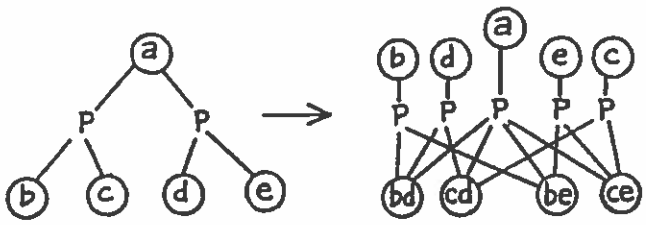

Fig. 3. An open graph and a logically equivalent closed graph.

ever, occurs in our conception of real objects.

The following methods are complete for answering questions of the form ?[b part-of a] and ?[a disjoint-from b] in fully consistent, closed graphs. Projecting a node a into a hierarchy H means determining a set of nodes $A = \{a_1,\ldots,a_m\}$ which lie in H and are the leaves of a subhierarchy with root a (if a already lies in H then $A = \{a\}$).

Preliminary step: Project a and b into a common subhierarchy H, obtaining respective projections $A = \{a_1,\ldots,a_m\}$ and $B = \{b_1,\ldots b_n\}$.

To answer ?[b part-of a]: If every node with an ancestor in B has an ancestor or descendant in A, return "yes"; if some nodes with ancestors in B have no ancestors and no descendants in A, and the merge of these nodes is known to be nonempty, return "no"; else return "unknown" (see Fig.4)

To answer ?[a disjoint-from b]: If no $a_i \in A$ has an ancestor in B and no $b_j \in B$ has an ancestor in A, return "yes"; if some nodes in A have an ancestor in B and/or some nodes in B have an ancestor in A, and the merge of these nodes is known to be nonempty, return "no"; else return "unknown" (see Fig.5).

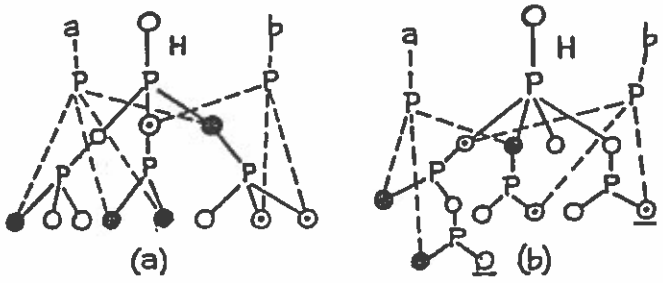

Fig. 4. Answering?[b part-of a] in a closed graph. In (a) the answer is "yes" (Note that one part of b has no ancestor in A, but is partitioned into parts belonging to a). In (b) the answer is "no" if the merge of the underlined nodes is known to be nonempty, and "unknown" otherwise.

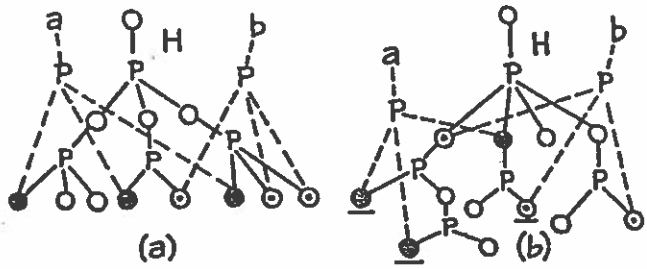

Fig. 5. Answering ?[a disjoint-from b] in a closed graph. In (a) the answer is "yes". In (b) the answer is "no" if the merge of the underlined nodes is known to be nonempty, and "unknown" otherwise.

In [10] the completeness claim is made precise and proved. It is assumed that the merge of a set of nodes $A = \{a_1,\ldots,a_m\}$ is known to be nonempty iff[$(\sqcup a_1\ldots a_m) \neq \Theta$] is a logical consequence of the graph, the assumed properties of "part-of", and a set of "nonemptiness assertions" of the form [$a \neq \Theta$] about some of the nodes of the graph. It is shown that the predicate "known to be nonempty" is efficiently decidable when so defined. The given question-answering methods are then shown to be implementable in linear space-time relative to the number of edges of the closed graph. Heuristic methods are discussed which can often give nearly constant question-answering times for closed graphs.

