253

0-315-12564-0

60475

PERMISSION TO MICROFILM — AUTORISATION DE MICROFILMER

• Please print or type — Écrire en lettres moulées ou dactylographier

Full Name of Author — Nom complet de l'auteur
Mary-Angela Papalaskaris

Date of Birth — Date de naissance	Country of Birth — Lieu de naissance Greece

Permanent Address — Résidence fixe
Dept. of Artificial Intelligence, U of Edinburgh
Forrest Hill, Edinburg. Scotland
EH1 2QL

Title of Thesis — Titre de la thèse
Special Purpose Inference Methods

University — Université
U of Alberta

Degree for which thesis was presented — Grade pour lequel cette thèse fut présentée
M Sc

Year this degree conferred — Année d'obtention de ce grade 1982	Name of Supervisor — Nom du directeur de thèse L. K. Schubert

Date Sept 30/82	Signature Mary Angela Papalaskaris

NL-91 (4/77)

THE UNIVERSITY OF ALBERTA

Special Purpose Inference Methods

by

Mary-Angela Papalaskaris

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE

OF Master of Science

Department of Computing Science

EDMONTON, ALBERTA

Fall 1982

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled Special Purpose Inference Methods submitted by Mary-Angela Papalaskaris in partial fulfilment of the requirements for the degree of Master of Science.

L.K. Schubert
Supervisor

[illegible]

Bernard Linsky

[illegible]

Date Sept. 16, 1982

ABSTRACT

A semantic net system in which knowledge is topically organized around concepts has been under development at the University of Alberta for some years. This thesis attempts to extend the inference capabilities of the resolution based theorem-prover of the system, so as to enable it to efficiently answer some of the questions that people answer "without thinking". Two problems dealt with are, first, generalizing and formalizing to a certain extent earlier results on taxonomic inference (e.g., for part-of relationships or taxonomies of types) and, second, providing a numerically coded spatial representation for colours which allows constant time compatibility checking of various hedged and unhedged colour terms.

ACKNOWLEDGMENTS

I would like to thank my supervisor, Len Schubert, for actively participating in the work of this thesis. The guidance, criticism and encouragement that he provided steadily through the course of my studies has been invaluable to me.

I would also like to thank the other members of my committee, Stan Cabay, Bernard Linsky and Francis Chin, for reading and commenting on my thesis. Bernard Linsky and Jeff Pelletier gave me many useful insights in connection with logic and underivability, during some of our earlier discussions.

The financial support from the Department of Computing Science and from NSERC grant A8818 is gratefully acknowledged.

I am indebted to John de Haan, Darrell Makarenko, Lisa Higham, Bob Bramwell and Jen Penny for the various forms of help and moral support they have given me.

Above all I want to thank my friend, Leo Hartman for being critical about everything, yet listening to all I had to say and being so supportive at the same time.

Table of Contents

Chapter Page

1. Introduction 1

1.1 Inference about Parts 4

1.2 Inference about Colours 11

2. Special Purpose Inference 14

3. Inference About Parts 22

3.1 The part-of relation, partitionings and P-graphs 22

3.2 Simple graphs and hierarchies 29

3.3 Closed P-graphs 42

3.4 Semi-closed P-graphs 45

4. Inference about Colours 72

4.1 Predicates modified by hedges 74

4.2 The cube model 79

4.3 The cylinder model 82

4.4 Inference methods 84

5. Conclusions 92

6. Bibliography 95

List of Figures

Figure Page

1. A simple non-closed P-graph 10
2. General P-graphs and hierarchies 32
3. Some examples of semi-closed P-graphs 47
4. The semi-closed P-graph G 49
5. G is a fully consistent semi-closed P-graph 51
6. Diagram for lemma 4.1 55
7. Diagram for lemma 4.2 57
8. Diagram for lemma 4.3 60
9. Diagram for lemma 4.4 61
10. Diagram for lemma 4.6 64
11. Diagram for lemma 4.7 66
12. Diagram for the corollaries to theorem 4.2 71
13. Strong incompatibility links between colour terms 78
14. Views of the colour cube 80
15. The purity-dilution-hue colour space 83
16. The cone model 84
17. Regions of colours A and B 90
18. Relationship between two literals 91

1. Introduction

A semantic net system in which knowledge is topically organized around concepts has been under development at the University of Alberta for some years [Schubert, Goebel & Cercone 1979, Covington & Schubert 1980]. The system is capable of automatic topical classification and insertion of modal logic input sentences, concept and topic oriented retrieval, and property inheritance of a rather general sort.

Inferences are generally handled in the system by a resolution theorem prover. Resolution was chosen because a canonical representation was necessary for storing propositions and because disjunctive clauses facilitate the kind of retrieval of relevant propositions that is required for question answering. Disjunctive clauses usually contain relevant facts, while conjunctive clauses, say, do not always contain literals that should be connected in any manner. For example, the clause elephant(x)$\vee\neg$animal(x) reflects a natural connection between the concepts "elephant" and "animal" which ought to be exploited in retrieving relevant propositions for question answering, while a clause such as animal(Clyde)$\wedge$grey(Clyde) makes a connection that cannot be of any use for these purposes. Moreover, disjunctive clauses retain the symmetry in the meaning of utterances, such as "if x is a cat then x is not a dog", which might, in a natural deduction system be translated into the clause $cat(x) \Rightarrow \neg dog(x)$ while it is logically equivalent to $dog(x) \Rightarrow \neg cat(x)$. In a resolution theorem prover its representation is $\neg cat(x) \vee \neg dog(x)$. For all these reasons resolution was chosen.

This thesis is concerned with extending the inference capabilities of the system being developed, to enable it to answer some of the kinds of questions which people can answer "without thinking". Such special-purpose mechanisms are essential adjuncts to any general inference system based on symbolic logic. They may be used by a resolution based theorem prover to evaluate propositions or to extend the types of allowable resolutions. These operations can often reduce long portions of resolution proofs to single steps.

Specifically, evaluating propositions is useful as a means of eliminating literals that cannot contribute to the proof. For example, suppose literal P(a,b) expresses that "a is part of b", and can be proved true or false by a special reasoning subsystem. Suppose further that the clause $P(a,b) \vee Q(z)$ has been generated in the course of a proof. Then if P(a,b) is proved true, Q(z) can be removed from the clause (since it does not contribute any new information) and if P(a,b) is proved false, it can be itself deleted from the clause. Without special methods of evaluation, elimination of these literals might require many resolution steps.

Extended resolution allows literals that are provably incompatible to be resolved. For example, it should be possible to resolve {red(x),green(x)} once it has been determined by the special reasoning subsystem that "red" and "green" are incompatible predicates; or {elephant(Clyde),¬animal(x)} could be resolved by a specialized subsystem that can reason about generalization. Again, many resolution steps might be required to detect such incompatibilities without special inference methods.

Special purpose systems have been under development for many types of inference. Previous work on this subject in various areas is surveyed in chapter 2. This thesis has concentrated on efficient inference of inclusion and disjointness relationships in quasi-hierarchies of parts and for (possibly modified) colour predicates.

Most of the work presented here also appears in [Papalaskaris & Schubert 1981] and [Papalaskaris & Schubert 1982]

1.1 Inference about Parts

Consider the relative ease with which people can solve "problems" such as

(1) Does a dog have a spine?

(2) Is sulphur a precious metal?

in comparison with a problem such as the following:

(3) The members of a certain group of people have the following properties. If any one member of the group envies another member, and that other member envies a third, then the first also envies the third; and if any two members of the group envy the same person then they love each other. Al, Bill, Cecil and Didi are members of the group, and Al envies Bill, Cecil envies Bill, and Didi envies Cecil. Does Didi love Al?

(1) and (2) can be solved "without thinking", but (3) requires some deliberate thought. (Of course some mental effort is required merely to understand the problem, but some additional effort is required to solve it). Yet from a logical point of view (1)-(3) are very much the same kinds of problems, namely problems of inferring inclusion or disjointness relationships in taxonomic structures, and (2)

probably requires as many inference steps as (3) if it is answered by applying a uniform proof procedure (resolution based or otherwise) to a set of axioms describing a taxonomy of substances. Note that it would be implausible to suppose that people recall that sulphur is not a precious metal as an explicitly known fact, rather than an inference.

This suggests that (1) and (2) are solved by very efficient special-purpose methods that exploit the structure of taxonomies, while (3) is solved by more laborious general methods. Which type of method is used is a matter of familiarity: if we have reflected on the relationships between Al, Bill, Cecil and Didi - or a much larger group - at length, and the relationships can be viewed taxonomically, we will eventually assimilate the taxonomy in the same way we have assimilated the taxonomies of animal parts, or the taxonomies of substances.

Even if these comments are psychologically incorrect, they serve to make a practical point concerning AI systems: if such systems are to use taxonomic knowledge with the same ease as humans, they will have to be equipped with special-purpose inference mechanisms for doing so instead of relying on general problem-solving strategies such as recursive problem reduction. This as an important challenge in AI, given the ubiquity of parts hierarchies and concept hierarchies in virtually all fields of knowledge. A great many past and present AI systems have made allowance for hierarchies of various kinds. For example, Raphael's SIR [1968] effectively exploited the transitivity of part-of relationships and Quillian's Semantic Memory [1968] organized concepts as "subset-superset" taxonomies; (neither, incidentally, paid much attention to possible exclusion relationships among subparts or subconcepts). More recently Philip Hayes [1977] has developed network structures and techniques for using knowledge about part-of relationships, and Fahlman [1977] has made proposals for reasoning about "tangled" overlapping concept hierarchies in his NETL system.

A shortcoming of much of this work has been the lack of any attempt to analyse the adequacy of the proposed methods. What types of questions can they answer? Are the answers they derive reliably correct? To what classes of hierarchies or "tangled" hierarchies do they apply? Will an answer be derived within a reasonable length of time?

In an attempt to remedy this shortcoming, Schubert [1979; 1980] studied sets of partitioning assertions of the form [a P $a_1 \ldots a_n$], meaning that object a is partitioned into disjoint parts $a_1, \ldots, a_n$, with P defined in terms of a part-of relation "$\subseteq$". Such sets of assertions correspond to arbitrarily "tangled" hierarchies. One of the first findings was that in this general case even the simplest questions, such as ?[a part-of b] can be forbiddingly difficult to answer (co-*NP*-complete). This is surprising if one is inclined to believe in the generality and efficiency of "label-propagation" methods. The next step was to define a class of P-graphs (where a P-graph is essentially a set of partitioning assertions) which avoids the intractability of unrestricted P-graphs, yet permits "tangled hierarchies" of sufficiently general kinds to be useful in practical inference problems. To this end a <u>closed</u> P-graph was defined, roughly as a set of P-assertions which (directly or indirectly) decompose all parts mentioned into a subset of a fixed set of ultimate parts. Graphically, closed P-graphs have the appearance of overlapping partitioning hierarchies in which all downward paths terminate at the leaves of some common "main" hierarchy whose root represents the merge of all parts mentioned. Thus closed P-graphs can represent "multiple views" of the same object.

Closed P-graphs admit very efficient (linear or sublinear) inference methods for questions of type ?[a part-of b] or ?[a disjoint-from b] while still allowing some tangling of hierarchies [Schubert 1979]. Moreover, these methods are provably complete. This partially solves the problem originally addressed.

The first objective of chapter 3 is to formalize the definition of P-graphs, hierarchies and closed P-graphs and, in general, to shore up the theoretical foundations of the earlier work by supplying axioms for the part-of relation, stating some immediate consequences and carefully defining various kinds of P-graphs and relevant notions. Model-theoretic techniques are developed which provide a basis for proving inference algorithms for P-graphs correct and complete.

The second objective is to liberalize the notion of a closed P-graph so as to provide a more flexible representation for parts structures without sacrificing inference efficiency. It was noted in [Schubert 1979] (and proved in [Schubert 1980]) that an arbitrary P-graph can in principle be converted to a logically equivalent closed P-graph. However, the equivalent closed graph may be much larger than the original open graph.

Consider the following situation. Suppose that a person (or computer) knows who the faculty members $a_1,...,a_{15}$ of certain computer science department C are, and also knows that the department divides organizationally into a set of non-overlapping administrative units, viz. a chairman c_1, an advisory committee c_2, a library committee c_3, a colloquium committee c_4, and graduate and undergraduate committees, c_5 and c_6. He/she/it doesn't know the current chairman or constitution of the committees (perhaps after being out of touch for a year). This information, in the form of a

P-graph, is shown in Fig. 1.[1] Note that not all paths in this graph terminate at the leaves of a *common* main hierarchy (though all terminate at the leaves of one of the two main hierarchies), so that the graph is open. Conversion of the graph to a closed graph would introduce 90 new parts in addition to the 22 already present! (We are ignoring constraints such as that c_1, the chairman, must equal one of $a_1,\ldots,a_{15}$ and that each c*i* must consist of a subset of the a*i*, for simplicity). This is because conversion to a closed graph produces an "artificial" integration of the alternative viewpoints in the original graph, introducing nodes for all the ways in which parts in one view may overlap with parts in the other. The question-answering algorithms rely on the presence of these overlap nodes. Yet it is obvious that part-of questions and disjointness questions can be answered very easily for the original graph: everything is part of C, and *within* each of the two partitionings all distinct parts are disjoint while for parts a*i*, c*j*, taken from both partitionings, the correct answer to ?[a*i* part-of c*j*] or ?[a*i* disjoint-from c*j*] is "unknown". A reduction to closed graphs would only obscure the logic of the requisite reasoning process.

----

[1] For the purposes of this illustration, C is to be interpreted as the (disconnected) physical whole composed of the department members, *not* as a set. Thus the a*i* and c*i* are *parts* of C, not elements or subsets.

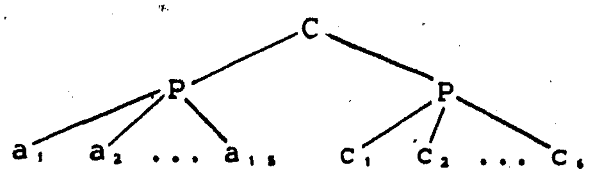

Fig.1 A simple non-closed P-graph

A similar example would be provided by a partly functional and partly anatomic representation of brain structure in which the postulated functional subsystems (say, perceptual subsystems, motor control subsystems, short term and long term memory, language understanding subsystems, etc.) cannot be reliably identified with particular anatomic structures. A computer encoding of such incomplete knowledge should not require introduction of identifiers and partitioning assertions for all possible overlap parts corresponding to the two views. Examples of this type, involving poorly integrated alternative views of some physical or abstract entity are readily constructed, and could easily occur in an AI system, particularly one which is fed its knowledge piecemeal.

This motivates the introduction of recursively defined semi-closed P-graphs. A <u>semi-closed</u> P-graph is either a closed P-graph, or a semi-closed P-graph with another semi-closed P-graph attached to it by one of its main roots. Clearly the P-graph of Fig. 1 is a semi-closed P-graph, since it consists of the <u>closed</u> committee-structure subgraph attached <u>by its main root</u> C to the closed faculty-roster subgraph. Efficient complete algorithms for answering part-of and disjointness questions on the basis of semi-closed P-graphs are developed and their correctness and completeness is shown.

The class of semi-closed P-graphs is probably as large a class of P-graphs as is needed for most practical applications to taxonomic structures, and as can be easily mechanized, as far as answering part-of and disjointness questions is concerned. P-graphs can also represent partitionings based on relations other than the part-of relation, as long as these relations satisfy the assumed properties of "part-of". This includes the subset-of relation and the subconcept-of (IS-A) relation commonly used in taxonomies of types.

1.2 Inference about Colours

As indicated in the previous section, procedures that efficiently infer taxonomic relationships can be built, making it possible to directly "resolve" literals of the form $\{\neg M(x), m(x)\}$, when M subsumes m, and of the form $\{M(x), N(x)\}$ when M and N are incompatible. Thus, the resolution-based theorem prover of the semantic net should be capable of directly resolving not only complementary pairs of literals such as {elephant(Clyde), ¬elephant(x)}, but also incompatible pairs such as:

{elephant(Clyde), ¬animal(x)},

{elephant(Clyde), canary(x)},

{yellow(Clyde), grey(x)},

{(sort-of elephant)(Clyde), ¬(sort-of animal)(x)},

{(sort-of tan)(Clyde), ¬(sort-of brown)(x)},

{(sort-of yellow)(Clyde), (sort-of blue)(x)}.

Originally, it appeared that all such examples of "extended resolution" could be handled by means of the algorithms for parts. However, certain classes of colour resolutions, namely ones involving the very common, yet troublesome, "sort of" modifi~~er~~ on colours (exemplified by the last pair of literals above), do not easily lend themselves to lattice methods.

The problem stems from the fact that it seems impossible to find a graphical representation of colour terms, in which links capture the relationship of anything other than the two terms involved. This means that the graph will have $O(n^2)$ edges.

In chapter 4 a special group of algorithms is presented, based on a three-dimensional numerically coded representation. This solution seems to deal with the problem of inference of subsumption and incompatibility relationships between (possibly hedged) colour terms neatly.

The representation is based on the observation that different colours can be constructed by picking a rainbow hue and "adding white" or "adding black".

2. Special Purpose Inference

Although formal systems for theorem proving have been available for many years, it is now generally accepted that proof procedures that are guaranteed to find a proof eventually are often of no practical use. Even in a formal, well-defined domain, such as mathematical theorem proving there is a need for some more specific knowledge about the problem to guide the proof. In reasoning about everyday facts, even more specialized types of inference are needed, that take advantage of the structure of some of the relationships, allowing the use of special representation and inference methods. For example, in reasoning about taxonomies, the transitivity of some of the relations involved must be exploited, if inferences are to be made within a reasonable time frame. The use of uniform inference methods for these classes of relationships is too slow. Special representations for such relations, coupled with methods that exploit their structure in inferring facts not explicitly stored are essential. Indeed, several systems have recognized this need.

This chapter presents a survey of some of the work aimed at finding such representations and methods, mainly for reasoning about taxonomic, temporal and spatial relations. These relations arise a great deal in everyday inferences and are almost instantaneous for people, to the point that some appear as if they are a retrieval of an explicitly known fact (but this is very unlikely in view of questions such as "is sulphur a precious metal?", as mentioned in the introduction).

