# PARTS INFERENCE: CLOSED AND SEMI-CLOSED PARTITIONING GRAPHS

Mary Angela Papalaskaris Lenhart Schubert

Department of Computing Sciança
University of Alberta
Edmonton. Alberta T6G 2H1

## MilBASI

Ma consider the problem of answering part-of questions and questions about overlap in partitioning structures. which is of importance in A.I. systems knowledgeable about parts relationships, set inclusion relationships or taxonomies of types in an earlier paper 1t was noted that the problem of extracting information from arbitrary sets of partitioning assertions ("P-graphs") Is intractable (at least if P = NP) and the more restrictive class of quasi-hierarchical <u>closed</u> P-graphs was introduced as a fairly flexible representation of partitioning structures permitting efficient information extraction. The present paper introduces the larger class of <u>semi-closed</u> P-graphs. and provides efficient and complete methods for answering part-of and disjointness questions based on such P-graphs

## I INTRODUCTION

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 1f any two members of the group envy the same person then they love each other At, Bill. Cecil and Dld1 are members of the group, end A) envies B111. Cecil envies Bill, and Oidi envies Cecil. Doee Dldi 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). Vet from a logical point of view (1)-(3) are very much the same kinds of problems, namely problems of inferring inclusion or dlsjomtness relationships in taxonomlc structures, and (2) probably requires as many inference steps as (3) Note that it would be implausible to suppose that people recall that sulphur 1s 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) 1s solved by more laborious general methods. Which type of method 1s used 1s a matter of familiarity: if we have reflected on the relationships between Al. Bill. Cecil and Old! - 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 A.I systems: if such systems are to use taxonomlc 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, we see this as an important challenge in A.I,, given the ubiquity of parts hierarchies and concept hierarchies in virtually all fields of knowledge

Of course, a great many past and present A,I. systems have made allowance for hierarchies of various kinds For example. Raphael's SIB[1] effectively exploited the translvity of part-of relationships and QuillMan's Semantic Memory(2) organized concepts as "subset-superset" taxonomies; (neither. Incidentally, paid much attention to possible exclusion relationships among subparts or subconcepts) More recently Philip Hayes(3] has developed network structures and techniques for using knowledge about part-of relationships, and Fahlman(4j 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(5,6] studied sets of partitioning assertions of the form [a P af ..an], meaning that object a 1s partitioned into disjoint parts a1. ..,an, with P defined in terms of a part-of relation "c, 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 1s surprising 1f 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-assert ions 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 1n which all downward paths terminate at the leaves of some common "main" hierarchy whose root represents the merge of all parts mentioned Examples given in [5J illustrate how closed P-graphs can represent overlap parts and "multiple views" of the same object. They can. of course, also represent partitionings based on relations other than the part-of relation, as long as these relations satisfy the assumed properties of "part-of" (as mentioned later, they must induce boolean lattices). This includes the subset-of relation and the subconcept-of (IS-A) relation commonly used in taxonomies of types

While allowing some tangling of hierarchies, closed P-graphs still admit very efficient (linear or sub11near) inference methods for questions of type ?(a part-of b) or ?(a disjoint-from b). as outlined in (5]. Moreover, these methods are provably complete This partially solves the problem originally addressed.

The purpose of the present paper 1s twofold. The first objective is to shore up the theoretical foundations

• This work was supported by NSERC operating grant AS818

of the earlier work by supplying axioms for the part-of relation.* stating soma immediate consequences and carefully defining various kinds of P-graphs and relevant notions. This is the subject of sections 2 and 3. Later (in section 6) we also Illustrate the model-theoretic techniques 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 m (S) (and proved tn [6]) 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', ..,a?5 of certain computer science department C are, and also knows that the department divides organizationally into a chairman cf. an advisory committee *c2,* a library committee *c3.* a colloquium committee c-?, and graduate and undergraduate committees, c5 and c6. 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. •• Note that not all paths 1n 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 dosed graph would Introduce 90 new parts in addition to the 22 already present! (We are Ignoring constraints such as that cf. the chairman, must equal one of af,...,af5 and that each *ci* must consist of a subset of the af, for simplicity). This 1s because

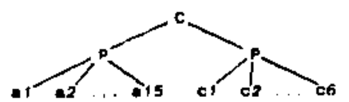


Fig.1 A simple non-closed P-graph

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 ona view may overlap with parts 1n the other Tha question-answering algorithms rely on the presence of these overlap nodes Vet it 1s obvious that part-of questions and dlsjolntness questions can be answered very easily for the original graph; everything 1s part of C, and wlthin each of the two part11onmgs all distinct parts are disjoint while for parts ai. cj. taken from both part1tlonlngs. the correct answer to ?(a/ part-of cjj or ?ta/ dlsjolnt-from c)) Is "unknown". A reduction to closed graphs would only obscure the logic of the requisite reasoning process.