Similar methods can be used to compute the "disjoint merge" of two parts, i.e., a set of disjoint parts whose merge equals the merge of the given parts (retain elements of A∪B which do not have ancestors in B), the overlap of two parts (retain elements of A with ancestors in B and vice versa), and other functions over tuples of parts. Thus questions involving such functions could also be answered.

Needless to say, reasoning about parts cannot in general go very far on the basis of "part-of" predications or partitionings alone. In the case of physical objects, for example, knowledge about the mode of connection and relative position of parts is indispensable. But the fact remains that people can answer questions such as "Is a toe part of a leg?" or "Is a spark plug part of an automobile engine?" with consummate ease. The proposed methods may help to equip machines with comparable abilities.

One serious limitation of the proposals is that they do not allow for disjunctive parts relationships (even "alternative viewpoints" are logically conjoined). A second is that only the structure of particular objects has been considered, whereas much human parts knowledge is generic. A third

is that no provision has been made for avoiding individual mention of multiple parts of the same type, such as a centipede's legs or a human being's neurons.

In [10] some extensions to generic graphs and graphs containing representations of sets of parts are sketched. The chief problem with generic graphs is that they are dispersed over (possibly overlapping) type hierarchies. For example, only elephant-specific partitioning assertions, such as those subdividing the proboscis, would be associated directly with the concept of an elephant. Most of the remaining partitioning assertions, such as that dividing the body into head, neck, torso, tail, and limbs would be associated with higher-level concepts, such as that of a legged animal. Assuming that the fragments form a closed graph when brought together at corresponding nodes, the question-answering methods of this section can be adapted to generic graphs. The main requirement is the availability of methods for identifying corresponding nodes. This problem is discussed in a more general context in the next section.

The introduction of nodes representing sets of parts into parts graphs complicates both their syntax and use. Additional partitioning relations are needed (e.g., for partitioning an object into a set of parts, and for partitioning the sets themselves), and methods must be provided to deal with more complicated questions, such as "Is every leg part of some body segment?" Nevertheless, the definition of a closed graph can be modified to allow for sets of parts, and linear or sublinear methods can be provided for answering several types of questions for such graphs.

4. Property inheritance

When one describes a type of object, such as a robin, in some representation system, one would like many of its anatomical characteristics to be "inherited" from the description of a superordinate concept; for example, the position of the beak on the head or of the wings on the body should transfer smoothly from "bird" to "robin".

Hayes [3,9] has discussed two alternative network methods for facilitating property and relationships inheritance from parts of more general to parts of less general kinds of objects. The first method involves connecting corresponding parts with "binders" which then serve as inheritance paths. The second method actually uses the same nodes for corresponding parts of the two kinds of objects; however, the "depictions"[4] for the two kinds of objects provide access to different sets of propositions about the parts. The latter proposal was adopted and given a logical interpretation in terms of shared variables in [15]. However, a serious flaw has become apparent in this scheme. The telescoping of nodes, carried all the way down to the level of instances, assigns all instances of a concept to a single node, and similarly for corresponding parts of those instances. As a result, relationships between distinct instances or parts of distinct instances, such as "Polly's beak is larger than Tweety's" cannot be stated. Even above the level of instances, the sharing of nodes prevents expression of relational propositions such as "The African elephant has larger ears than the Indian elephant", "Seagulls recognize each other by the colouration of their eyelids", "Dogs hate cats" (assuming "dog" and "cat" have a common superordinate concept), "No two dollar bills have the same serial number", etc.

The key to a logically respectable alternative which avoids these difficulties lies in the use of functions. For example, suppose that h is the Skolem function determined by the statement "Every higher animal has a head". Suppose further that the head of the universally quantified robin y has been identified with the value (h y) and similarly, that the head of the universally quantified bird x has been identified with the value (h x). If other corresponding parts have been similarly linked via their functional names, it is clear that parts relationships can be trasferred from parts of the bird to parts of the robin as easily as the corresponding universally quantified variables can be matched and the function values dependent on those variables located.