Lindsay [1973] argues in favour of *ad hoc* systems, claiming that generality can be retained by seeking a general schema of <u>devising</u> special purpose representations, as one departs from a general schema of representation. He and his students formulated a set of properties of relations, such as reflexivity, functionality, transitivity, etc. They devised methods for answering questions about relations, which were defined in terms of a subset of this set of properties. For example, an equivalence relation is reflexive, symmetric and transitive. Ways to deal with the interaction between several relations are also discussed. One important finding was that only in cases involving the transitive property are there examples of non-trivial inferences to true facts, i.e., to facts not explicitly stored in the representation. Although the approach taken was indeed general in many respects, the nine properties that he chose could be reformulated so that more of the useful relations can be defined in terms of these properties. For example it seems impossible to define a partial ordering directly in terms of the given properties.

Quillian's Semantic Memory [1968], whose motivation was partly to model the human brain, was capable of representing specialization. The emphasis, however, was on retrieval of relevant information to be used in inferencing, rather than using the representation as a reasoning aid.

Another early system, Raphael's SIR [1968] chose a number of relations (set inclusion, part-whole, numeric quantity for parts, set membership, left to right and ownership), giving each a detailed description of "purpose, method and procedure". "Procedure" describes the interaction with other functions; for example, it describes how part-of interacts with set relations (containment and inclusion) and with equivalence. Raphael uses "models" to model meanings of sentences, which are to be understood as internal representations of nodes and links that contain the "meaning" of the sentence. These are recognized as distinct from models in the logical sense, as they are used in chapter 3. In fact, they have almost the opposite meaning: Raphael defines a model as a symbolic representation of the real world entities and relations. The semantics of the symbolic representation are thus yet to be defined. One drawback of SIR and of Quillian's system is their inability to represent negative information, so "unknown" is often the answer when a "no" would have been preferred.

More recent efforts have shown greater concern for efficiency. Fahlman [1977] in his parallel marker propagation network describes a way of representing practically all that Raphael or Quillian could represent, in a network that lends itself to very fast parallel seaches. He also represents "splits", i.e., disjointness in specialization hierarchies, and allows for "tangled hierarchies" which can represent multiple views. It is still not possible to represent splits with respect to parts, rather than specialization. More important is the lack of a systematic way to deal with inference. Many examples are given on how to handle specific problems of reasoning and representation, but no indication of how the system decides to adopt a certain approach.

This observation applies also to Philip Hayes' [1977] CSAW system which is meant to facilitate associative searches of context for potentially appropriate senses of ambiguous words. CSAW deals also with the representation of multiple subparts of similar but different entities, while the main problem addressed is that of property inheritance.

McSkimin and Minker [1979] use a representation for taxonomies that is very similar to the one used in this thesis (and also in [Schubert 1979; Schubert 1980; Papalaskaris & Schubert 1981]). Indeed, they can represent partitioning assertions by way of representing superset, disjointness and equality relations. The difficulty in allowing so much freedom in the representation is recognized. (In [Schubert 1980] it is shown that confirming disjointness or part-whole relationships in partitioning graphs is co-*NP*-complete). McSkimin and Minker deal with this problem by precomputing and storing the relationships between every possible pair of semantic categories in a triangular matrix. This information can then be extracted directly by a simple access to the matrix, provided such a matrix is of manageable size (the storage requirements for the matrix are $O(n^2/2)$). Unfortunately this is not the case for most applications involving real world knowledge.

Fikes and Hendrix' [1977;1979] K-Net also has the capability of representing the same types of taxonomies but their approach is informal. They show that much can be inferred from the representation, but provide only sketches of the methods for mechanizing the inferences.

Special purpose inference has had by far the greatest attention with respect to taxonomies in the AI literature. There has also been some interest in some other areas. Prior work on representation and methods for reasoning about colours is summarized in chapter 4.

Bundy [1973] describes SUMS, a theorem prover that draws on the use of diagrams in constructing proofs for arithmetic. SUMS arose from an attempt to represent the concept of an "ideal (or typical) integer", i.e., an object with all the properties of an integer (e.g., being equal to, less than, not equal to or a divisor of some other integer), but which is not any particular integer. In a "diagram" composed of "ideal integers" no spurious coincidences arise, so that anything true in the diagram is provable from the hypotheses of the theorem and anything false is not provable. Of course something may be neither true nor false in the diagram. He succeeded in proving a good proportion of simple arithmetic theorems. A problem may be created by the fact that, in order not to make assumptions about the nature of the integers involved besides those given by the set of hypotheses of the theorem, it may be necessary to break down the proof into a rather large number of cases (e.g., A=0, A≠0, A=1, A≠1, etc.).

Bundy's "ideal integers" and diagrams as they are used in proofs can be viewed as a kind of envisioning. Envisioning is the subject of a recent paper by de Kleer and Brown [1982]. Their aim is to construct correct predictions of a physical device's qualitative behaviour and to produce causal explanations of how a device functions (i.e., explanations that can be used to predict behaviour of a device under novel conditions). Their system, ENVISION, accepts input in the form of a device topology. The nodes represent the important components (down to a certain level of detail that the description confines itself to) and edges represent connections between them. There is a component library which contains component models. These describe the behaviour of components independent of the device they belong to. The descriptions consist of all potential behaviours the component can manifest and are expressed in qualitative equations called "confluences".

Envisioning determines (1) in which state(s) the overall device could be, (2) causal behaviour of the device in each of these constituent states and (3) possible transisions between pairs of global device states. Their paper concentrates mainly in providing methods·for determining the causal behaviour of the device, given the possible states in which it can be.

The work of de Kleer and Brown can be viewed as providing representation and methods for specialized reasoning about physical properties of objects. Causal, qualitative reasoning, even as applied to learned (assimilated) structures, is a very difficult task, but people usually handle it with ease. The authors have adressed several important issues. The problem of deciding whether deductions form a <u>significant</u> portion of the total effort required to describe and analyze a physical situation (i.e, is the work of causal reasoning already encoded in the evidence provided to it?) is dicussed extensively and ways are proposed which ensure that the structure and function descriptions are kept distinct.

Kahn and Gorry [1977] studied temporal reasoning and have constructed a time specialist which is capable of storing, retrieving and reasoning about events. The representation of events takes three forms: (a) organized by dates, (b) organized by special reference events and (c) organized by before-after chains. The choice of representation is left up to the user. Events can be specified with a certain degree of fuzziness and are treated as if they do not have any duration. The "fetcher" selects one or more appropriate methods for answering an incoming question, from a set of available programs to deal with particular kinds of questions making use of particular organizations of facts. It, for example, tries to find equivalent temporal specifications, use dates, etc.

A question that arises is why not convert all equivalent temporal specifications to a uniform representation, since the system has such a capability. It is not clear that the user is not in a sense helping the system make its inferences by specifying the input in a certain form. It appears that temporal reasoning systems could benefit from a more uniform representation. [Taugher & Schubert 1982]

3. Inference About Parts

3.1 The part-of relation, partitionings and P-graphs

The part-of relation, $\subseteq$, that the given methods rely upon is assumed to have the following properties [Schubert 1980]:

(i) '$\subseteq$' is a partial ordering:

$(\forall x)[x\subseteq x]$

$(\forall xy)[[[x\subseteq y]\&[y\subseteq x]]\supset[x=y]]$

$(\forall xyz)[[[x\subseteq y]\&[y\subseteq z]]\supset[x\subseteq z]]$

(ii) Existence of a unique empty object θ such that

$(\forall x)[\theta\subseteq x]$

(iii) Existence of an 'overlap' function $\cap$ such that

$(\forall xyz)[[z\subseteq(\cap xy)]\equiv[[z\subseteq x]\&[z\subseteq y]]]$

(iv) Existence of a 'merge' function $\cup$ such that

$(\forall xyz)[[(\cup xy)\subseteq z]\equiv[[x\subseteq z]\&[y\subseteq z]]]$

(v) Existence of a 'remainder' function $\sim$ such that

$(\forall xyz)[[z=(\sim xy)]\equiv[[x=(\cup z(\cap xy))]\&[(\cap yz)=\theta]]]$

(vi) Mutual distributivity of $\cap$ and $\cup$:

$(\forall xyz)[(\cap x(\cup yz))=((\cup(\cap xy)(\cap xz))]$

$(\forall xyz)[(\cup x(\cap yz))=((\cap(\cup xy)(\cup xz))]$

It is shown in [Schubert 1980] that the assumed properties of the part-of relation induce a boolean lattice on the set $\{x|x\subseteq w\}$ for any w. The following are some consequences of this that are used throughout the proofs. From now on brackets are omitted where no ambiguity arises.

(a) $(\forall xy)[[\cup xy=\cup yx] \;\&\; [\cap xy=\cap yx]]$

(b) $(\forall xy)[x\subseteq y\equiv[\cap xy=x \;\&\; \cup xy=y]]$

(c) $(\forall xyz)[y=\cap xz \equiv [(\cup y(\sim xz))=x \;\&\; (\cap y(\sim xz))=\theta]]$

(d) $(\forall x)[\cap x\theta=\theta \;\&\; \cup x\theta=x]$

(e) $(\forall xy)[(\cup x(\cap xy))=x \;\&\; (\cap x(\cup xy))=x]$

(f) The functions $\cap$ and $\cup$ are associative.

For brevity, $\cap$ and $\cup$ will be informally used as many-place functions since, from (f), this results in no ambiguity.

Partitionings are defined in terms of the part-of relation; intuitively, a partitioning assertion enumerates a set of disjoint parts of the object that it pertains to.

Definition: A <u>partitioning</u> <u>assertion</u> is of the form $[x \; Pm \; y_1 ... ym]$, $m\geq 2$, where $x, y_1, ..., ym$ can be constants or variables of the object language and Pm is the $m+1$ place predicate symbol defined by:

$$(\forall xyz)[[x \; P_2 \; y \; z]\equiv[x=\cup yz \;\&\; \theta=\cap yz]]$$

and for all $m\geq 2$:

$$(\forall y_1 ... ym)[[x \; Pm \; y_1 ... ym] \equiv$$
$$[[x \; Pm\text{-}1 \; (\cup y_1 y_2) \; y_3 ... ym] \;\&\; \cap y_1 y_2=\theta]]$$

Thus, by definition, and (i) above, the order of the y's is immaterial. "P" will be used for "P_2".

Partitioning assertions are the building units of P-graphs. Theorem 1.1 relates the part-of relation to partitionings.

Theorem 1.1: $(\forall xy)[[y \subseteq x] \equiv (\exists x')[x\ P\ y\ x']]$

Proof: Suppose $y \subseteq x$. Let $x' = (\sim xy)$, then by axiom (v):

$$[x = \cup x'(\cap xy)]\ \&\ [\cap yx' = \theta] \qquad (1)$$

From (1)

$$[x \subseteq \cup x'(\cap xy)]$$

$$\Rightarrow [x \subseteq \cap(\cup x'x)(\cup x'y)] \quad \text{(axiom (vi))}$$

$$\Rightarrow [x \subseteq \cup x'x]\ \&\ [x \subseteq \cup x'y] \quad \text{(axiom (iii))}$$

$$\Rightarrow x \subseteq \cup x'y. \qquad (2)$$

Also from (1),

$$[\cup x' \cap xy \subseteq x] \Rightarrow [x' \subseteq x\ \&\ \cap xy \subseteq x] \quad \text{(axiom (iv))}$$

$$\Rightarrow x' \subseteq x \qquad (3)$$

From our assumption and (3), using axiom (iii)

$$[\cup yx' \subseteq x] \qquad (4)$$

so, from (2) and (4)

$$x = \cup yx'$$

also, from (1)

$$\cap yx' = \theta.$$

Therefore, by the definition of partitionings, $[x\ P\ y\ x']$.

Now suppose $[x\ P\ y\ x']$, then

$\cup y x' = x$

So that, by axioms (i) and (iv)

$(\cup y x') \subseteq x \supset y \subseteq x.$ □

Theorem 1.2: For all $m \geq 2$,

$$(\forall x y_1 \ldots y_m)[[x \; P_m \; y_1 \ldots y_m] \equiv [[x = \cup y_1 \ldots y_m] \; \& \; [\bigwedge_{1 \leq i < j \leq m} \cap y_i y_j = \theta]]]$$

Proof: For $m=2$ this is true by definition. Assume it holds true for all $m<n$. Then,

$$[x \; P_n \; y_1 \ldots y_n]$$
$$\equiv [x \; P_{n-1} \; (\cup y_1 y_2) \; y_3 \ldots y_n] \; \& \; \cap y_1 y_2 = \theta$$
$$\equiv [x = \cup(\cup y_1 y_2) y_3 \ldots y_n$$
$$\& \bigwedge_{3 \leq i < j \leq n} [\cap y_i y_j = \theta \; \& \; \cap(\cup y_1 y_2) y_i = \theta]]$$
$$\& \; \cap y_1 y_2 = \theta$$

but $\cap(\cup y_1 y_2) y_i = \theta$

$$\equiv \cup(\cap y_1 y_i)(\cap y_2 y_i) = \theta$$
$$\equiv \cap y_1 y_i = \theta \; \& \; \cap y_2 y_i = \theta$$

thus

$$[x \; P_n \; y_1 \ldots y_n]$$
$$\equiv [[x = \cup y_1 \ldots y_n]$$
$$\& \bigwedge_{3 \leq i < j \leq n} [\cap y_i y_j = \theta \; \& \; \cap y_1 y_i = \theta \; \& \; \cap y_2 y_i = \theta]$$
$$\& \; \cap y_1 y_2 = \theta]$$
$$\equiv [x = \cup y_1 \ldots y_n] \; \& \; [\bigwedge_{1 \leq i < j \leq n} \cap y_i y_j = \theta]$$

By mathematical induction this follows for all $n \geq 2$. □

Corollary: For all $m \geq 2$, $1 \leq i \leq m$,

$$(\forall x y_1 \ldots y_m)[[x\ P_m\ y_1 \ldots y_m] \supset y_i \subseteq x].$$

Definition: A <u>P-graph</u> is a finite non-empty set of partitioning assertions of the form [**x** P **y** **z**...] together with non-emptiness assertions of the form [**x**≠θ], where **x** occurs in some partitioning assertion, over constants of the object language. The distinct constants of a P-graph will be referred to as the "nodes" of the P-graph.

The reader's attention is drawn to the fact that distinct object language constants correspond to distinct nodes by definition. Consequently a statement such as a=b can only express that a and b denote the same <u>object</u>, not that a and b are the same <u>node</u>. However, a metalinguistic statement such as "**x=y**", where **x** and **y** stand for metalanguage variables ranging over the nodes of a P-graph is taken to mean that **x** and **y** denote the same node. Generally statements of the object language assert facts such as "a is part of b" and so on, while those of the metalanguage are about relations between nodes of P-graphs such as the descendant relation defined below. Similarly, "=>" means "it follows that" in the object language and "=>" means "it follows that" in the metalanguage. In general, boldface symbols will be used in the metalanguage in order to stress this distinction.

It is assumed throughout this chapter that the assertions which make up a P-graph along with the part-of axioms are consistent. It should be noted, however, that a P-graph G along with the part-of axioms and the usual rules of inference of first order logic in general amounts to an <u>incomplete</u> theory of the objects of the graph; i.e., not every well formed formula w built up from constants that are nodes of G, θ, predicate symbols $\subseteq$, P*m*, =, function symbols $\cap$, $\cup$, ~ and the logical connectives is either provable ($G \vdash w$) or disprovable ($G \vdash \neg w$).

In deciding whether a statement regarding objects represented in a P-graph is a valid inference from the P-graph, the part-of axioms and the assertions that make up the P-graph are viewed as a theory (in the logical sense). Thus, the question reduces to what is a theorem for that theory.

Because of the soundness and the completeness of first-order logic, we can write

$$G \vdash \phi \text{ iff } G \models \phi$$

for any fully consistent P-graph G, meaning that ϕ is derivable as a theorem iff ϕ is true in all models of G.

A model of a P-graph is an interpretation of the parts nodes of G, the function symbols $\cup$, $\cap$, $\sim$, the relation $\subseteq$ and constant θ, such that the part-of axioms and the assertions of G are satisfied.

When dealing with knowledge representation we are interested in interpretations whose domain consists of real world objects (concrete or abstract). Thus we have "real world" models for P-graphs.

As an example consider the case where all that is known is that "b is part of a" and "c is part of a"; from this it is clearly undecidable whether or not b and c are disjoint. Although this is intuitively obvious in such simple cases, in general, showing that some question cannot be answered from the information available requires a formal argument. Typically, showing that a particular statement cannot be proved, given the information available from the P-graph, will consist of exhibiting a model of the graph in which the statement is in fact false, since, by soundness of the rules of inference, a statement can be derived from a consistent set of sentences (i.e., a theory) only if it is true in all models of the theory.

3.2 Simple graphs and hierarchies

Simply stated, a descendant of a node **x** of a P-graph is any node reachable by a downward path from **x**. Similarly, a hierarchy is a P-graph that takes the form of a tree with no more than one P-assertion about each node.

Definition: The descendant relation $\leq$ between the nodes **x**, y, and **z** of a P-graph G is defined by:

(i) $x \leq x$ for all nodes **x** of G

(ii) if **x** is a direct descendant of **y** and $y \leq z$ then $z \leq y$.

We say that x is a direct descendant of y if there is an assertion in G of the form:
[y P*k* **x** z_1...z*k-1*] for some k. (Of course, **x** need not be the first argument following P*k*).

The ancestor relation is the inverse of the descendant relation.

Definition: A leaf is a node that has no descendants other than itself.

Definition: A root is a node that has no ancestors other than itself.

Definition: A P-graph G is <u>acyclic</u> if and only if for any two nodes **x** and **y** of G, if **x**≤**y** and **y**≤**x**, then **x=y**.

The graphs that will be considered here are acyclic. Note that all the nodes that make up a cycle in a P-graph are forced to be identical in denotation, i.e., the object language formula **x**=**y** can be derived for any two nodes **x**, **y** from the assertions of the P-graph and the part-of axioms.[1] Thus cycles are easily eliminated by collapsing the cyclic nodes.