A similar example would be provided by a partly functional and partly anatomic representation of brain structure m 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, political, or abstract entity are easily constructed, and could easily arise In an AI system, particularly one which 1s fed its knowledge piecemeal.

This motivates the Introduction of recursively defined semi-closed P-graphs m section 4. A semi-closed P-graph is either a closed *P-Qr»ph,* 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 1s a semi-closed P-graph, since it consists of the closed committee-structure subgraph attached by its main root C to the closed faculty-roster subgraph

In Sec. 5 efficient complete algorithms for answering part-of and disjomtness questions on the basts of semi-closed P-graphs are developed. We feel that the class of semi-closed P-graphs is probably as large a class of P-graphs as 1s needed for most practical applications to taxonomlc structures, and as can be easily mechanized, as far as answering part-of and disjomtness questions 1s concerned.

* We take this opportunity to correct *mn* error In (3): In the first sentence of Sec. 2. the assumption that part-of has the extension property should be replaced by the assumption that the merge and overlap functions "U* and *rt[N] are mutually distributive (see below).

•• For the purposes of this Illustration, we intend C to be interpreted as the (disconnected) physical whole composed of the department members, not as a set. Thus the ef and *ci* are parts of C. not elements or subsets.

## II THE PART-OF RELATION, PARTITIONINGS AND P-GRAPHS

The part-of relation, $\sqsubseteq$, that the given methods rely upon is assumed to have the following properties [6]:

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

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

$(\forall xy)[[[x \sqsubseteq y] \& [y \sqsubseteq x]] => [x = y]]$

$(\forall xyz)[[[x \sqsubseteq y] \& [y \sqsubseteq z]] => [x \sqsubseteq z]]$

(ii) Existence of a unique empty object $\theta$ such that

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

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

$(\forall xyz)[[z \sqsubseteq (\sqcap xy)] <=> [[z \sqsubseteq x] \& [z \sqsubseteq y]]]$

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

$(\forall xyz)[[(\sqcup xy) \sqsubseteq z] <=> [[x \sqsubseteq z] \& [y \sqsubseteq z]]]$

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

$(\forall xyz)[[z = (\backslash xy)] <=> [[x = (\sqcup z(\sqcap xy))] \& [(\sqcap yz) = \theta]]]$

(vi) Mutual distributivity of $\sqcap$ and $\sqcup$:

$(\forall xyz)[(\sqcap x(\sqcup yz)) = ((\sqcup(\sqcap xy)(\sqcap xz))]$

$(\forall xyz)[(\sqcup x(\sqcap yz)) = ((\sqcap(\sqcup xy)(\sqcup xz))]$

It is shown in [6] that the assumed properties of the part-of relation induce a boolean lattice on the set $\{x | x \sqsubseteq w\}$ 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) $(\forall xy)[[\sqcup xy = \sqcup yx] \& [\sqcap xy = \sqcap yx]]$

(b) $(\forall xy)[x \sqsubseteq y <=> [\sqcap xy = x \& \sqcup xy = y]]$

(c) $(\forall xyz)[y = \sqcap xz <=> [(\sqcup y(\backslash xz)) = x \& (\sqcap y(\backslash xz)) = \theta]]$

(d) $(\forall x)[\sqcap x\theta = \theta \& \sqcup x\theta = x]$

(e) $(\forall xy)[(\sqcup x(\sqcap xy)) = x \& (\sqcap x(\sqcup xy)) = x]$

(f) The functions $\sqcap$ and $\sqcup$ are associative.