Locating values of functions can be made efficient by attaching a "function table" to each node, in which known values (nodes) of functions applicable to that node are indexed by function name. This scheme resembles Hayes' "binder" scheme, but the correspondence between a part of a subordinate concept and a part of a superordinate concept several levels above it are established in a single, trivial matching operation, instead of a series of binder traversals. Some complications are described in [10], including those arising from the presence of nodes representing sets of parts, and from the application of a function at different levels of a parts hierarchy (e.g., application of a "heart-of" function at the level of the whole animal and at the level of the animal's chest).

Any system for representing parts correspondences explicitly, whether it is based on shared nodes, binders, or functional indexing, presupposes a method of establishing these correspondences in the first place. Hand-coding such correspondences

---

[4] Essentially, a "depiction" is a list of assertions which are specific to a particular (kind of) object, accessible via the name of that object.

ces ought to be a temporary expedient, to be replaced eventually by a method of inferring and representing them on the basis of arbitrary sets of parts relationships such as might be obtained from natural language input. To see the problem and an approach to a solution, suppose that a natural language system is told for the first time that "A bird has a tail". The system cannot assume that "tail" is translatable as a logical function; for this would lead to the unwarranted (and false) conclusion that "a bird with two tails" is a contradiction in terms. A plausible translation is

(∀x)(∃y)[[x bird] → [[y tail] & [y part-of x]]].

At the same time, a conjecture that y is normally the only tail should be stored, since the wording would more likely have been "at least one tail", "one or more tails", or something to that effect, had that been the interpretation intended by the informant. Skolemization of y gives

(∀x)[[x bird] → [[(t x) tail] & [(t x) part-of x]]].

If told subsequently that "A magpie has a long tail", how could the system identify "a...tail" with t applied to the universal "magpie" variable instead of creating a new Skolem function? It would at least have to check for a possible referent of "a...tail" among the parts of concepts superordinate to "magpie". After finding such a part at "bird", and verifying that this is probably the only part of a bird satisfying the predicate "tail", it could transfer the name of the Skolem function into the new context, at least tentatively.

In general, a wider search of prior instantiations of "tail" would be required since the first occurrence need not have been at the superconcept level, but might instead have been at the subconcept level, or in connection with a concept which is neither a superconcept nor a subconcept of "magpie" (e.g., "dog"). If an occurrence of "tail" is found for a subconcept of "magpie", then the name of the corresponding Skolem function can be transferred to the "magpie" context, provided that the newly mentioned "tail" is likely to be the magpie's only "tail" (in which case it is also the subconcept's only "tail"). If an occurrence of "tail" is found for a concept such as "dog", then again the Skolem function can be transferred to the "magpie" context provided that both the magpie's tail and the dog's tail are likely to be unique (this policy prevents later naming conflicts at the level of common superconcepts of "magpie" and "dog").

This approach is also workable for compound identifying descriptions. For example, the information that a perch has a spiny fin on its back might be rendered as

(∀x)(∃y)[[x perch] → [[y fin] & [y part-of x]
& [y joins (back-of x)] & [y spiny]]].

If a part can be found for the superconcept "fish" which uniquely satisfies "is a fin, is part of the fish, and joins the back of the fish", then its Skolem function name can be used in the Skolemization of the perch's fin y. Much the same as before can be said about occurrences of parts satisfying the description in question but belonging to subconcepts or unrelated concepts.

All this presupposes a method for locating parts satisfying given descriptions, i.e. associative accessing. This topic cannot be covered here; Hayes' associative accessing methods are easily modified for use with semantic nets in which corresponding nodes are linked via their Skolem function names, rather than being telescoped or connected with binders.

5. Concluding discussion

The seemingly trivial problem of extracting parts relationships from sets of partitioning assertions was seen to be quite difficult. The suggested methods for "closed graphs" provide a partial solution.

Effective property inheritance requires effective ways of establishing node correspondences. A method of establishing correspondences through names of Skolem functions was suggested which overcomes a difficulty with Hayes' node-sharing method.