Graphs in which some node is provably empty are of relatively little practical AI interest, since they do not reflect the kinds of knowledge typically employed in "common sense" inferences. For example, people presumably do not usually hold beliefs about human anatomy (or about the anatomy of a bicycle, organization, or computer program) which logically require some of the parts about which the beliefs are held to be empty.

Definition: The <u>projection</u> of a node **x** into the leaves of a closed P-graph G is the set of those leaves of G which are descendants of **x**.

----

[1] Note the distinction between the object language formula x=y and the metalinguistic assertion of node equality **x=y**, in which the equality symbol is boldface.

Definition: A P-graph is fully-consistent iff none of its nodes can be proven to be empty.

Definition: A simple graph is an acyclic P-graph with a unique root and at most one partitioning assertion about each node.

Definition: A path is a sequence of nodes $\mathbf{x}_1,\ldots,\mathbf{x}_n$ of a P-graph such that $\mathbf{x}_{i+1}$ is a direct descendant of $\mathbf{x}_i$ for any i, $1 \leq i \leq n-1$.

Definition: A hierarchy is a simple P-graph in which there is at most one path between any two nodes.

Examples of simple graphs and hierarchies are shown in Fig. 2.

Lemma 2.1: The root of a simple graph is an ancestor of every node.

Proof: Let G be a simple graph and let **x** denote one of the nodes of G.
If **x** is the root, the property trivially holds.
Consider the set of all ancestors of **x**; note that if a node is in the set, so are all its ancestors; further, the set is finite (since G is finite).
Since G contains no cycles, it follows that there is a node

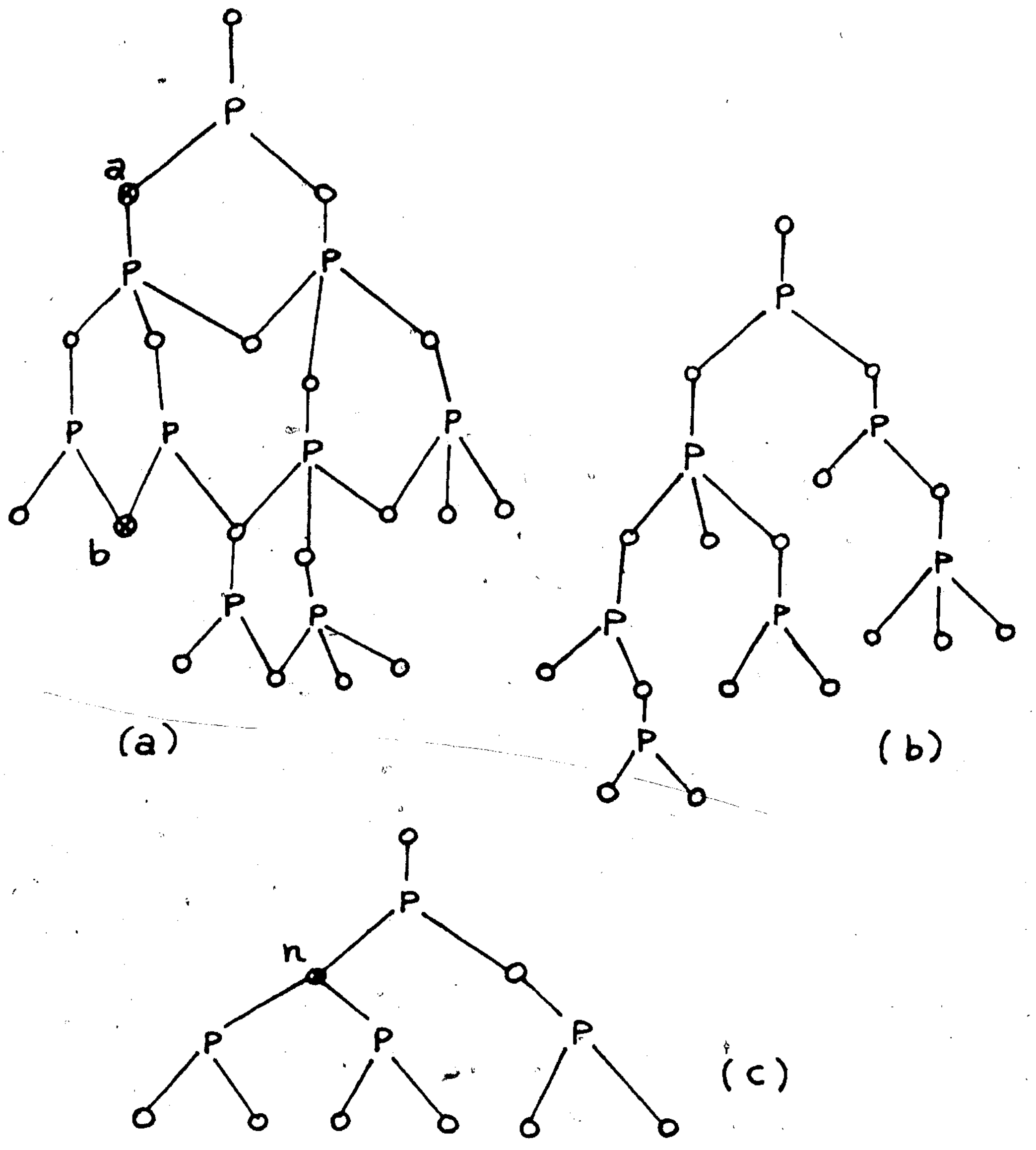

Fig. 2: General P-graphs and hierarchies. (a),(b) are simple; (b) is a hierarchy but (a) is not since there are two paths between "a" and "b". (c) is not simple, since there are two assertions about "n".

in that set that has no ancestors other than itself. This node is the (unique) root of G. □

Lemma 2.2: Let **x**, **y**, **z** range over the nodes of a hierarchy H, then, if **z**≤**x** and **z**≤**y**, then there is a path between **x** and **y** (i.e., there is a path from **x** to **y** or from **y** to **x**).

Proof: **x** and **y** are both descendants of the root (lemma 2.1); thus there is a path from the root to **z** (lemma 2.1) that contains **x**, and a path from the root to **z** that contains **y**. Since H is a hierarchy these are in fact the same path, so **x** and **y** lie on the path from the root to **z**. So there is a subsequence of this path which constitutes a path from **x** to **y** or from **y** to **x**. □

Corollary: In a hierarchy every two nodes have a least common ancestor.

Proof: From lemma 2.2 there is a path between every two common ancestors; thus they can all be ordered by the relation ≤, and since there are no cycles, there must be a least one (i.e., a common ancestor that is a descendant of every other common ancestor of the two nodes). □

So far the distinction between theorems of the object language (i.e., theorems of the axiomatic theory of parts) and theorems of the metalanguage (specifically, theorems about P-graphs) has been quite clear. In particular theorems 1.1 and 1.2 are evidently object language theorems while lemmas 2.1 and 2.2 are metatheorems. Correspondingly, the proofs in the former case are in essence sequences of object language formulas, where these object language formulas can be thought of as having the force of assertion, while in the latter case the proofs are themselves metalinguistic, and object language formulas occurring in them are <u>mentioned</u> rather than <u>used</u>. The remaining theorems and lemmas will be of the latter kind but will occasionaly contain object language proofs, as a concise way of showing that certain formulas are first-order deducible from the stipulated premise formulas (equivalently, that they are true in any model satisfying the premise formulas).

Lemma 2.3: Let H be a hierarchy with leaves $l_1,\ldots,l_n$. For all i, j, $1 \le i < j \le n$

$$H \vdash \cap l_i l_j = \theta$$

(i.e., the leaves of a hierarchy are pairwise disjoint).

Proof: It is clear that any hierarchy can be constructed by successively adding assertions that preserve the hierarchy properties. Given any hierarchy, this can be accomplished by starting with the root assertion and systematically adding at most one partitioning assertion about each of the leaf nodes to the present (intermediate) hierarchy until all the assertions of the target hierarchy are accommodated. This observation legitimizes the following form of induction.

<u>Basis</u>: The leaves of hierarchies that contain one partitioning assertion are pairwise disjoint, by theorem 1.2.

<u>Induction:</u> Consider a hierarchy H with leaf set $\{l_1,\ldots l_n\}$ and assume $H \vdash \cap l_i l_j = \theta$ for any i, j, $1 \le i < j \le n$. (1)

Now if the assertion $[l_k\ P_m\ k_1,\ldots k_m]$ is added for some k, $1 \le k \le n$[1], then

$H \vdash \cap k_i k_j = \theta$ for all i, j, $1 \le i < j \le m$. (2)

But

$$\cap l_k l_i = \theta \Rightarrow \cap(\cup k_1,\ldots k_m) l_i = \theta \text{ for all } i,\ 1 \le i \le n$$
$$\Rightarrow \cup(\cap k_1 l_i)\ldots(\cap k_m l_i)) = \theta$$
$$\Rightarrow \cap k_j l_i = \theta \text{ for all } i, j,\ 1 \le i \le n,\ 1 \le j \le m.$$

Thus from (1), $H \vdash \cap k_j l_i = \theta$ and this, together with (2), shows that the leaves of the hierarchy obtained by adding another partitioning assertion are pairwise disjoint.

Thus the result holds for any hierarchy. □

----

[1] Any assertion added that preserves the hierarchy property and retains the same root must be of this form since any assertion of the form $[x\ P_m\ x_1,\ldots x_m]$, where **x** is already in the graph but is not a leaf node, will be a second assertion about the node denoted by **x**, so that the graph would not be simple and thus not a hierarchy.

Lemma 2.4: The merge of the leaves $l_1,\ldots,l_n$ of a hierarchy H is equal to the root ~, i.e., $H \vdash (\cup l_1,\ldots l_n)=r$.

Proof: By induction on the construction of hierarchies.

<u>Basis</u>: The merge of the leaves of a hierarchy that consists of one partitioning assertion is equal to the root by theorem 1.2.

<u>Induction</u>: Refer to the previous proof.

Suppose $H \vdash (\cup l_1,\ldots l_n) = r$ and the assertion $[l_k\ P_m\ k_1,\ldots k_m]$ is added.

Then $H \vdash \cup k_1,\ldots k_m = l_k$, using theorem 1.2.

Thus $H \vdash (\cup l_1,\ldots l_{k-1}(\cup k_1,\ldots k_m)l_{k+1}\ldots l_n) = r$

so $H \vdash (\cup l_1,\ldots l_{k-1}k_1,\ldots k_m l_{k+1}\ldots l_n) = r$.

$l_1,\ldots l_{k-1}k_1,\ldots k_m l_{k+1}\ldots l_n$ are the leaves of the new hierarchy obtained from H by the addition of the assertion $[l_k\ P_m\ k_1,\ldots k_m]$. □

Corollary: The merge of the leaves of a set of hierarchies is equal to the merge of the roots.

Proof: Immediate from lemma 2.4 and the associativity of the merge operator. □

Lemma 2.5: Hierarchies are fully consistent.

Proof: Let H be a hierarchy with leaves $l_1, \ldots l_n$. We wish to exhibit a model of H that assigns non-empty parts to all the nodes of H. This is done by:

(a) constructing an interpretation of the functions, relations and the objects denoted by the nodes of H, such that the part-of axioms are satisfied, all the leaves are assigned non-empty, pairwise-disjoint objects from the domain of interpretation and the interpretation of a node is the same as the interpretation of the merge of its projection onto the leaves of H, and

(b) subsequently showing that this interpretation is a model of H (i.e., it satisfies all the assertions of H).

(a) can be accomplished by finding an interpretation α in which the functions and relations satisfy the part of axioms and that contains at least as many disjoint objects in its domain as leaves in H, and then interpreting the nodes of H as follows:

$$\alpha(l_i) \neq \alpha(\theta) \text{ and } \alpha(\cap l_i l_j) = \alpha(\theta) \text{ for all i, j, } 1 \leq i < j \leq n \qquad (1)$$

and, for any node **x** of H, if $\mathbf{x}_1 \ldots \mathbf{x}_k$ is its projection onto the leaves of H, then

$$\alpha(\mathbf{x}) = \alpha(\cup \mathbf{x}_1 \ldots \mathbf{x}_k). \qquad (2)$$

For (b) let $[a\ P_m\ a_1 \ldots a_m]$ be an assertion of H and let $A_1 \ldots A_m$ be the projections (sets) of $a_1 \ldots a_m$ onto the leaves of H, respectively.

Note that $A_i \cap A_j = \emptyset$ for all i, j, $1 \le i < j \le m$ (since otherwise there would be multiple paths between a and some leaf node). It is required to show that:

(a) $\alpha(\cup a_1 \ldots a_m) = \alpha(a)$ and

(b) $\alpha(\cap a_i a_j) = \alpha(\theta)$ for all i, j, $1 \le i < j \le m$

(a) From (2): $\alpha(\bigcup_{x \in A_i} x) = \alpha(a_i)$

so, $\alpha(\cup a_1 \ldots a_m) = \alpha(\bigcup_{x \in A} x)$

where $A = A_1 \cup \ldots \cup A_m$,

thus by the assumption (2), since A is the projection of a, $\alpha(\cup a_1 \ldots a_m) = \alpha(a)$.

(b) Again, since $\alpha(\bigcup_{x \in A_i} x) = \alpha(a_i)$,

$\alpha(\cap a_i a_j)$

$$= \alpha(\cap(\bigcup_{x \in A_i} x)(\bigcup_{y \in A_j} y))$$

$$= \alpha(\bigcup_{y \in A_j}(\cap(\bigcup_{x \in A_i} x)y)) \qquad \text{(distr.)}$$

$$= \alpha(\bigcup_{y \in A_j}(\bigcup_{x \in A_i}(\cap xy))) \qquad \text{(distr.)}$$

$$= \alpha(\bigcup_{\substack{y \in A_j \\ x \in A_i}}(\cap xy)) \qquad \text{(assoc.)}$$

by the assumption (1) $\alpha(\cap xy) = \alpha(\theta)$ for all $x \in A_i$, $y \in A_j$ (since these are leaves).

Therefore, $\alpha(\cap a_i a_j) = \alpha(\theta)$ for all i, j, $1 \le i < j \le m$. □

Lemma 2.6: For any two nodes **x**, **y** of a P-graph G, if **$x \leq y$** then G $\vdash$ **$x \sqsubseteq y$**.

Proof: The claim obviously holds for **$x=y$**, by axiom (i). For **x** a direct descendant of **y**, it holds by theorem 1.2 and axiom (iv). In general, for $\{y, z, z_1, \ldots, z_k, x\}$ a path from **y** to **x**, the claim holds by the transitivity of $\sqsubseteq$ (axiom (i)).
□

Theorem 2.1: A simple fully consistent graph is a hierarchy.

Proof: Let G be a simple fully consistent P-graph. Suppose that $a \leq b$ and that there are two paths from **b** to **a**.

path 1: $\{b, x_n, \ldots, x_1, a\}$

path 2: $\{b, y_m, \ldots, y_1, a\}$

Further assume, without loss of generality, that $x_i \neq y_j$ for all $1 \leq i \leq n$, $1 \leq j \leq m$, so that

$a \leq x_1 \leq \ldots \leq x_n \leq b$ and $a \leq y_1 \leq \ldots \leq y_m \leq b$.

Now there is only one partitioning assertion about **b** and since x_n and y_m are direct descendants of **b** (by the definition of path) they must be mentioned in that assertion.

Therefore, by theorem 1.2, G $\vdash$ $\sqcap x_n y_m = \theta$.

Now G $\vdash$ $a \sqsubseteq x_n$ and G $\vdash$ $a \sqsubseteq y_m$ (by lemma 2.6)

so G $\vdash$ $a \sqsubseteq (\sqcap x_n y_m)$, (axiom (iii))

therefore G $\vdash$ $a = \theta$, which contradicts the assumption that G is fully consistent. □

Theorem 2.2: For any two nodes **x**, **y** of a hierarchy H, if H $\vdash$ **x**$\sqsubseteq$**y** then **x**$\leq$**y**.

Proof: Suppose H $\vdash$ x$\sqsubseteq$y. For **x=y** the conclusion **x**$\leq$**y** holds trivially. Assume henceforth that **x**$\neq$**y**. Consider the projection of **x** and **y** onto the leaves of H, $\{x_1, \ldots x_n\}$ and $\{y_1, \ldots y_m\}$ respectively.

For any i, $1 \leq i \leq n$: $x_i \leq x$, so by lemma 2.6,

$$H \vdash x_i \sqsubseteq x$$

therefore H $\vdash$ $x_i \sqsubseteq y$. (1)

But,

$$x_i \sqsubseteq y \Rightarrow \cap x_i y = x_i \Rightarrow \cap x_i (\cup y_1 \ldots y_m) = x_i, \text{ (by lemma 2.4)}$$

$$\Rightarrow \cup(\cap x_i y_1) \ldots (\cap x_i y_m) = x_i.$$

Thus, from (1),

$$H \vdash [\cup(\cap x_i y_1) \ldots (\cap x_i y_m) = x_i]. \quad (2)$$

But for any j, $1 \leq j \leq m$, either $x_i = y_j$ or H $\vdash$ $\cap x_i y_j = \theta$ (lemma 2.3).

Thus from (2) and lemma 2.5, $x_i = y_j$ for some j, $1 \leq j \leq m$.

Now for all j, $1 \leq j \leq m$, $y_j \leq y$, therefore $x_i \leq y$, and from lemma 2.2, since $x_i \leq y$, it follows that there is a path between **x** and **y**.

Now either **x**$\leq$**y** or **y**$\leq$**x** (but not both). Suppose **y**$\leq$**x**.

Let [**x** P*k* $z_1, \ldots z_k$] be the partitioning assertion about **x** in H, and since **y**$\leq$**x** and **x**$\neq$**y**, assume without loss of generality that $y \leq z_1$.