For brevity, $\sqcap$ and $\sqcup$ will be informally used as many-place functions since, from (vi), 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 partitioning assertion is of the form $[x\ P_m\ y_1 \ldots y_m]$, $m \geq 2$, where $x, y_1, \ldots, y_m$ can be constants or variables of the object language and $P_m$ is the $m+1$ place predicate symbol defined by:

$(\forall xyz)[[x\ P_2\ y\ z] <=> [x = \sqcup yz \& \theta = \sqcap yz]]$

and for all $m \geq 2$:

$(\forall y_1 \ldots y_m)[[x\ P_m\ y_1 \ldots y_m] <=> [[x\ P_{m-1}\ (\sqcup y_1 y_2)\ y_3 \ldots y_m] \& \sqcap y_1 y_2 = \theta]]$

Thus, by definition, and (i) above, the order of the y's is immaterial. "P" will be used for "P2"

Partitioning assertions are the building units of P-graphs.

<u>Definition</u>: 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≠∅] over constants of the object language. The distinct constants **x**, **y**, **z** 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 descendent relation defined below. Boldface symbols will be used in the metalanguage in order to stress this distinction.

It is assumed throughout this paper 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, ∅, predicate symbols ⊑, Pm, =, function symbols ∩, ∪, \ and the logical connectives is either provable ($G \vdash w$) or disprovable ($G \vdash \neg w$) [8]

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.

## III <u>HIERARCHIES AND CLOSED P-GRAPHS</u>

Simply stated, a descendant of a node "a" of a P-graph is any node from which there is an upward path in the graph that leads to "a". Similarly, a hierarchy is a P-graph that takes the form of a tree with no more than one P-assertion about each node; a closed graph is one in which all nodes are ultimately partitioned into a set of disjoint "leaves".

These notions are made formal in the following definitions:

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

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

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

We say that **x** is a <u>direct descendant</u> of **y** if there is an assertion in G of the form:
[y Pk x z1...zk-1] for some k. (Of course, x need not be the first argument following Pk).

The <u>ancestor</u> relation is the inverse of the descendant relation.

<u>Definition</u>: A <u>leaf</u> is a node that has no descendants other than itself.

<u>Definition</u>: A <u>root</u> is a node that has no ancestors other than itself.

<u>Definition</u>: A P-graph G is <u>acyclic</u> if and only if for any two nodes **x** and **y** of G, if **x**$\le$**y** and **y**$\le$**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.* Thus cycles are easily eliminated by collapsing the cyclic nodes.

Graphs in which some node is provably empty are of relatively little practical A.I. 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.

<u>Definition</u>. A P-graph is <u>fully-consistent</u> iff none of its nodes can be proven to be empty.

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

<u>Definition</u>: A <u>path</u> is a set of nodes of a P-graph ordered by the direct descendant relation.

<u>Definition</u>: A <u>hierarchy</u> 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. Hierarchies are the most desirable form of P-graph, because parts-reasoning for hierarchies is trivial: in a hierarchy x⊑y is provable for any two nodes **x**, **y** if and only if **x**$\le$**y**. Further, x∩y=∅ is provable if and only if there is no path connecting **x** and **y**, and x=y is provable if and only if **x**=**y** i.e., all the nodes represent possibly distinct objects. It can also be shown that hierarchies are fully consistent and that any fully consistent simple graph is a hierarchy [7].

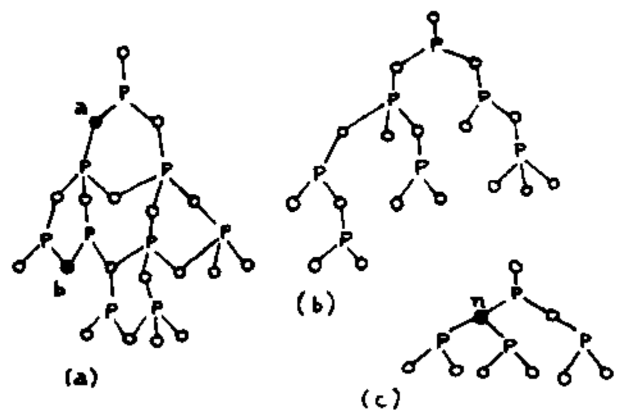


Fig. 2 General P-graphs and hierarchies. (a),(b) are simple; (b) is a hierarchy but (a) is not since there are two paths between "a" and "b". (c) is not simple, since there are two assertions about "n".