Many problems in generalizing the proposed methods remain to be solved, such as that of providing detailed inference algorithms for extracting "part-of" relationships from fragmented generic parts graphs, or for establishing node correspondences on the basis of composed functions such as (f (h x))). Moreover, several important problems with parts have not been touched on here, such as the problem of inferring the mode of connection and relative position of parts of physical objects, the problem of determining the number of parts of a given type of a given object (counting the known instances only provides a lower bound on that number), and the problem of accessing parts associatively on the basis of compound descriptions (Hayes considered unitary descriptions only).

In view of the centrality of parts relationships in human thought, these are important areas for future research.

Acknowledgements

Alan Covington and Randy Rawson contributed to this paper through discussions and their work on a semantic net system. Pat Hayes' careful refereeing provided much useful guidance. The research was supported by an Alexander von Humboldt Fellowship and by Operating Grant No. A8818 of the National Research Council of Canada.

References

[1] Clowes, M.B. (1969). "Transformational grammars and the organization of pictures", in Grasselli, A., Automatic Interpretation and Classification of Images, Acad. Press, New York, pp. 43-77.

[2] Winston, P. (1970). "Learning structural descriptions from examples", Ph.D. Thesis, MIT, TR-76, Cambridge, MA.

[3] Guzman, A. (1971). "Analysis of curved line drawings using context and global information", Mach. Intell. 6, Meltzer, B. & Michie, D. (eds.), American Elsevier, New York, 325-375.

[4] Barrow, H.G., Ambler, A.P., & Burstall, R.M. (1972). "Some techniques for recognizing structures in pictures", in Watanabe, S. (ed.), Frontiers of Pattern Recognition, Academic Press, New York, pp. 1-29.

[5] Kanaff, S. (1972). "Pattern cognition and the organization of information", in Watanabe, S. (ed.), Frontiers of Pattern Recognition, Academic Press, New York, pp. 193-222.

[6] Raphael, B. (1968). "SIR: A computer program for semantic information retrieval", in Minsky, M.L. (ed.), Semantic Information Processing, MIT Press, Cambridge, MA, pp. 33-134.

[7] Minsky, M. (1975). "A framework for representing knowledge", in Winston, P. (ed.), The Psychology of Computer Vision, McGraw-Hill, New York.

[8] Hayes, Philip J. (1977a). "On semantic nets, frames and associations", Proc. 5th Int. Joint Conf. on Artificial Intelligence, MIT, Cambridge, MA, Aug. 22-25, pp. 99-107.

[9] Hayes, Philip J. (1977b). "Some association - based techniques for lexical disambiguation by machine", TR25, Comp. Sci. Dept., University of Rochester, Rochester, N. Y.

[10] Schubert, L.K. (1979). "Representing and using knowledge about parts", in preparation as Computing Science Tech. Note, University of Alberta, Edmonton, Alberta.

[11] Bunt, H.C. (1978). "A formal semantic analysis of mass terms and amount terms", Amsterdam Papers on Formal Grammar, Vol. 2 (Proc. 2nd Amsterdam Symp. on Motague Grammar & Related Topics, Amsterdam, Jan. 9-13, 1978).

[12] Grossman, R.W. (1976). "Some data-base applications of constraint expressions, LCS TR-158, MIT Lab. for Computer Science, MIT, Cambridge, MA.

[13] Fahlman, S.E. (1977). "A system for representing and using real-world knowledge", AI-TR-450, AI Lab., MIT, Cambridge, MA.

[14] Schubert, L.K. (1976). "Extending the expressive power of semantic networks", AI Journal 7, pp. 163-198.

[15] Schubert, L.K., Goebel, R. & Cercone, N. (1979). "The representation and organization of a semantic net for comprehension and inference" in Findler, N.V. (ed.), Associative Nets - The Representation and Use of Knowledge by Computers, Academic Press; Preliminary version: Techn. Rep. TR78-1, Dept. of Computer Science, University of Alberta, Edmonton, Alberta.