From theorems 1.1 and 1.2,

$$H \vdash z_1 \subseteq x, \; H \vdash z_2 \subseteq x, \; H \vdash \cap z_1 z_2 = \theta$$

also $y \leq x \Rightarrow H \vdash y \subseteq x \Rightarrow H \vdash y = x.$ (3)

Now $y = x, \; y \subseteq z_1 \subseteq x \Rightarrow y = z_1 = x \Rightarrow z_2 \subseteq z_1 \Rightarrow z_2 = \theta.$

Thus $H \vdash y \subseteq x \Rightarrow H \vdash z_2 = \theta$ and thus, by lemma 2.5 and (3), $y \leq x$ yields a contradiction.

Therefore $x \leq y$. □

Theorem 2.3: Let **x**, **y** denote nodes of a hierarchy H. Then either:

(i) there is a (unique) path from **x** to **y** or from **y** to **x** or

(ii) $H \vdash \cap xy = \theta$

Proof: Suppose there is no path between **x**, **y**. Then by lemma 2.2 their projections onto the leaves $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$ respectively have no common element, i.e., $X \cap Y = \emptyset$.

From lemma 2.4

$$H \vdash x = \cup x_1 \ldots x_n$$

$$H \vdash y = \cup y_1 \ldots y_m$$

Now $\cap xy = (\cap(\cup x_1 \ldots x_n)(\cup y_1 \ldots y_m))$

$= (\cup(\cap x_1 y_1) \ldots (\cap x_1 y_m)$

$\vdots$

$(\cap x_n y_1) \ldots (\cap x_n y_m)\;)$

So, by lemma 2.3, $H \vdash \cap \mathbf{xy}=\theta$.

Conversely, suppose $H \nvdash \cap \mathbf{xy}=\theta$.

By the previous argument $X \cap Y \neq \emptyset$, so let $\mathbf{z} \in X \cap Y$; then $\mathbf{z} \leq \mathbf{x}$ and $\mathbf{z} \leq \mathbf{y}$. Therefore by lemma 2.2 there is a path between **x** and y, and since H is a hierarchy the path is unique. □

3.3 Closed P-graphs

Hierarchies are the most desirable form of P-graph, because parts-reasoning for hierarchies is trivial: in a hierarchy $\mathbf{x} \sqsubseteq \mathbf{y}$ is provable for any two nodes **x**, **y** if and only if $\mathbf{x} \leq \mathbf{y}$. Further, $\cap \mathbf{xy}=\theta$ is provable iff there is no path connecting **x** and y, and $\mathbf{x}=\mathbf{y}$ is provable iff $\mathbf{x}=\mathbf{y}$, i.e., all the nodes represent possibly distinct objects (theorems 2.2, 2.3). It was also shown that hierarchies are fully consistent and that any fully consistent simple graph is a hierarchy (lemma 2.5 and theorem 2.1)

Unfortunately, hierarchies can only represent some very restricted kinds of information about parts. Usually, parts knowledge about the world takes the form of "tangled hierarchies". Inferring part-of and disjointness relationships in arbitrary P-graphs is known to be co-*NP*-complete.

This problem of computational intractability is dealt with in [Schubert 1979] by converting arbitrary P-graphs to closed P-graphs. These allow greater freedom in representation of partitioning relationships than hierarchies and are still computationally tractable. Intuitively, a P-graph is closed if it is acyclic and can be built up by beginning with a simple graph and then adding a sequence of partitioning assertions, where all arguments of each assertion except possibly the first argument are already in the graph. Formally, a P-graph G is _closed_ iff it is acyclic and any two of its nodes are projectible into a common simple subgraph. A node n is _projectible_ into a subgraph H if it lies in H or if it is the root of a simple subgraph whose leaves lie in H. Note that any two nodes of a simple subgraph H are projectible into a common simple subgraph , viz., H.

It is shown in [Schubert 1980] that for every P-graph there is an equivalent closed P-graph. Inference methods are given to answer the questions ?[b part-of a] and ?[a disjoint-from b] for fully consistent closed P-graphs, in linear space and time relative to the number of edges of the closed graph.

It is proved in [Schubert 1980] that all the leaves of a fully consistent, closed P-graph belong to a single (not necessarily unique) main hierarchy whose root represents the whole entity. Such a root will be called a <u>main</u> <u>root</u> of the closed P-graph. (This result is included as a theorem below).

Theorem 3.1: If G is closed then G is fully consistent if and only if every simple subgraph of G is a hierarchy, the projection of every node into the leaves of G is unique and some subhierarchy of G contains all the leaves of G. [Schubert 1980]

Theorem 3.2: For every P-graph there is a logically equivalent closed P-graph. [Schubert 1980]

In [Schubert 1980] an algorithm is presented that will transform any P-graph to a logically equivalent closed P-graph.

Theorem 3.3: In a fully consistent closed P-graph G:

(a) all the leaves are pairwise disjoint

(b) for any node **a** and leaf **l**

$$G \vdash l \subseteq a \iff l \leq a$$

(c) for any node **a** and leaf **l**

$$G \vdash [l \not\subseteq a \vee l = \theta] \iff l \not\leq a$$

Proof:

(a) This follows immediately from theorem 3.1 since there is a subhierarchy that contains all the leaves of G and by lemma 2.3 the leaves must be disjoint.

(b) Let $\{a_1, \ldots, a_n\}$ be the projection of **a** onto the leaves of G.

$G \vdash a = \cup a_1 \ldots a_n$ (1)

$l \leq a$

$\iff l = a_i$ for some i, $1 \leq i \leq n$, by the uniqueness of the projection of a (i.e., l and a_i denote the same leaf)

$\iff G' \vdash l \subseteq a$ (from (1) along with part (a) and the assumption of full consistency).

(c) $l \not\leq a$

$\iff l \neq a_i$ for all i, $1 \leq i \leq n$

$\iff G \vdash \cap l a_i = \theta$ (from (a))

$\iff G \vdash \cup(\cap l a_1) \ldots (\cap l a_n) = \theta$

$\iff G \vdash \cap l(\cup a_1 \ldots a_n) = \theta$

$\iff G \vdash \cap l a = \theta$

$\iff G \vdash [l \not\leq a \vee l = \theta]$ □

3.4 Semi-closed P-graphs

Semi-closed P-graphs relax some of the restrictions of closed P-graphs, thus forming a larger class, allowing still greater flexibility in the representation of partitioning relationships. The tacit restriction to fully consistent graphs should be kept in mind.

Definition: A semi-closed P-graph is:

(i) a closed P-graph, or

(ii) a semi-closed P-graph that has a semi-closed P-graph attached by a main root to one of its nodes. (It is easy to see that a semi-closed P-graph, like a closed P-graph, must have at least one main root).

As semi-closed P-graphs are defined in terms of closed P-graphs, the inference methods presented here rely on those developed for closed P-graphs [Schubert 1979].

The design of the following algorithms is based on the observation that semi-closed P-graphs can be viewed as trees of closed P-graphs; each vertex represents a closed subgraph and each edge a common node of the two P-graphs (parent and child subgraphs) that it connects. Since the closed subgraphs can have at most one node in common, this will be a tree. Examples of corresponding trees for P-graphs are given in figure 3(c),(d) and (e).

Note that edges out of distinct vertices correspond to distinct nodes in the P-graph while edges out of the same vertex may represent the same node.

For any given semi-closed P-graph, it is possible to attach labels to the nodes which indicate the position in

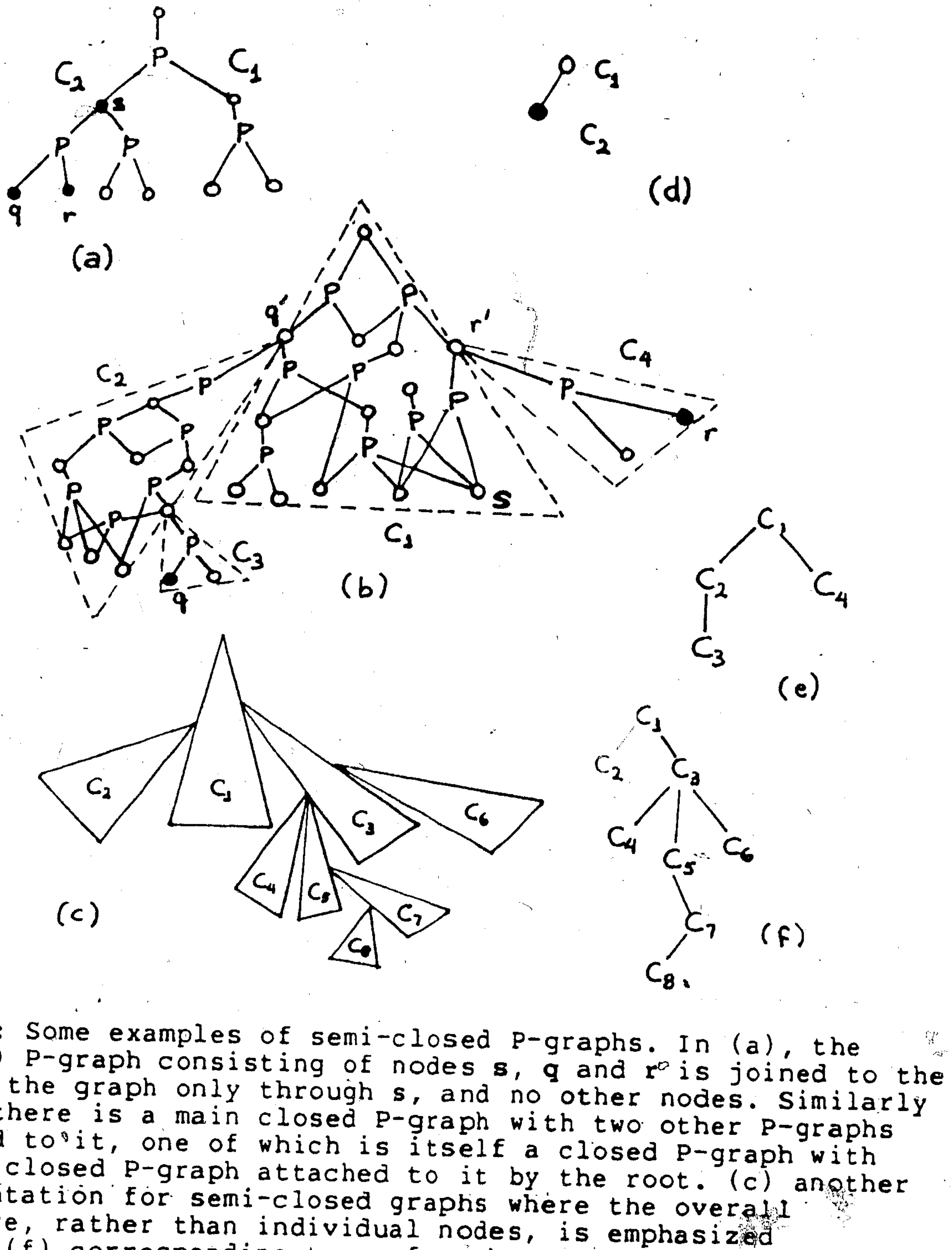

Fig. 3: Some examples of semi-closed P-graphs. In (a), the (closed) P-graph consisting of nodes **s**, **q** and **r** is joined to the rest of the graph only through s, and no other nodes. Similarly in (b) there is a main closed P-graph with two other P-graphs attached to it, one of which is itself a closed P-graph with another closed P-graph attached to it by the root. (c) another representation for semi-closed graphs where the overall structure, rather than individual nodes, is emphasized (d),(e),(f) corresponding trees for the P-graphs of (a),(b),(c).

the corresponding tree of closed P-graphs. Implementation details are of no concern at the moment; we assume semi-closed P-graphs to be searched by the algorithms presented here have been preprocessed, with labels being attached to all nodes which indicate their position in the corresponding tree of closed P-graphs. Thus, for any pair of nodes of a semi-closed P-graph, it will be possible to arrive at a pair of "ancestor" nodes which both belong to the same closed subgraph tree vertex. Note that one (or even both) of the "ancestors" sought may be the same as the corresponding initial node.

In figure 3(b), for example, for **r** and **q** the corresponding pair is **r**' and **q**', while for **r** and **s** the corresponding pair is **r**' and **s**.

The terms "ancestor" and "descendant" are in quotes above, since we are dealing with an ancestor (descendant) relation which is somewhat more general than that formally defined earlier: **r**' is an "ancestor" of **r** if and only if **r**$\leq$**r**' or **r** is projectible into a set of nodes $n_1,\ldots,n_k$ such that for all i, $1\leq i\leq k$, either $n_i\leq$**r**' or **r**' is an "ancestor" of n_i (see Fig. 4).

Algorithms for answering the questions ?[**x** part-of **y**] and ?[**x** disjoint-from **y**] in fully consistent closed P-graphs have been developed in [Schubert 1979] and

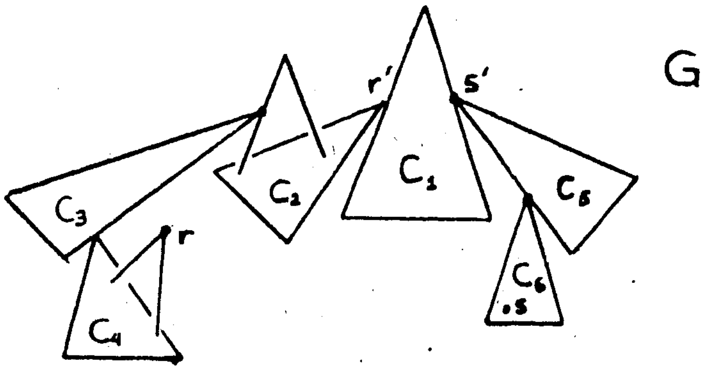

Fig. 4: The semi-closed P-graph G has closed subgraphs $C_1,\ldots,C_6$. The node r belongs to C_4 but is not a descendant of the main root of C_4. It is a "descendant" of r' as defined in this section.

are incorporated in the methods given below. So, for any two nodes **x**, **y** of a fully consistent closed P-graph G, assume there are algorithms P(**x**,**y**) and D(**x**,**y**) that will return "yes", "no" or "unknown" to the respective questions, on the basis of what can be logically deduced from the closed P-graph G. The algorithms are complete in the sense that they return "unknown" only if neither a positive nor a negative answer logically follows from the P-graph and the part-of axioms. The same property is desired for the new algorithms.

Algorithms P(**x**,**y**) and D(**x**,**y**) make use of a predicate N($\mathbf{x}_1,\ldots,\mathbf{x}n$) which is **true** if the merge of $\mathbf{x}_1,\ldots,\mathbf{x}n$ is <u>provably</u> non-empty and **false** otherwise. It was noted in [Schubert 1979] that this predicate is efficiently decidable for closed P-graphs. In applying P(**x**,**y**) and D(**x**,**y**) to closed P-graphs embedded within semi-closed P-graphs, we need to assume that N is still efficiently decidable, with the provability requirement now referring to the <u>entire</u> semi-closed graph. The assumption is justified since the only changes in the truth values of $N(x_1,\ldots,x_n)$ over nodes of a closed graph C, resulting from attachment of semi-closed P-graphs to C, are those due to the non-emptiness of nodes to which a semi-closed graph containing a provably non-empty node was attached (this information propagates "upward" in the tree of closed graphs); and the only change potentially resulting from the attachment of C <u>to</u> a semi-closed P-graph is that due to provable non-emptiness of the node to which C was attached (this information propagates "downward" via main nodes which are points of attachment in the tree of closed P-graphs). The emptiness assertions thus necessitated at points of attachment by the upward and downward flow of information can be computed in one "pass" each over all the nodes of the semi-closed P-graph, in the worst case.

In the following algorithms the test "**a≡b**" is an abbreviation which stands for:
"((C(**a**,**b**) **and** P(**a**,**b**)) **or** ?[**a** part-of **b**])"
where C(**a**,**b**) is a predicate which is **true** if the nodes a and b belong to a common closed subgraph, and **false** otherwise.

"**a≡b**" incorporates a recursive call to the algorithm ?[**a** part-of **b**] to determine whether the answer to the

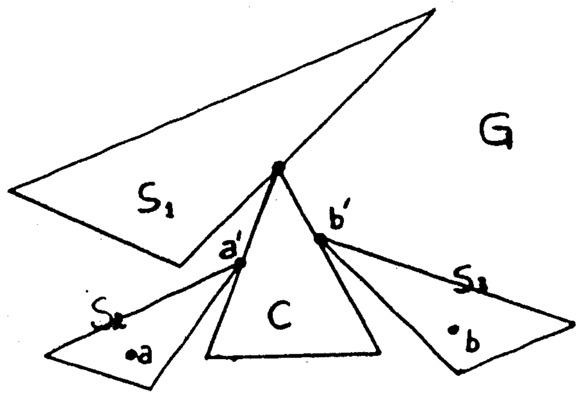

Fig. 5: G is a fully consistent semi-closed P-graph. Subgraphs S_1, S_2, S_3 are semi-closed, and C closed. **x**' and y' belong to the same closed subgraph C, so P(**x**',y') and D(**x**',y') are used. y and y' belong to a smaller semi-closed subgraph, thus the algorithms can be applied recursively to determine whether G $\vdash$ y=y'. Thus if G $\vdash$ y=y', then if P(x',y')="yes" the algorithm yields "yes" for ?[**x** part-of y]

question "a=b?" (i.e., do the nodes **a** and **b** denote the same object?) is "yes", in cases where it is already known that **b** is part of **a**; (this test is only used where **a** is an ancestor of b, as for **x**' and **x**). Let **x**', y' denote the nearest pair of "ancestors" which belong to a common closed subgraph for **x** and y respectively, as described in the above discussion.

<u>To answer ?[x part-of y]</u>:

```
if (x'≡x and y'≡y) then return P(x',y')
else if y'≡y then
      if P(x',y')="yes" then return "yes"
      else if (D(x',y')="yes" and N(x)) then return "no"
         else return "unknown"
else if x'≡x then
      if P(x',y')="no" then return "no"
      else return "unknown"
else if (D(y',x')="yes" and N(x)) then return "no"
      else return "unknown"
```

<u>To answer ?[x disjoint-from y]</u>:

```
if (x'≡x and y'≡y) then return D(x',y')
else if D(x',y')="yes" then return "yes"
else if x'≡x then
      if (P(y',x')="yes" and N(y)) then return "no"
      else return "unknown"
else if y'≡y then
      if (P(x',y')="yes" and N(x)) then return "no"
      else return "unknown"
else return "unknown"
```

The rest of this section is devoted to showing how much can be inferred from P-graphs and, in particular, proving the question answering methods given above correct and complete.