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'. Reasoning about arbitrary P-graphs is known to be co-NP-complete. This problem is dealt with in [5] by reducing arbitrary P-graphs to <u>closed</u> P-graphs: G is closed iff any two of its nodes are projectible into a common subhierarchy. A node n is <u>projectible</u> into a subhierarchy H if G contains a subhierarchy rooted at n whose leaves lie in H. In theory, a single parts node will be regarded as a subhierarchy, although, strictly speaking, it cannot stand on its own as a P-graph, according to the definition. Thus any node is trivially projectible into any subhierarchy to which it belongs. Hence any two nodes of a subhierarchy H are projectible into a common subhierarchy, viz., H.

-------------------

* Formulas involving object language and metalanguage symbols are interpreted as object language <u>formulas</u> (symbol sequences), i.e., the object language symbols are <u>mentioned</u> rather than <u>used</u>. Thus x=y (unlike **x**=**y**) denotes an object language formula, rather than a proposition.

The Projection of a node n into the leaves of a closed P-graph Q is the largest subset L of the leaves of G that are also leaves of a subhlerarchy rooted at n.

It is shown in (6) that for every P-graph there 1s an equivalent closed P-graph. Inference methods are given to answer the questions ?[b part-of a) and ?(a dlsjolnt-from b) for fully consistent closed P-graphs. 1n linear space-time relative to the number of edges of the closed graph.

It 1s proved 1n [6] 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 root</u> of the closed p-graph.

## IV SEMI-CLOSED P-GRAPHS

Semi-closed P-graphs relax soma of the restrictions of dos«d P-graphs, thus forming a larger class. The tacit restriction to fully consistent graphs should be kept 1n mind.

<u>Def mi t lon</u> A <u>semi-closed</u> 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 sent 1-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 (5]

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 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 of Sec. 5 have been preprocessed, with labels being attached to all nodes which Indicate their position 1n 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 1s r' and q', while for r and s the corresponding pair (s r' and s.

We have put "ancestors" 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<r'* or r 1s projectible Into a set of nodes n», ...nk such that for all 1, 1<1<k either *ni<r'* or r' 1s an "ancestor" of n» (see Fig 4)

## V ALGORITHMS FOR EXTRACTING INFORMATION FROM SEMI-CLOSED P-GRAPHS

Algorithms for answering the questions *?[x* part-of yj and ?(x disjolnt-from y] in fully consistent closed P-graphs have been developed in [5] and 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 1s desired for the new algorithms.

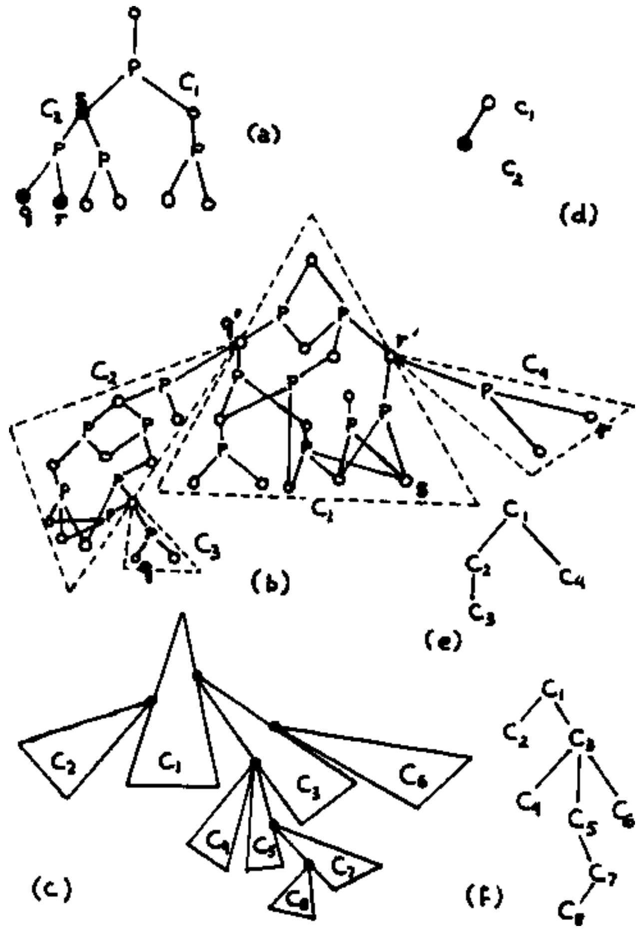

Fig. 3 Some examples of semi-closed P-graphs. In (a), the (closed) P-graph consisting of nodes s. q and r 1s joined to the rest of the graph only through s, end no other nodes. Similarly 1n (b) there 1s e main closed P-graph with two other P-graphs attached to it. one of which is itself a closed P-graph 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, 1s emphasized (d),(e),(f) corresponding trees for the P-graphs of (a),(b),(c).