The logical foundations of these proofs have been briefly discussed in section 2. The completeness and correctness of the given algorithms relates to the derivability of statements of the form $a \subseteq b$, $a \not\subseteq b$, $\cap ab=\theta$ or $\cap ab \neq \theta$. To show that the methods are complete and correct it is necessary to prove that for an assertion ϕ, the algorithm will return "yes" if $G \vdash \phi$, "no" if $G \vdash \neg\phi$, and "unknown" otherwise.

Theorem 4.1: Let G be a fully consistent closed P-graph. Let α be an interpretation that assigns functions to the symbols $\cup$, $\cap$, and $\sim$, and some relation to the symbol $\subseteq$ of the object language and assigns objects from a set **A** to the constants (nodes) of the P-graph so as to satisfy the part-of axioms and:

(i) if **n** and **k** are leaves of G, then $\alpha(\cap nk)=\alpha(\theta)$

(ii) if **n** is a node of G marked as non-empty, then for some leaf ni of G, where $ni \leq n$, $\alpha(ni) \neq \alpha(\theta)$;

(iii) if $n_1, \ldots, nk$ is the projection of a node **n** onto the leaves of G, then $\alpha(n)=\alpha(\cup n_1, \ldots, nk)$

Then α constitutes a model of G.
(i.e., all the assertions of G are true under such an interpretation)

Proof: Let $[n\ P\ n_1, \ldots, nk]$ be an assertion of G, and let $N_1, \ldots, Nk$ be the (sets) projections of $n_1, \ldots, nk$ into the leaves of G, respectively.

From (iii)

$$\alpha(\cap ni\,nj) = \alpha(\cap(\cup Ni)(\cup Nj))$$

($\cup$ is informally used here as an operator on sets).

Since G is a closed P-graph therefore $Ni \cap Nj=\emptyset$ (for if $x \in Ni \cap Nj$ for some **x**, then $G \vdash x=\theta$, since $G \vdash \cap ni\,nj=\theta$). Thus from (i) $\alpha(\cap ni\,nj)=\alpha(\theta)$.

Now $\alpha(\cup n_1 \ldots nk) = \alpha(\cup(\cup N_1)\ldots(\cup Nk$

$$= \alpha(\cup(N_1 \cup \ldots \cup Nk))$$

Furthermore $N_1 \cup \ldots \cup N_k$ is the projection of **n** into the leaves of G, since:

1) any leaf in the projection of $\mathbf{n}_i$ is also in the projection of **n** since it is a descendant of **n**.

2) let **x** be a leaf in the projection of **n**

$$G \vdash \cap \mathbf{x}\mathbf{n} = \mathbf{x},$$

Thus $G \vdash \cap \mathbf{x}(\cup \mathbf{n}_1 \ldots \mathbf{n}_k) = \mathbf{x},$

$$G \vdash \cap \mathbf{x}(\cup(\cup N_1) \ldots (\cup N_k)) = \mathbf{x}$$

hence $G \vdash \cup(\cap \mathbf{x}(\cup N_1) \ldots \cap \mathbf{x}(\cup N_k)) = \mathbf{x}.$

But if $\mathbf{x} \neq \mathbf{y}$, for all leaves **y** in the union of the projections of the $\mathbf{n}_i$'s, then $G \vdash \cap \mathbf{x}\mathbf{y} = \theta$ so $G \vdash_T \mathbf{x} = \theta$ and G would not be fully consistent. Thus **x** must be in the projection of one of the $\mathbf{n}_i$'s.

So, by (i), $\alpha(\cup(N_1 \cup \ldots \cup N_k)) = \alpha(\mathbf{n})$.

Thus $\alpha(\cup \mathbf{n}_1 \ldots \mathbf{n}_k) = \alpha(\mathbf{n})$, and hence, $\alpha([\mathbf{n}\ \mathrm{P}\ \mathbf{n}_1 \ldots \mathbf{n}_k]) = \mathbf{true}$. Finally, from (ii) we have that $\alpha(\mathbf{n}) \neq \alpha(\theta)$ for any node **n** marked as non-empty, by theorem 3.3(b) and the part-of axioms, so that the non-emptiness assertions are also true in α. □

The following lemmas lead to the proof of correctness and completeness of the algorithms. The proofs are very similar to that of theorem 4.1.

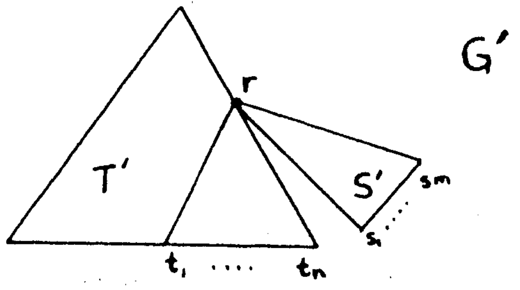

Fig. 6: Diagram for lemma 4.1.

Lemma 4.1: If G is a semi-closed P-graph, consisting of fully consistent semi-closed subgraphs S and T, such that T is attached to S by its main root, **a**, only, then G is fully consistent.

Proof: Since S and T are fully consistent it is not provable from the assertions of S and the part-of axioms that any of its nodes is empty. The same holds for T.
From theorem 3.2, there exist closed fully consistent P-graphs S' and T' logically equivalent to S and T, respectively, and thus the same also holds for S' and T'.
Let $\{t_1,...,t_n\}$ and $\{s_1,...,s_m\}$ be the projections of **a** onto the leaves of S' and T' respectively.
Construct an interpretation α such that $\cup$, $\cap$, $\sim$ and $\subseteq$ are assigned functions or relations that satisfy the part-of axioms and where

$$\alpha(\cup t_1...t_n)=\alpha(\cup s_1...s_m). \tag{1}$$

(This can be accomplished by assigning non-empty, pairwise disjoint objects to the leaves of T' and S' such that

$\alpha(t_1)=\alpha(s_1)$, ..., $\alpha(t_{p-1})=\alpha(s_{p-1})$

and

$\alpha(\cup t_p \ldots t_n)=\alpha(\cup s_p \ldots s_m)$,

where p=min{n,m}).

Also, for all leaves **l**, **k** of S' (respectively T')

$\alpha(\cap \mathbf{lk})=\alpha(\theta)$, (2)

and for any node **x** of S' (respectively T') with projection $\{\mathbf{x}_1,\ldots,\mathbf{x}_k\}$ onto the leaves,

$\alpha(\mathbf{x})=\alpha(\cup \mathbf{x}_1 \ldots \mathbf{x}_k)$. (3)

By theorem 4.1, α is a model of S' and T'; thus, since from (1) and (3) it assigns a the same interpretation in each subgraph, it satisfies all the assertions of G', so that it is also a model of G'.

Also, for any node **x** of G', $\alpha(\mathbf{x})\neq\alpha(\theta)$; thus, by soundness,

$G' \not\vdash \mathbf{x}=\theta$,

and so $G \not\vdash \mathbf{x}=\theta$, since G and G' are logically equivalent. Therefore G is fully consistent. □

For lemmas 4.2 through 4.9 let G be a semi-closed P-graph, consisting of fully consistent **closed** subgraphs S and T, such that T is attached to S by one of its main roots, **a**, only.

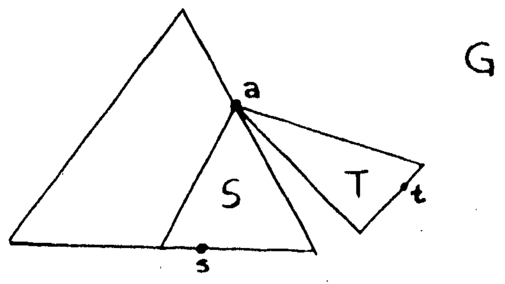

Fig. 7: Diagram for lemma 4.2.

Lemma 4.2: If s and t are leaf descendants of a in S and T respectively, then $G \not\vdash \cap st=\theta$

Proof: Since G is fully consistent, there is an interpretation α which assigns non-empty objects to all nodes of G and satisfies G.

Suppose for some s, t,

$G \vdash \cap st=\theta$.

Then by the soundness,

$$\alpha(\cap st)=\alpha(\theta). \qquad (1)$$

<u>Claim</u>: There is a leaf descendant of a in S, s', such that $\alpha(\cap s't)\neq\alpha(\theta)$

<u>Proof</u>: Suppose $\alpha(\cap s't)=\alpha(\theta)$ for every s' in the projection of a onto the leaves of S, $\{1_1,\dots,1_n\}$.

Thus $\alpha(\cap 1_i t)=\alpha(\theta)$ for all i, $1\leq i\leq n$

$\Rightarrow \alpha(\cup(\cap 1_1 t)\dots(\cap 1_n t)) = \alpha(\theta)$

$\Rightarrow \alpha(\cap(\cup 1_1,\dots 1_n)t) = \alpha(\theta)$

$\Rightarrow \alpha(\cap at) = \alpha(\theta)$

but since $t\leq a$, $G \vdash \cap ta=t$.

Therefore, $\alpha(\cap \mathbf{ta})=\alpha(\mathbf{t})$, so $\alpha(\mathbf{t})=\alpha(\theta)$
which contradicts the assumption that α assigns
non-empty objects to all the nodes.

We now construct a new interpretation β such that for any l_i such that $l_i \neq \mathbf{s}$ and $l_i \neq \mathbf{s}'$:

$$\beta(l_i)=\alpha(l_i),\ \beta(\mathbf{s})=\alpha(\mathbf{s}') \text{ and } \beta(\mathbf{s}')=\alpha(\mathbf{s})$$

(i.e., we interchange the interpretations of s and s') and for any node $\mathbf{n}$ with projection $\mathbf{n}_1,\dots \mathbf{n}_k$ onto the leaves of G:

$$\beta(\mathbf{n})=\beta(\cup \mathbf{n}_1,\dots \mathbf{n}_k),$$

where the interpretation of $\cup$, $\cap$, $\sim$ and θ is the same as in α. Note that the projection of **a** is not uniquely defined, but $\beta(\mathbf{a})=\alpha(\mathbf{a})$, since the interpretation of all the leaves of T is the same as in α and $\beta(\cup l_1,\dots l_n)=\alpha(\cup l_1,\dots l_n)$, the interpretations of $l_1,\dots l_n$ are all the same as in α except for s and s' which are interchanged. $\alpha(\cup l_1,\dots l_n)=\alpha(\mathbf{a})$, since α is a model of G, and therefore $\beta(\cup l_1,\dots l_n)=\alpha(\mathbf{a})$, so that **a** has the same interpretation in both cases.

<u>Claim</u>: β is a model of G.

<u>Proof</u>: The interpretation of all the nodes of T is the same as in α, so that β satisfies all the assertions of T.
In S, for any two leaves **x**, **y** except **s**, **s**',

$$\beta(\mathbf{x})=\alpha(\mathbf{x}) \text{ and } \beta(\mathbf{y})=\alpha(\mathbf{y}), \text{ so } \beta(\cap \mathbf{xy})=\beta(\theta),$$

since by theorem 3.3 all the leaves of S are pairwise disjoint, because S is closed and α satisfies G.
Also

$$\beta(\cap \mathbf{s}'\mathbf{y}) = \alpha(\cap \mathbf{sy}) = \alpha(\theta)$$

and

$$\beta(\cap sy) = \alpha(\cap s'y) = \alpha(\theta)$$

and

$$\beta(\cap s's) = \alpha(\cap s's) = \alpha(\theta).$$

Thus, by theorem 4.1 β satisfies G.

Now

$$\beta(\cap st) = \alpha(\cap s't) \neq \alpha(\theta),$$

so $\beta(\cap st) \neq \beta(\theta)$.

Thus G $|\neq$ $\cap st=\theta$, which is a contradiction to our assumption.

Hence G $\not\vdash$ $\cap st=\theta$. □

Lemma 4.3: Let **x** and y be nodes of S and T respectively, such that **x** is projectible into a subhierarchy of S rooted at a.

(a) G $\vdash$ y$\subseteq$**x** => G $\vdash$ **a=x**

(b) G $\vdash$ x$\subseteq$y => G $\vdash$ **a=y**

Proof: (a) Consider all the leaf nodes of S and T. Let $\{x_1,\ldots,x_n\}$ and $\{y_1,\ldots,y_m\}$ be the projections of **x** and **y** onto the leaves of S and T, respectively.

Thus,

G $\vdash$ $x=\cup x_1\ldots x_n$ and G $\vdash$ $y=\cup y_1\ldots y_m$.

Now assume there is a leaf **1** in the projection of **a** that is not in that of **x**.

By theorem 3.3(c),

G $\vdash$ $\cap 1x=\theta$. (1)

Suppose y$\subseteq$**x**; then for all i, $1\leq i\leq m$

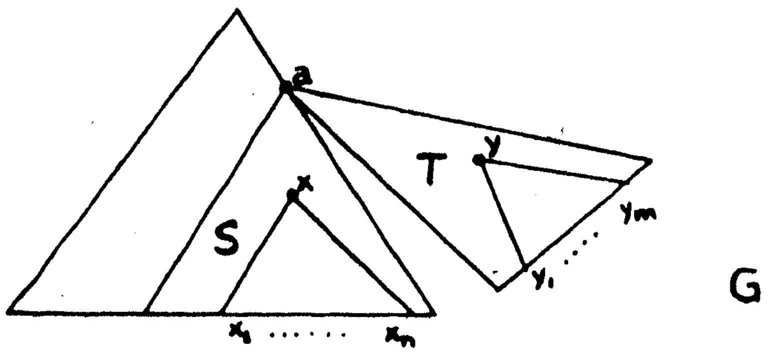

Fig. 8: Diagram for lemma 4.3.

$y \subseteq x$

$\Rightarrow \cup y_i x = x$

$\Rightarrow \cap 1(\cup y_i x) = \theta$

$\Rightarrow \cup(\cap 1 y_i)(\cap 1 x) = \theta$

$\Rightarrow \cup(\cap 1 y_i)\theta = \theta$

$\Rightarrow \cap 1 y_i = \theta$

So, from (1), G $\vdash \cap 1 y_i = \theta$ for all i, $1 \le i \le m$, which, by lemma 4.2 is a contradiction. Thus $\{x_1, \ldots, x_n\}$ is also the projection of a onto the leaves of S. Hence G $\vdash a = \cup x_1 \ldots x_n$, i.e., G $\vdash x = a$.

(b)The proof is the same as for (a). □

Lemma 4.4: Let x and y be nodes of S and T respectively. Then

$$G \vdash y \subseteq x \iff G \vdash a \subseteq x.$$

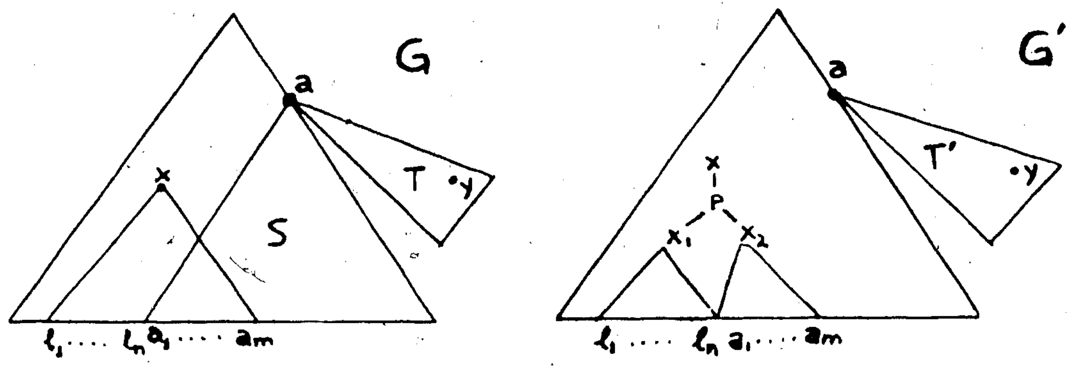
G
a
T
•y
x
S
$l_1 \ldots l_n a_1 \ldots a_m$
G'
a
T'
•y
x
P
x_1
x_2
$l_1 \ldots l_n a_1 \ldots a_m$

Fig. 9: Diagram for lemma 4.4.

Proof: Since G $\vdash$ $y \leq a$, it follows that

$$G \vdash a \leq x \Rightarrow G \vdash y \leq x.$$

Let $\{l_1, \ldots, l_n, a_1, \ldots, a_m\}$ be the projection of **x** onto the leaves of S, where $a_1, \ldots, a_m$ are also in the projection of a; Let the (object language) assertions

$$x_1 = \cup l_1 \ldots l_n$$

and $$x_2 = \cup a_1 \ldots a_m$$

be added to the assertions of G to give a set of assertions G', where x_1, x_2 are constants distinct from each other and from all the nodes of G. Then

$$G' \vdash [x, P\ l_1 \ldots l_n],$$

$$G' \vdash [x_2\ P\ a_1 \ldots a_m]$$

(since the leaves of S are pairwise disjoint)

and

$$G' \mid\ [x\ P\ x_2 x_1].$$

Thus the ve assertions, when added to G' to form G", do not provide any new information, i.e., for any assertion A

$$G'' \vdash A \Rightarrow G' \vdash A.$$

The converse is also true, since G" contains all of the assertions of G', so that

$$G'' \vdash A \iff G' \vdash A. \quad (1)$$

Now,

$$G'' \vdash x = \cup x_1 x_2 \quad (2)$$

and

$$G'' \vdash \cap(\cup l_1 \ldots l_n)(\cup a_1 \ldots a_m \ldots a_r) = \theta,$$

where $\{a_1, \ldots, a_r\}$ is the projection of **a** into the leaves of S.

Therefore $G'' \vdash \cap x_1 a = \theta$. (3)

Thus, since $G'' \vdash y \subseteq a$,

$$G' \vdash \cap x_1 y = \theta. \quad (4)$$

Now suppose $G \vdash y \subseteq x$.

$$y \subseteq x$$

$$\Rightarrow \cap yx = y$$

$$\Rightarrow \cap y(\cup x_1 x_2) = y$$

$$\Rightarrow \cup(\cap y x_1)(\cap y x_2) = y.$$

Thus $G'' \vdash \cup(\cap y x_1)(\cap y x_2) = y$ and hence, from (4),

$G'' \vdash \cup\theta(\cap y x_2) = y$, so that $G'' \vdash y \subseteq x_2$.

Using lemma 4.3 , since x_2 is projectible into a subhierarchy of S rooted at a,

$$G'' \vdash x_2 = a.$$

Hence from (2)

$$G'' \vdash a \subseteq x.$$

Therefore

$$G'' \vdash y \subseteq x_2 \Rightarrow G'' \vdash a \subseteq x$$

and from (1)

$$G \vdash a \subseteq x, \text{ or equivalently, } G' \models a \subseteq x.$$

But since any model α satisfying G (and the part-of axioms) can be modified so as to satisfy G' without affecting the truth value of $a \subseteq x$ (namely, by setting the denotations of x_1 and x_2 to $\alpha(\cup l_1, \ldots l_n)$ and $\alpha(\cup a_1, \ldots a_m)$ respectively), we have

$$G \models a \subseteq x,$$

and hence

$$G \vdash a \subseteq x. \ \square$$

Lemma 4.5: Let x and y be nodes of S and T respectively. If $G \nvdash a \subseteq y$ then $G \nvdash x \subseteq y$

Proof: If $G \vdash x \subseteq y$, then $G \vdash x \subseteq a$ (because $G \vdash y \subseteq a$), so x is projectible into a subhierarchy of S rooted at a. Thus by lemma 4.3 $G \vdash a = y$, so that $G \vdash a \subseteq y$. $\square$

Lemma 4.6: Let x and y be nodes of S and T respectively. If $G \nvdash a \subseteq y$ then

$$G \vdash y \not\subseteq x \text{ iff } G \vdash \cap xa = \theta \text{ and } G \vdash y \neq \theta.$$

Proof: Clearly if $G \vdash \cap ax = \theta$ and $G \vdash y \neq \theta$ then $G \vdash y \not\subseteq x$, since $G \vdash y \subseteq a$.

Also $G \vdash y \not\subseteq x \Rightarrow G \vdash y \neq \theta$

The proof for $G \vdash y \not\subseteq x \Rightarrow G \vdash \cap ax = \theta$ is similar to the proof of lemma 4.2. Suppose $G \nvdash \cap ax = \theta$. Then the projections of a and x onto the leaves of S must have nodes in common.

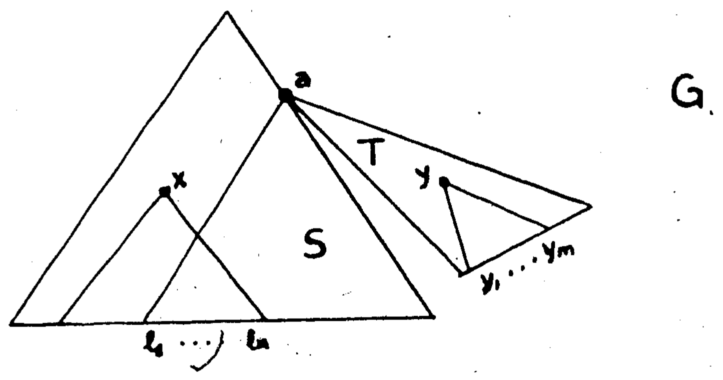

Fig. 10: Diagram for lemma 4.6.

Let $l_1, \dots, l_n$ denote the common leaf nodes. A model α of G can be constructed using a method similar to the one in the proof of lemma 4.2 so that $l_1, \dots, l_n$ and the projection of y onto the leaves of T are interpreted in the following way.

<u>Case 1:</u> If $l_1, \dots, l_n$ is the entire projection of a (i.e., G $\vdash a \subseteq x$), then

$\alpha(\cup l_1 \dots l_n) = \alpha(\cup t_1 \dots t_k) = \alpha(a)$,

where $t_1, \dots, t_k$ are all the leaves of T.

<u>Case 2:</u> If some leaf of S is a descendant of a but not of **x**, then

$\alpha(\cup l_1 \dots l_n) = \alpha(\cup y_1 \dots y_m) = \alpha(y)$

(Note that there is a potential inconsistency in the interpretations of a which are "propageted up" from the leaves of S and T, in the case where $y_1, \dots, y_m$ is the set of all leaves of T and there is a leaf l of S, with $l \notin \{l_1, \dots, l_n\}$ and l is marked non-empty; but this inconsistency is avoided by the premise G $\not\vdash a \subseteq y$, which guarantees that there are leaves of T other than $y_1, \dots, y_m$). But then since

$G \vdash ul_1,\ldots l_n \subseteq x$

by soundness, $\alpha(ul_1,\ldots l_n \subseteq x)$=**true**, so that $\alpha(y \subseteq x)$=**true**. Hence $G \not\vdash y \not\subseteq x$. □

Lemma 4.7: Let **x** and **y** be nodes of S and T respectively. If $G \not\vdash a \subseteq x$ then

$G \vdash x \not\subseteq y \Leftrightarrow G \vdash x \not\subseteq a$.

Proof: $G \vdash x \subseteq y \Rightarrow x \subseteq a$ so that $G \vdash x \not\subseteq a \Rightarrow x \not\subseteq y$; therefore

$G \vdash x \not\subseteq a \Rightarrow G \vdash x \subseteq y$.

(Forward direction). Let $\{x_1,\ldots,x_r,l_1,\ldots,l_n\}$ and $\{l_1,\ldots,l_n,k_1,\ldots,k_s\}$ be the projections of **x** and **a** respectively onto the leaves of S, and let $\{y_1,\ldots,y_m\}$ be the projection of **y** onto the leaves of T.

Now suppose that $G \not\vdash a \subseteq x$ and $G \vdash x \not\subseteq y$, but $G \not\vdash x \not\subseteq a$. Then $\{k_1,\ldots,k_s\} \neq \emptyset$, for otherwise we would have $G \vdash x \not\subseteq a$. Then $\{k_1,\ldots,k_s\} \neq \emptyset$, for otherwise we would have $G \vdash a \subseteq x$; and no node projectible onto $\{x_1,\ldots,x_r\}$ is marked nonempty, for otherwise we would have $G \vdash x \not\subseteq a$

Then we are able to construct the following sort of model α of G. We assign $\alpha(x_i)=\alpha(\theta)$ for $1 \leq i \leq r$; and we assign non-empty pairwise disjoint objects to the remaining leaves of S, and non-empty pairwise disjoint objects to the leave of T, with the following additional constraints.

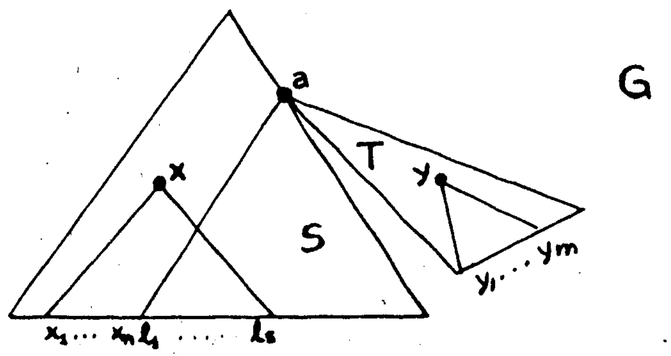

Fig. 11: Diagram for lemma 4.7.

Case 1: If $\{z_1,\dots,z_t\}=\emptyset$, then we let

$\alpha(\cup l_1,\dots l_n k_1,\dots k_s) = \alpha(\cup y_1,\dots y_m)$ (much as in lemma 4.1).

Case 2: If $\{z_1,\dots,z_t\}\neq\emptyset$, then we let $\alpha(\cup l_1,\dots l_n) =$

$\alpha(\cup y_1,\dots y_m)$, and

$\alpha(\cup k_1,\dots k_s) = \alpha(\cup z_1,\dots z_t)$.

Then we set

$\alpha(n) = \alpha(\cup n_1,\dots n_u)$

for any non-leaf node of either closed graph with projection $\{n_1,\dots,n_u\}$ onto the leaves of that graph.

Now we observe that

(i) α satisfies S and T separately, by theorem 4.1. (Note that only nodes projectible onto $\{x_1,\dots,x_r\}$ receive empty interpretations, and no such nodes are marked non-empty, so that α satisfies all non-emptiness assertions.)

(ii) In both cases above,

$\alpha(\cup l_1,\dots l_n k_1,\dots k_s) = \alpha(\cup y_1,\dots y_m z_1,\dots z_t)$,

so that the two assignments "propaga[illegible] up to a are identical, i.e., α simultaneously satisfies S and T.

(iii) In both cases,

$\alpha(\mathbf{x})=\alpha(\cup 1_1,\ldots 1_n)$ and

$\alpha([\cup \mathbf{1}_1,\ldots \mathbf{1}_n \subseteq \mathbf{y}])=$ **true**,

so that $\alpha([\mathbf{x}\subseteq\mathbf{y}])=$ **true**.

Hence $G \not\vdash \mathbf{x}\subseteq\mathbf{y}$, in contradiction with one of the assumptions. □

Lemma 4.8: Let **x** and **y** be nodes of S and T respectively. Then

$$G \vdash \cap\mathbf{xy}=\theta \Leftrightarrow G \vdash \cap\mathbf{ax}=\theta.$$

Proof: Clearly, $G \vdash \cap\mathbf{ax}=\theta \Rightarrow G \vdash \cap\mathbf{xy}=\theta$. Let $\{\mathbf{a}_1,\ldots,\mathbf{a}_k\}$, $\{\mathbf{x}_1,\ldots,\mathbf{x}_n\}$ and $\{\mathbf{y}_1,\ldots,\mathbf{y}_n\}$ be the projections of **a**, **x** and **y** onto the leaves of S and T respectively. Now $G \vdash \cap\mathbf{xy}=\theta \Rightarrow G \vdash \cap\mathbf{x}_i\mathbf{y}_j=\theta$, for all i, j, $1\le i\le n$, $1\le j\le m$. Thus, by lemma 4.2, the projections of **a** and **x** have no leaves in common. But then

$G \vdash \cap\mathbf{x}_i\mathbf{a}_j=\theta$ for all i, j, $1\le i\le n$, $1\le j\le k$

$\Rightarrow G \vdash \cup(\cap\mathbf{x}_1\mathbf{a}_1)\ldots(\cap\mathbf{x}_1\mathbf{a}_k)=\theta$,

$\vdots$

$G \vdash \cup(\cap\mathbf{x}_n\mathbf{a}_1)\ldots(\cap\mathbf{x}_n\mathbf{a}_k)=\theta$

$\Rightarrow G \vdash \cap\mathbf{x}_1(\cup\mathbf{a}_1,\ldots\mathbf{a}_k)=\theta$, ..., $G \vdash \cap\mathbf{x}_n(\cup\mathbf{a}_1,\ldots\mathbf{a}_k)=\theta$

$\Rightarrow G \vdash \cap\mathbf{x}_1\mathbf{a}=\theta$, ..., $G \vdash \cap\mathbf{x}_n\mathbf{a}=\theta$

$\Rightarrow G \vdash \cap\mathbf{xa}=\theta$ □

Lemma 4.9: Let **x** and **y** be nodes of S and T respectively. If $G \nvdash \mathbf{a} \subseteq \mathbf{y}$ then

$$G \vdash \cap\mathbf{xy} \neq \theta \iff G \vdash \mathbf{a} \subseteq \mathbf{x} \text{ and } G \vdash \mathbf{y} \neq \theta.$$

Proof: If $G \vdash \mathbf{a} \subseteq \mathbf{x}$ then $G \vdash \mathbf{y} \subseteq \mathbf{x}$ so that $G \vdash \cap\mathbf{xy}=\mathbf{y}$, therefore

$$(G \vdash \mathbf{a} \subseteq \mathbf{x} \text{ and } G \vdash \mathbf{y} \neq \theta) \Rightarrow G \vdash \cap\mathbf{xy} \neq \theta.$$

Now suppose $G \vdash \cap\mathbf{xy} \neq \theta$. Then $G \vdash \mathbf{y} \neq \theta$ and

$$\begin{aligned} &G \vdash \mathbf{y} \subseteq \mathbf{a} \\ &\quad \Rightarrow G \vdash \cap\mathbf{xa} = \cap\mathbf{x}(\cup\mathbf{ya}) \\ &\quad \Rightarrow G \vdash \cap\mathbf{xa} = \cup(\cap\mathbf{xy})(\cap\mathbf{xa}) \\ &\quad \Rightarrow G \vdash \cap\mathbf{xa} \neq \theta. \end{aligned}$$

Since **x**, **a** are in the same closed P-graph, there are common leaves $l_1,\ldots,l_k$ in the projections of **x** and **a** onto the leaves S. The number of such leaves cannot be zero, for in that case $G \vdash \cap\mathbf{xa}=\theta$ would hold.

Suppose there is a leaf $\mathbf{a}_1$ in the projection of **a** but not that of **x**. Then a model α can be constructed as follows: Let the projection $\{y_1,\ldots,y_m\}$ of **y** onto the leaves of T, be assigned pairwise disjoint objects and let $\mathbf{a}_1$ be assigned $\alpha(\mathbf{a}_1)$ such that

$$\alpha(\mathbf{a}_1) = \alpha(\cup y_1 \ldots y_m).$$

Let the rest of the leaves of T be assigned pairwise disjoint objects (here we used $G \vdash \mathbf{a} \subseteq \mathbf{y}$), disjoint from $y_1,\ldots,y_m$, and let the rest of the leaves in the projection of **a** onto the leaves of S be assigned pairwise disjoint with the same merge as those assigned to the rest of the leaves

of T. Let the leaves of S be assigned other pairwise disjoint objects, disjoint from those.
Since all the leaves are interpreted by disjoint objects, the rest of the nodes of G can be interpreted so as to form a model for G (theorem 4.1). But $\alpha(\mathbf{y})=\alpha(\mathbf{a}_1)$, so, since for all i, $1\le i\le k$, $\alpha(\cap \mathbf{a}_1 \mathbf{l}_i)=\alpha(\theta)$, $\alpha(\cap \mathbf{yx})=\alpha(\theta)$.
Hence G $\not\vdash$ $\cap\mathbf{yx}\neq\theta$.
This is a contradiction, so there is no leaf $\mathbf{a}_1$ in the projection of **a** that is not in the projection of **x**.
Thus G $\vdash$ $\mathbf{a}\subseteq\mathbf{x}$, so G $\vdash$ $\cap\mathbf{yx}\neq\theta$ => G $\vdash$ $\mathbf{a}\subseteq\mathbf{x}$. □

The next theorem relaxes the restriction that the subgraphs T and S be closed, in lemmas 4.4 through 4.9.

Theorem 4.2: Let G be a semi-closed P-graph, consisting of semi-closed subgraphs S and T, such that T is attached to S by its main root, **a**, only. Let **x** and **y** be nodes of S and T respectively. If G $\not\vdash$ $\mathbf{a}\subseteq\mathbf{y}$ then:

(a) G $\vdash$ $\mathbf{y}\subseteq\mathbf{x}$ iff G $\vdash$ $\mathbf{a}\subseteq\mathbf{x}$

(b) G $\not\vdash$ $\mathbf{x}\subseteq\mathbf{y}$

(c) G $\vdash$ $\mathbf{y}\not\subseteq\mathbf{x}$ iff G $\vdash$ $\cap\mathbf{xa}=\theta$ and G $\vdash$ $\mathbf{y}\neq\theta$

(d) If G $\not\vdash$ $\mathbf{a}\subseteq\mathbf{x}$ then G $\vdash$ $\mathbf{x}\not\subseteq\mathbf{y}$ iff G $\vdash$ $\mathbf{x}\not\subseteq\mathbf{a}$

(e) G $\vdash$ $\cap\mathbf{xy}=\theta$ iff G $\vdash$ $\cap\mathbf{ax}=\theta$

(f) G $\vdash$ $\cap\mathbf{xy}\neq\theta$ iff G $\vdash$ $\mathbf{a}\subseteq\mathbf{x}$ and G $\vdash$ $\mathbf{y}\neq\theta$

Proof: In [Schubert 1980] an algorithm is given for generating a closed, logically equivalent P-graph (with

respect to assertions involving constants of the original P-graph) from an arbitrary P-graph. Thus there is a logically equivalent semi-closed P-graph for G, G', where the semi-closed subgraph T has been replaced by a closed subgraph T' and the rest of G without T has been replaced by a closed subgraph S' (theorem 3.2).
Now, for any assertion A involving constants of G only,

$$G \vdash A \iff G' \vdash A,$$

and using lemma 4.4 through 4.9 the theorem follows. □

For the following corollaries to the theorem let G be a fully consistent semi-closed P-graph with subgraphs T_1, S, T_2 such that T_1 and T_2 are rooted at nodes **a'**, **b'** of S, respectively and have no other common nodes with S or each other. Let **a**, **b** be nodes of T_1, T_2 respectively, such that

$$G \nvdash \mathbf{a}' \subseteq \mathbf{a} \text{ and } G \nvdash \mathbf{b}' \subseteq \mathbf{b}.$$

Corollary: $G \nvdash \mathbf{a} \subseteq \mathbf{b}$.
(by part (b))

Corollary: $G \vdash \mathbf{a} \nsubseteq \mathbf{b}$ iff $G \vdash \cap \mathbf{a}'\mathbf{b}'=\theta$ and $G \vdash \mathbf{a}\neq\theta$.
(by parts (c) and (d))

Corollary: $G \vdash \cap \mathbf{ab}=\theta$ iff $G \vdash \cap \mathbf{a}'\mathbf{b}'=\theta$.
(by part (e))

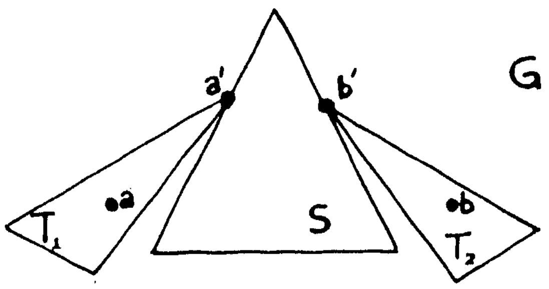

Fig. 12: Diagram for the corollaries to theorem 4.2.