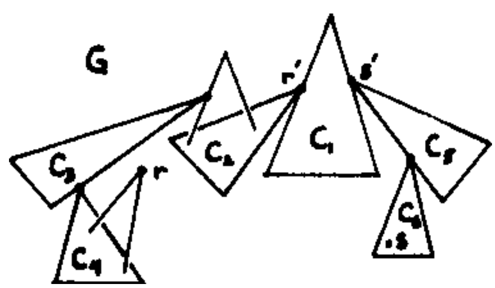

Fig. 4 The semi-closed P-graph G has closed subgraphs C1.....C6. the node r belongs to C4 but 1s not a descendant of the main root of C4, It is a "descendant" of r' as defined in this section.

Algorithms *P(x,y)* and D(x.y) make use of a predicate N($x_1$....xn) which is true if the merge of xr,...,xn 1s <u>provably</u> non-empty and false otherwise. It was noted in [5] 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, wa need to assume that N 1s still efficiently decldeble. with tha provability requirement now referring to the entire semi-closed graph. The assumption is justified since the only changes In the truth values of N(xf. . . . $x_n$) ovar nodas of a closed graph C. resulting from attachmant of semi-closed P-graphs to C, ara those duo to tha non-amptinesa of nodas to which a semi-cloaad graph containing a provably non-ampty noda was attachad (this information propagatas "upward" in tha traa of cloaad graphs); and tha only changa potentially resulting from tha attachmant of C to • semi-closed P-graph is that dua to provable non-amptmass of tha noda to which C was attachad (this information propagatas "downward" via main nodas which ara points of attachmant 1n tha traa of closed P-graphs). Tha amptlnass assartions thus nacassMated at points of attachmant by tha upward and downward flow of Information can ba computed (n ona "pass* aach ovar all tha nodas of tha aeml-cloaad P-graph, (n tha worst ease

In tha following algorithms tha test "a*b" 1s 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 cloaad subgraph, and falsa otherwise.