Corollary: G $\not\vdash$ $\cap$**ab**$\neq\theta$.
(since by part (b), G $\not\vdash$ **b**$\subseteq$**a**, hence G $\not\vdash$ **b'**$\subseteq$**a**, hence by (f), G $\not\vdash$ $\cap$**ab**$\neq\theta$).

The correctness and completeness of the proposed algorithms follows from the above theorem and its corollaries. In both algorithms line 1 eliminates the case where **x** and **y** belong to a common closed subgraph. In the algorithm to answer ?[**x** part-of **y**], lines 2 through 5 correspond to the case where **y** is an "ancestor" of **x** and make use of theorem 4.2 (a) and (c). Lines 6 through 8 correspond to the case where **x** is an "ancestor" of **y** and make use of (d) and (b). When neither is the case (line 9), the first and second corollaries provide an answer.

Similarly, in the algorithm to answer disjointness questions, line 2 uses the third corollary, lines 3 through 8 theorem 4.2 (f) and the rest of the cases are taken care of by the fourth corollary.

4. Inference about Colours

Consider a system with a resolution-based deductive component. The following are some trivial problems it may be faced with, either in answering user questions or in checking new information for inconsistency and redundancy:

1. <u>Given</u> <u>knowledge:</u> elephant(Clyde),
   ¬elephant(x)∨grey(x)

   <u>Question:</u> ?grey(Clyde)

2. <u>Given</u> <u>knowledge:</u> elephant(Clyde),
   plus knowledge about types of animals

   <u>Question:</u> ?animal(Clyde)

3. <u>Given</u> <u>knowledge:</u> elephant(Clyde),
   plus knowledge about types of animals

   <u>Question:</u> ?canary(Clyde)

4. <u>Given</u> <u>knowledge:</u> yellow(Clyde),
   ¬elephant(x)∨grey(x),
   plus knowledge about colours

   <u>Question:</u> ?elephant(Clyde)

Question 1 can be answered by resolving the two complementary "elephant" literals, with result grey(Clyde). In a refutation proof, this would in turn be resolved against the denial ¬grey(Clyde) of the question. The resultant empty clause justifies a "yes" answer.

Question 2 could be answered by a series of resolution steps that progress along the superconcept sequence connecting "elephant" and "animal"; but this is just where we would like instead to invoke special inference methods for type lattices, as described in the previous chapter. From the theorem prover's point of view, this should be a one-step inference: in terms of a refutation proof, elephant(Clyde) is incompatible with the denial ¬animal(Clyde) of the question in much the same way that complementary literals are incompatible, and should yield the null "resolvent". Similarly, it should be possible to obtain a one step disproof for question 3 by "resolving" the incompatible literals elephant(Clyde) and canary(Clyde). In question 4, one "resolving" step should recognize the incompatibility of yellow(Clyde) and grey(x) and hence infer the "resolvent" ¬elephant(Clyde), which then resolves in the proper sense with the question clause, to yield a negative answer.

One interesting question which arises about this sort of "resolving" is whether it can be extended to deal with <u>modified</u> <u>predicates</u> such as "large animal", "dark brown" and "sort of brown" (or brown<u>ish</u>). Natural language, after all, provides a large repertoire of predicate modifiers, and presumably any adequate knowledge representation language must contain the logical counterparts of at least some of these.

4.1 Predicates modified by hedges

A particularly troublesome is "sort of"; it is characteristic of this modifier (and of "hedges" in general) that (sort-of P)(x) fails to entail P(x); this is in contrast with cases like (large P)(x), (typical P)(x), and (dark P)(x). Note however that we can take P(x) to entail (sort-of P)(x). For example, an elephant is certainly sort of an elephant, although the maxims of cooperative conversation (specifically the quantity and brevity maxims) imply that use of the hedge is improper and therefore misleading if the unhedged predicate is known to apply [Grice 1975].

Let us reconsider the (extended) resolution steps postulated in examples (1) - (4) with some of the predicates modified by sort-of. In example (1) the standard resolution {elephant(Clyde), ¬elephant(x)} was required. According to the assumed properties of sort-of, the pair {(sort-of elephant)(Clyde), ¬elephant(x)} is compatible while the pair {elephant(Clyde), ¬(sort-of elephant)(x)} is not. The latter incompatibility is easily detected in two resolution steps given the axiom schema

$$\neg P(x) \vee (\text{sort-of } P)(x)$$

which captures the entailment postulated above. No methods other than standard resolution (in conjunction with rules for applying schemata involving predicate modifiers) appear to be required in this case.

Example (2) called for "resolving" the pair {elephant(Clyde), ¬animal(Clyde)} by special lattice methods, so as to avoid the need for constructing long resolution chains. Now {(sort-of elephant)(Clyde), ¬animal(Clyde)} are compatible, but {elephant(Clyde), ¬(sort-of animal)(Clyde)} plainly are not. In fact the stronger statement can be made that {(sort-of elephant)(Clyde), ¬(sort-of animal)(Clyde)} are incompatible. (The statement is stronger because by the axiom schema for sort-of, it entails the incompatibility of {elephant(Clyde), ¬(sort-of animal)(Clyde)}.) This incompatibility can again be efficiently detected with essentially the same lattice algorithms as were needed for the unhedged case, along with the general rule that if P is superordinate to Q in a type lattice, then (sort-of Q) is incompatible with ¬(sort-of P) (i.e., entails (sort-of P)).

For example (3), we observe that {(sort-of elephant)(Clyde), canary(Clyde)} and {elephant(Clyde), (sort-of canary)(Clyde)} are incompatible pairs. Given efficient methods for detecting incompatibility of type predicates P, Q, we can easily detect these new incompatibilities as well, using the rule that (sort-of P), Q are incompatible whenever P, Q are. Note, however that the stronger incompatibility observed in example (2) now fails: {(sort-of P)(x), (sort-of Q)(x)} are compatible even when P, Q are not; a thing can <u>conceivably</u> be both a sort of an elephant and a sort of canary (consider myths and fairy tales), though the actual existence of such a thing (even allowing prodigious advances in genetic engineering) may be wildly implausible.

The examples so far can inspire the hope that the data structures and algorithms developed for efficient detection of superordination and incompatibility relationships in predicate taxonomies are sufficient as well for detecting incompatibilities of hedged predicates. This hope begins to falter, however, as we proceed to example (4). Curiously colour predicates, which we might imagine to be particularly "primitive" and more simply structured than nominal predicates, appear to obey more complex laws. Not only is it correct to say that {(sort-of tan)(Clyde), ¬(sort-of brown)(Clyde)} for example, are incompatible, as in the strong analog of example (2), but the strong analog of 4 now holds as well in certain cases; for example, if Clyde is sort of yellow (or yellowish) he cannot be sort of blue (or blueish). If this incompatibility held in all cases, there would still be no need for specialized representations of relationships among colour terms apart from those which can be captured in a simple specialization lattice. However, this is not the case; while a colour cannot be both sort of yellow and sort of blue, it can be both sort of yellow and sort of green, for example; in some sense, this is because yellow and green are more nearly compatible than yellow and blue.

One possible solution is to augment the basic hierarchy with special "sort-of" links. Ordinary links in a specialization (IS-A) hierarchy indicate subordination, and the direct descendants of a node are implicitly taken to be incompatible. The "sort-of" links would exhaustively specify for each colour whether it can be "sort-of" another colour.

One disadvantage would be the loss of the tree structure; for example, there would be two "sort-of" links connecting "chartreuse" to yellow and green respectively, which are on separate branches of the basic hierarchy. Moreover, there would be very many such links. The most serious problem, however, is that compatibilities and incompatibilities of new colour terms could not be predicted. For example, the mere absence of any colour term

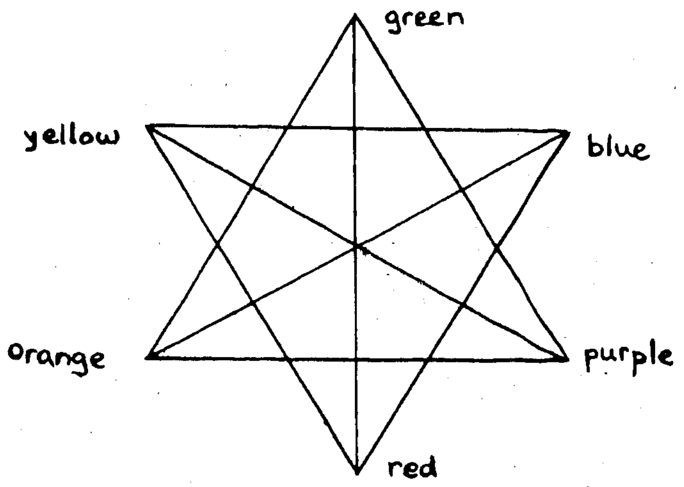

Fig. 13: Strong incompatibility links between major colour terms. (The remaining 5 terms black, brown, grey, white and pink can be added as well).

with "sort-of" links to both "blue" and "yellow" does not rule out the possibility that there could be a colour term with both links.

One approach considered introduces "strong incompatibility" links instead of sort-of links (fig. 13). Two colours P, Q are taken to be strongly incompatible just in case another colour cannot be both "sort-of P" and "sort-of Q". The resultant graph is rather pleasing and solves the problem of predicting incompatibility of hedged colour terms.

However, each new type of link introduced into a graphical representation of colours seems to capture only one type of relationship among colour terms. For example, the strong incompatibilities appear to provide no explanation of the intuition that if a colour is "sort-of scarlet" then it is not just "sort-of red", but simply red, whereas the analogous inference fails for "sort-of magenta". (Though magenta is a shade of red, a "sort-of magenta" may be too far towards the purple to be properly called red).

4.2 The cube model

Such subtleties have led to a third kind of representation, more specialized still than hierarchies, which takes <u>quantitative</u> account of the (perceived) composition of colours. Those with experience in painting or computer graphics may be familiar with three dimensional models of colour space. One of the simplest models for representing colours as mixtures of primitives is the colour cube (fig. 14). Starting in one corner (white) each of the edges coming out of that corner is increasing in the intensity of a primary. Any plane passing through the colour cube, parallel to one of the faces will be constant in one of the primaries and will contain all proportions of the remaining two. Thus the corners of the cube are: red, orange, black, purple, blue, green, yellow and white.

It is interesting to note that the star shaped graph of fig. 13, formed by the strong incompatibility relations among the major colours, can be embedded in the cube

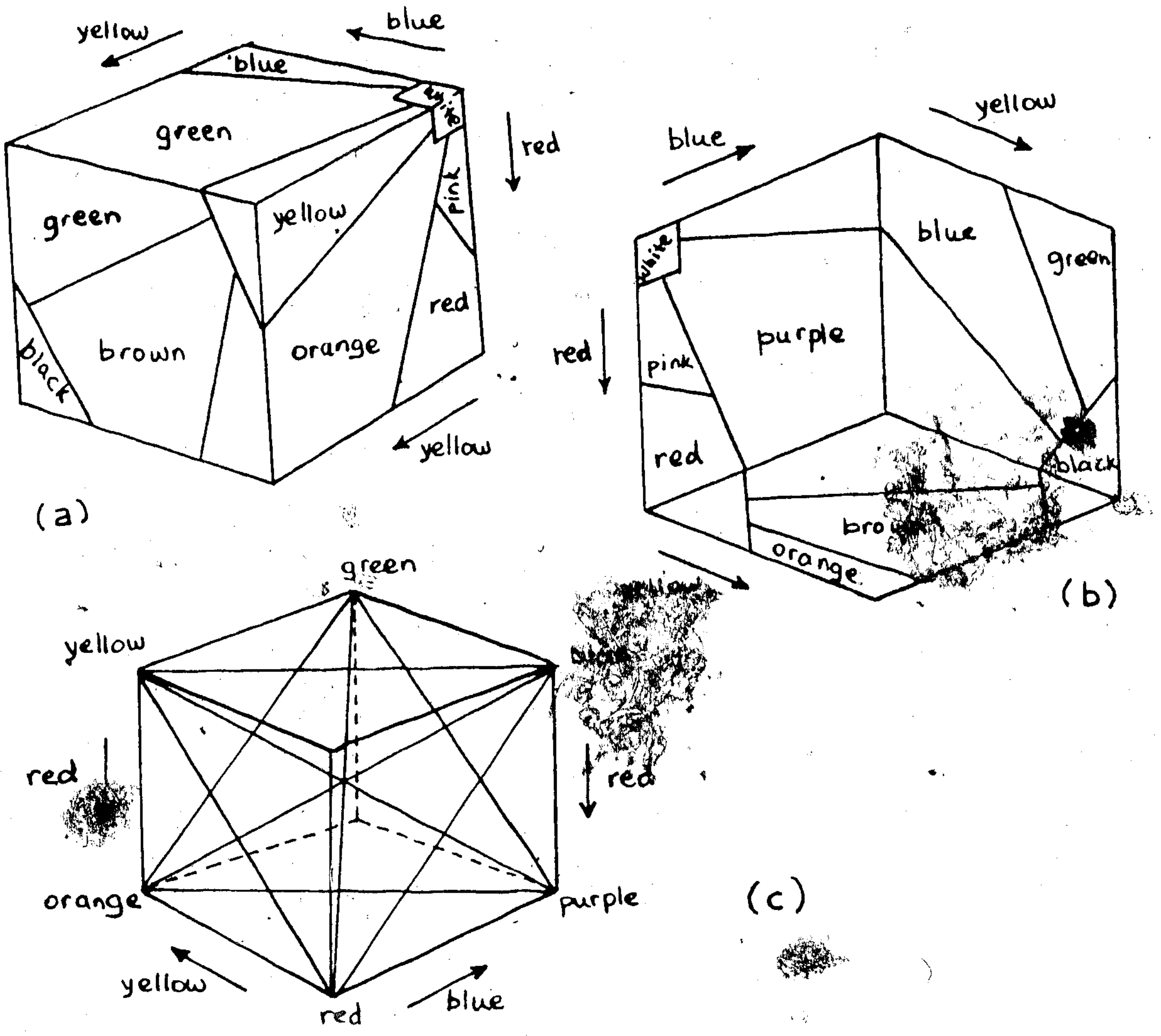

Fig. 14: (a),(b) Two views of the colour cube,
(c) Strong incompatibility links seen as diagonals of the colour cube.

(fig. 14 (c)). Also, although there is a smooth transition between any two colours, a given colour term corresponds to a certain sub-volume inside the cube. It becomes apparent that a criterion for strong incompatibility is that the defining volumes for two predicates are not adjacent.

Similarly, two colours are incompatible if the defining volumes are non-overlapping.

As one can appreciate from inspection of fig. 14, defining the subregions of the colour cube corresponding to the natural colours is not a trivial task. (Although the exact parameters are not a major concern, it is important do justice to the shapes of the regions and their interrelationships in order to achieve our inference objectives.) Instead of working directly with the primary values, it is helpful to re-parameterize colours in terms of

$$\text{purity} = \frac{\text{pure colour component}}{\text{pure colour component} + \text{black component}}$$

$$= \frac{\text{pure colour component}}{1 - \text{white component}}$$

and

$$\text{dilution} = \text{white component}$$

where

$$\begin{aligned}\text{pure colour} &= 1 - \text{white} - \text{black},\\ &= \max(\text{red,blue,yellow}) - \min(\text{red,blue,yellow}),\end{aligned}$$

$$\text{white} = 1 - \max(\text{red,blue,yellow}),$$

$$\text{black} = \min(\text{red,blue,yellow}),$$

and all quantities are assumed to range from 0 to 1.

A third quantity is needed to define the hue of the pure colour component.

The cube model, however, despite its initial appeal, and although it is as adequate as any model for describing colours as additive or subtractive mixtures of three primaries, fails to include all distinguishable colours. Surprising as this may seem, it is due to the fact that any three primaries can, at best, produce only part of the whole spectrum of hues [Judd 1963]. Of course, any graphics tool that we can use to experiment with the model will employ three primaries, but nonetheless a model that <u>in theory</u> can depict all distinguishable hues is preferable. Moreover, we would like regions defining the natural colour terms to be simpler than those in the cube model.

4.3 The cylinder model

The cylinder model is similar in many respects to other well known models ([Munsell [illegible]], [Ostwald 1969]) in that the hues are arranged in a circle around some axis and hue is specified by angular displacement. The essential difference from the cube model is that the chromatic (rainbow) hues are no longer viewed as composed of primaries, but simply as particular angular positions relative to a reference direction (say, scarlet). Purity and dilution, as defined earlier, are the other two dimensions. However, the "pure colour component" is no longer reducible to the difference between the max and min of the primaries,

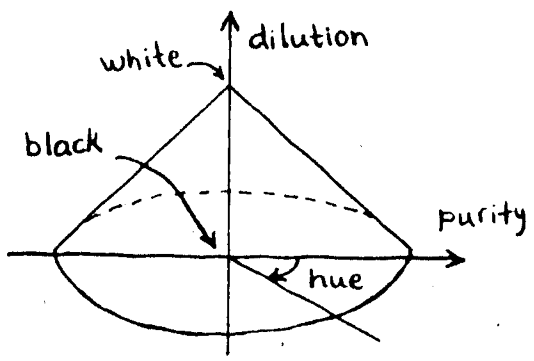

Fig. 15: The purity-dilution-hue colour space, illustrated as a cone.

but simply represents the amount of single-hue colour which has been "blended" with certain amounts of black and white to produce the colour in question. Purity is 0 on the axis and increases radially to 1, and dilution increases vertically (i.e., axially) from 0 to 1. For purposes of graphical illustration, it is natural to shrink the top (white) surface of the cylinder to a point. In the resultant cone there are no paths of zero colour gradient (see fig. 15)

The cylinder model not only encompasses all colours, but in addition makes it possible to define colours as regions bounded by surfaces which are defined simply by keeping one coordinate fixed; i.e., colour regions are pie shaped portions of cylindrical annuli. Of course, in a representation which shrinks the top surface to a point,

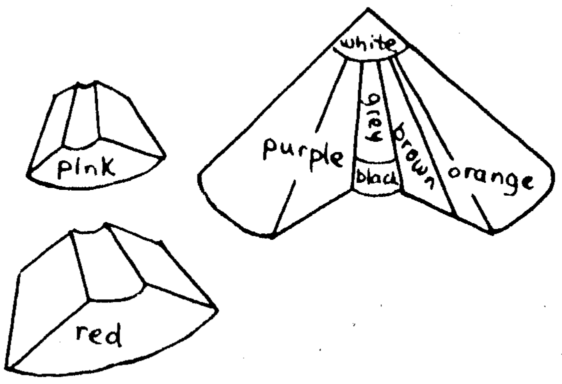

Fig. 16: The cone model as partitioned by the basic colour regions. Here the red and pink regions are taken out to give a view of the interior of the solid.

such regions taper towards the top (fig. 16).

4.4 Inference methods

As emphasized earlier, inference about colours is only a part of a general inference system. The goal is to design a special purpose mechanism that, given two possibly hedged colour predicates, applied to unifiable arguments, will determine whether one is subsumed by the other, or that they are incompatible or that neither is the case ("unknown"). Thus, if we wish to resolve colour predicate ¬M(x) against m(x), for example, we can query the special purpose system for the pair {M,m} and if M subsumes m then the resolution can be carried out.

Going back to question 4 at the beginning of the chapter, after resolving elephant(Clyde) against ¬elephant(x) of the second clause in the usual way, we are left with grey(Clyde) and yellow(Clyde) – these are the two literals supplied to the colour reasoning algorithm. In the model the predicates grey and yellow will correspond to non-overlapping regions so that grey(Clyde) can be resolved against yellow(Clyde) to yield the null clause, thus yielding "¬elephant(Clyde)", in a standard proof by contradiction.

The following examples illustrate the same type of reasoning as applied to a stucturally different kind of question:

1. <u>Given</u> <u>knowledge:</u> tan(x)∨¬elephant(x),
   ¬(sort-of brown)(Clyde),
   plus knowledge about colours

   <u>Question:</u> ?elephant(Clyde)

This states that all elephants are tan and that Clyde is not sort of brown. Since tan is a kind of brown it ought to be possible to prove that Clyde is not an elephant. Resolving the negation of this conclusion against ¬elephant(x) of the first clause we obtain tan(Clyde). The literals to be resolved by the colour reasoning algorithm are now "tan(Clyde)" and "¬(sort-of brown)(Clyde)". The defining region for tan is included in that for brown, so ¬(sort-of brown)(Clyde) can be resolved against tan(Clyde), thus disproving "elephant(Clyde)".

2. <u>Given knowledge:</u> grey(x)∨¬elephant(x),
   ¬(sort-of brown)(Clyde),
   plus knowledge about colours

   <u>Question:</u> ?elephant(Clyde)

This is very similar to question 1, but in this case there is not enough information to answer the question, since the resulting literals are grey(Clyde) and ¬(sort-of brown)(Clyde). It is clearly possible for Clyde to be grey and not sort of brown. Thus we cannot resolve these clauses.

The above examples indicate the need for:

1) a procedure <u>relation</u> which takes two colour predicates A and B and determines the relation of B to A, by comparing their hue, purity and dilution intervals,
2) a table which states whether for a given relation of B to A, and corresponding modes a, b (i.e., "hedged" or "simple", as given by the modifiers for A and B in the literals *A* and *B*) either "*A* includes *B*", or "*A* and *B* are incompatible" or "unknown".

If A and B are incompatible, then they can be resolved and if A includes B, then literal $\neg A$ can be resolved against B.

The relation of B to A can be one of the following:

(i) "apart" - iff one or more of the corresponding intervals for A and B is not overlapping or adjacent

(ii) "adjacent" - iff all of the intervals are adjacent and possibly some but not all are included or overlapping

(iii) "overlapping" - iff all are overlapping and possibly some, but not all, are included

(iv) "included" - iff all of the intervals of B are included in A

(v) "including" - iff all of the intervals of A are included in B

(vi) "centre-included" - iff they are included and none of the corresponding intervals have common endpoints

(vii) "centre-including" - iff they are including and none of the corresponding intervals have common endpoints

The table will contain one entry each for "included" and "including" and for "centre-included" and "centre-including" and the order in which the inputs appear will determine the direction of the inclusion. Thus, when the table is queried the included predicate will always be in the position of B. This is to avoid repetitions.

The reason why we consider "included" and "centre-included" as separate cases is that they do result in different resolving patterns. For example, although both magenta and fire-engine red are special cases of red and their regions are thus included in that for red, anything that is "sort of fire-engine red" is clearly red, but something that is "sort of magenta" maybe too far towards the purple to be considered red.

Another distinction that must be made is between basic and non-basic colours. Detailed criteria for "basicness" are listed and discussed in [Kay & McDaniel 1978], [Mervis & Roth 1981] and [Kay 1981]. We take as basic the following 11 terms and no others: black, grey, white, red, orange, yellow, green, blue, purple, pink, and brown. We feel there is sufficient evidence in everyday usage to assume that these completely partition the colour space. Non-basic colours, such as yellow-green, navy, maroon, etc., sometimes lie across boundaries and in any case overlap one or more basic colours. Regions ocuppied by non-basic colours are generally smaller. A result of this is that all shades included by a non-basic term which spans two basic colours lie near the boundary between these basic colours. Thus, for example, all that is yellow-green is sort of yellow, yet not all that is yellow is sort of yellow-green.

These facts can be dealt with by pre-ordering literal pairs *A*, *B* before table look-up, in a manner dependent on the basic or non-basic status of A and B. Since basic colours always include non-basic ones, rather than vice-versa, and to be consistent with the order already established for the including case, the non-basic term, if any, is taken to be B. If both are basic or both are non-basic and the order is not forced by an inclusion relation, in the case where only one is hedged, the hedged term is taken to be B.

How do hedges affect the relation of B to A once this has been established? A simple model is proposed that ultimately yields answers compatible with most common intuitions. In this model let the region of A be represented by a line segment of one unit length and of B by one of half a unit. A hedge is seen as an "aura" half the segment's length thick on either side of it. In fig. 17 a table is constructed using this model, where the possible combinations of hedges on A and B are enumerated.

The table required for the algorithms can now be constructed from that of fig. 17 by interpreting its entries as "*A* includes *B*" when the line segment of *A* is contains that of *B*, "incompatible" when the line segments do not overlap and "unknown" when there is a partial overalp (fig. 18).

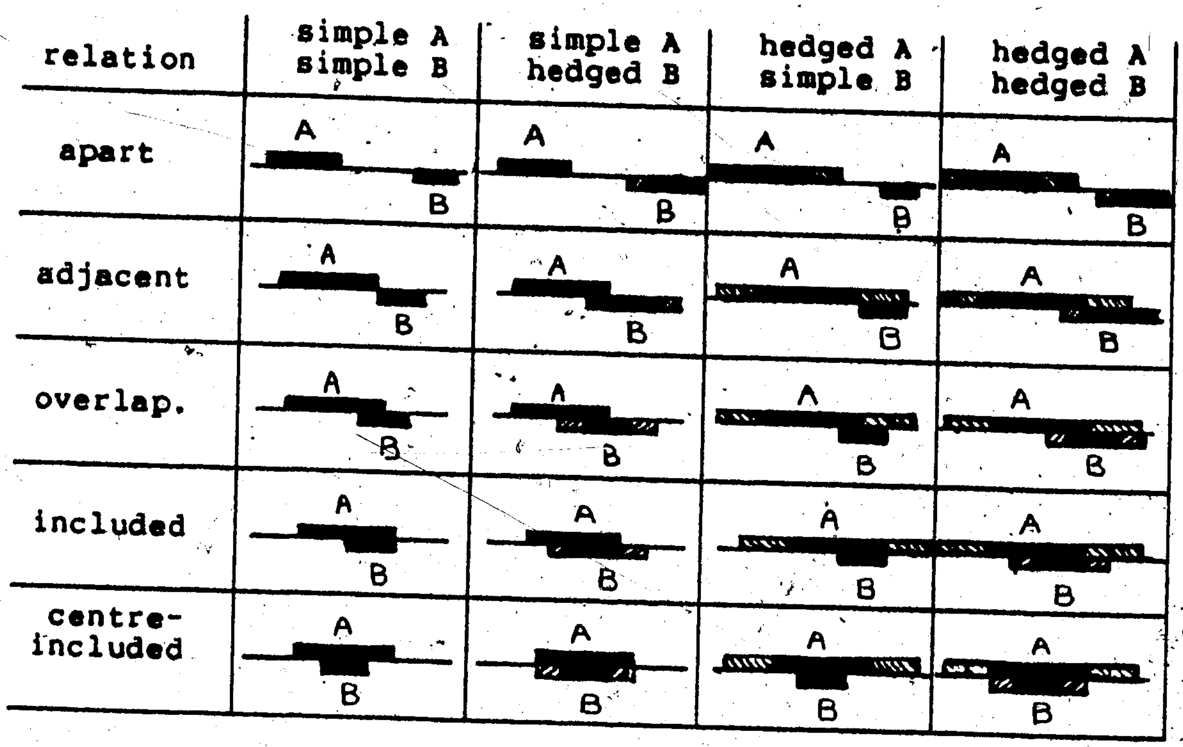

Fig. 17: Regions of colours A and B and their corresponding hedged terms are modelled by line segments.

The algorithm can be summarized as follows:

Given literals aA(x) and bB(x) where A and B are colour predicates and a and b are either "sort-of" or nothing, which correspond to "hedged" and "simple", respectively, first determine the relation between A and B.

Let

basic(x)=true iff x is one of the basic terms;

relation	simple A simple B	simple A hedged B	hedged A simple B	hedged A hedged B
apart	incomp.	incomp.	incomp.	incomp.
adjacent	incomp.	unknown	*A*	unknown
overlap.	unknown	unknown	*A*	*A*
included	*A*	unknown	*A*	*A*
centre-included	*A*	*A*	*A*	*A*

Fig. 18: This table determines the relationship between two literals of the form aA and bB given the relation of the colour regions A, B and their corresponding modifiers a, b. "incomp." means that *A* and *B* are incompatible and "*A*" means *A* includes *B*.

Then proceed as follows:

```
begin
r := relation(A,B);
ba := basic(A);
bb := basic(B);
if (r="centre-included" or r="included") then table(b,a,r);
if (r="adjacent" or r="overlapping") then
     if (¬ba and bb) then table(b,a,r)
     else if (ba=bb and a="hedged" and b="simple")
                                             then table(b,a,r)
else table(a,b,r);
end
```

5. Conclusions

Methods for handling special purpose reasoning have been presented for representing and making inferences in certain classes of taxonomic structures (P-graphs) and for reasoning about possibly hedged colours. Much of the work presented here, however, is not complete yet.

In [Schubert 1980] it is shown that the problem of answering questions about part-of and disjointness relations between nodes of a general P-graph is co-*NP*-complete. This motivates the search for algorithms which answer these questions efficiently for as large a class of P-graphs as possible, and hence the development of semi-closed P-graphs. Note, however, that in proving the co-NP-completeness of these problems, the restriction to fully consistent P-graphs had not been made; thus it is conceivable that methods can be devised to answer these questions efficiently for general fully consistent P-graphs. This would clearly make the foregoing obsolete; thus the co-NP-completeness of the corresponding problem needs to be investigated.

Semi-closed P-graphs are in most cases sufficiently general to accommodate all incoming parts information without an intervening conversion. Nevertheless, algorithms to convert general to semi-closed P-graphs need to be developed; the conversion can be accomplished in a bottom up fashion with relative ease. It should be noted that we are mostly interested in a knowledge assimilating system, so the order of entry of the assertions will be significant.

The restriction that a semi-closed P-graph consist of a semi-closed P-graph with another semi-closed P-graph attached by the main root to one of its nodes could yet be relaxed, leading to a larger class of graphs. In fact, it may be possible to allow intersection of two semi-closed P-graphs at any one node, without forfeiting computational tractability.

Another area of further investigation is the applications of P-graphs to propositional logic and theorem proving. Clause sets can be translated to P-graphs, exploiting the analogy between implication and part-of, and vice-versa. Thus P-graphs may offer a new approach to theorem proving for certain classes of clauses.

From the study of colour reasoning of chapter 4, we conclude that there are classes of predicates which at first seem to require no more than quasi-hierarchical representations, but on closer examination are seen to call for more specialized representations of a quite different sort - in the case of colours, a numerically coded "spatial" representation. The cylinder model proposed allows constant time compatibility checking of various hedged and unhedged colour terms, minimizing the need for combinatorial inference.

The model is also attractive for another reason: the colour cylinder could serve as the interface between the perceptual and conceptual systems of a robot; the required parameters should be quite easy to extract from the primary sensory data, and once extracted, could easily be used to compute an appropriate colour label.

A colour reasoning subsystem has been built which, given a pair of possibly hedged colour terms determines which one subsumes the other, if any, or that they are incompatible, or that the answer is "unknown". The implementation follows closely the description of the model and the algorithms of chapter 4.

6. Bibliography

[1] Birren, F. (editor) (1969), "Ostwald. The Color Primer". Van Nostrand Reinhold Co., New York.

[2] Birren, F. (editor) (1969), "Munsell. A Grammar of Color". Van Nostrand Reinhold Co., New York.

[3] Boolos, G., Jeffrey, R. (1974), "Computability and Logic". Cambridge University Press, London, pp.99-191.

[4] Bundy, A., (1973), "Doing Arithmetic with Diagrams". Proc. 3rd IJCAI, pp 130-136.

[5] Bunt, H. C., (1979), "Ensembles and the Formal Semantic Properties of Mass Terms". Pelletier, F. J., (editor), Mass Terms, pp 249-277.

[6] Covington, A. R. and Schubert, L. K., (1979), "Organization of modally embedded propositions and of dependent concepts". Proc. of the 3rd Bienn. Conf. of the CSCSI/SCEIO, Victoria, B.C., May 14-16, 1980, pp 87-94.

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

Cambridge, Ma.

[8] Fikes, R. E., and Hendrix, G. G., (1977), "A Network-based Knowledge Representation and its Natural Deduction System" Proc. 5th IJCAI, pp 235-246.

[9] Grice, H. P., (1975), "Logic and conversation". In Davidson, D. and Harman, G. (editors), The Logic of Grammar, Dickenson, Encino, CA, pp 64-75.

[10] Hayes, Philip J. (1977), "On semantic nets, frames and associations", Proc. 5th IJCAI, MIT, Cambridge, Ma., Aug. 22-25, pp. 99-107.

[11] Hendrix, G. G., (1979), "Encoding Knowledge in Partitioned Networks". Findler, N. V. (editor), Associative Networks - The Representation and Use of Knowledge by Computers, Academic Press, pp 51-92.

[12] Judd, D. and Wyszecki, G., (1963). "Color in Business, Science and Industry" (2nd ed.). John Wiley and sons, New York.

[13] Kahn, K. and Gorry, G. A., (1977). "Mechanizing Temporal Knowledge", Artificial Intelligence 9, pp 87-108.

[14] de Kleer, J., and Brown, J. S., (1982), "Assumptions and Ambiguities in Mechanistic Mental Models". Tech. Report, Xerox, Palo Alto.

[15] de Kleer, J., and Brown, J. S., (1982), "Foundations of Envisioning". Proc. 2nd. AAAI, August 16-20 1982, Pittsburgh.

[16] Kay, P. and McDaniel C. K., (1978), "The linguistic significance of the meanings of basic colour terms". Language 54, pp,610-146.

[17] Kay, P., (1981), "Colour perception and the meanings of colour words". Proc. of the 3rd Annual Conf. of the Cognitive Science Society, Berkeley, Calif., Aug. 19-21, 1981, pp 61-64.

[18] Lindsay, R. K., (1973), "In Defense of Ad Hoc Systems". Schank, R. G. (editor), Computer Models of Thought and Language, pp 372-395.

[19] Mervis, C. B. and Roth, E. M., (1981), "The internal structure of basic and non-basic color categories". Language 57, pp 383-405.

[20] McSkimin, J. R. and Minker, J., (1979) "A predicate calculus based semantic network for deductive

searching", Findler, N. V. (editor). Associative Networks - The Representation and Use of Knowledge by Computers, Academic Press, pp 121-175.

[21] Papalaskaris, M. A. and Schubert, L. K. (1981), "Parts inference: Closed and semi-closed partitioning graphs". Proc. 7th IJCAI, Vancouver, B.C., Aug. 24-28, pp 304-309.

[22] Papalaskaris, M. A. and Schubert, L. K. (1982), "Inference, Incompatible Predicates and Colours". Proc. 4th Bienn. Conf. of the CSCSI/SCEIO, Saskatoon, Canada, 1982, pp 97-102.

[23] Quillian, M.R. (1968), "Semantic memory" in Minsky, M.L.(ed.), Semantic Information Processing, MIT Press, Cambridge, Ma., pp. 227-270.

[24] 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.

[25] Reiter, R. (1973), "A Semantically Guided Deductive System for Automatic Theorem Proving". Proc. 3rd IJCAI, pp 41-46.

[26] Schubert, L. K., (1979), "Problems with parts". Proc. 6th IJCAI, Tokyo, Aug. 20-23, pp 778-784.

[27] Schubert, L.K. (1980), "Representing and using knowledge about parts". Unpublished manuscript, University of Alberta, Edmonton, Alberta.

[28] Schubert, L. K., (1976), "Extending the expressive power of semantic networks". Artificial Intelligence 7(2), pp 163-198.

[29] Schubert, L. K., Goebel, R. and Cercone, N., (1979). "The structure and organization of a semantic network for comprehension and inference". Findler, N. V. (editor), Associative Networks - The Representation and Use of Knowledge by Computers, Academic Press, pp 121-175.

[30] Taugher, J., and Schubert, L. K., (1982). Personal communication.