"a"b" incorporates a recursive call to the algorithm ?(a part-of b] to determine whether the answer to the question "a-b?" (i.e.. do the nodes a and b denote tha same object?) is "yes". 1n cases where it (s already known that b 1s part of a; (this test is only used where a 1s an ancestor of b, as for x' and x). Let x'. *y* denote tha nearest pair of "ancestors" which belong to a common cloaad subgraph for x and y respectively, as described in tha 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"
```

## VI CORRECTNESS OF THE PROPOSED METHODS

This section is devoted to showing how much can be inferred from P-graphs and, in particular, proving the question answering methods given in the previous section correct and complete.

The logical foundations of these proofs have been briefly discussed in section 2. 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 $\phi$ is derivable as a theorem iff $\phi$ 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 U, ∩, \, the relation ⊏ 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.

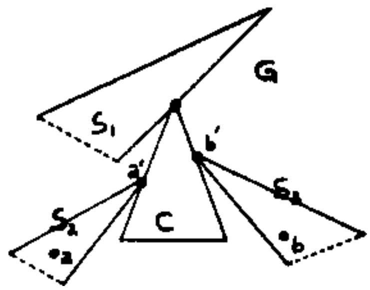


Fig. 5 G is a fully consistent semi-closed P-graph. Subgraphs S1, S2, S3 are semi-closed, and C closed. a' and b' belong to the same closed subgraph C. so P(a',b') and D(a',b') are used. b and b' belong to a smaller semi-closed subgraph, thus the algorithms can be applied recursively to determine whether G ⊢ b=b'. Thus if G ⊢ b=b', then if P(a',b')="yes" the algorithm yields "yes" for ?[a part-of b].

The completeness and correctness of the given algorithms relates to the derivability of statements of the form a ⊏ b, a ⊄ b, ∩ab=Θ or ∩ab≠Θ. To show that the methods are complete and correct it is necessary to prove that for an assertion $\phi$, the algorithm will return "yes" if $G \vdash \phi$, "no" if $G \vdash \neg\phi$, and "unknown" otherwise. The following lemma illustrates the sorts of techniques used and facilitates the proofs of correctness and completeness.

<u>Lemma</u>1: Let G be a fully consistent closed P-graph. Let I be an interpretation that assigns functions U, ∩ and \ to the symbols U, ∩, and \, and the relation ⊑ to the symbol ⊏ 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 the interpretations for two leaves n and k of G, respectively, then: ∩nk=Θ

(ii) if n and n1,...,nk are the interpretations of a node n and its projection n1,...,nk onto the leaves of G, respectively, then: n=(Un1,... ,nk)

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

<u>Proof</u>: Let [n P n1,...,nk] be an assertion of G, and let N1,. . ,Nk be the (sets) projections of n1,...,nk into the leaves of G, respectively.

From (ii)

∩ninj = ∩(UÑi)(UÑj)

where Ñi and Ñj are sets of interpretations of the nodes in the sets Ni and Nj respectively (U is informally used here as an operator on sets).

Since G is a closed P-graph therefore Ni∩Nj=ø (for if x∈Ni∩Nj then x=Θ, since G ⊢ ∩ninj=Θ). Thus from (i) ∩ninj=Θ.

Now Un1...nk = U(UÑ1)...(UÑk)

= (U(Ñ1U...UÑk))

Furthermore Ñ1U...UÑk is the projection of n into the leaves of G, since:

1) any leaf in the projection of ni is also in the projection of n since it is a descendant of n.
2) let "e" be a leaf in the projection of n

∩en=e,

Thus G ⊢ ∩e(Un1...nk)=e,

G ⊢ ∩e(U(UN1)...(UNk))=e

hence G ⊢ U(∩e(UN1)...∩e(UNk))=e.

But if $a \neq x$ for all leaves $x$ in the union of the projections of the $ni$'s, then $G \vdash \cap ax = \theta$ so $a = \theta$ and G would not be fully consistent. Thus a must be in the projection of one of the $ni$'s.

So, by (1), $(\cup(\hat{n}1 \cup \ldots \cup \hat{n}k)) = n$

therefore, $\cup n1 \ldots nk = n$

Hence, $I \vdash (n \; P \; n1 \ldots nk)$

The following lemmas sketch the proof of correctness and completeness of the algorithms; these are proved in [7], by methods very similar to those employed in the proof of lemma 1.

<u>Lemma</u>2: A semi-closed graph G consisting of fully consistent semi-closed graphs T and S, such that S is attached to T by its main root only, is fully consistent.

<u>Proof</u>: The proof of this lemma makes use of theorem 2 of [5] which states that for every P-graph there is a logically equivalent closed P-graph. Once dealing with closed P-graphs we can construct a model in which all the nodes have non-empty interpretations.

For the following three lemmas let G be a semi-closed fully consistent P-graph. Let S and T be closed subgraphs of G, both rooted at some node "a" with no other common nodes, and suppose G without T is a closed subgraph.

<u>Lemma</u>3: Let $s1, \ldots, sn$ and $t1, \ldots, tn$ be nodes of S and T respectively. For any $si$, $tj$, $G \nvdash \cap si \, tj = \theta$

<u>Proof</u>: The same model theoretic methods are used in this proof. The idea is to suppose there is a model of G that satisfies $\cap s1 tn = \theta$ and from that construct another model that satisfies $\cap s1 tn \neq \theta$.

<u>Lemma</u>4: Let x and y be nodes of S and T respectively.

(a) $G \vdash y \sqsubset x \Rightarrow G \vdash a = x$

(b) $G \vdash x \sqsubset y \Rightarrow G \vdash a = y$

<u>Proof</u>: Straightforward.

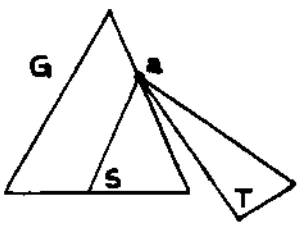


Fig. 8 Diagram for lemmas 4,5 and 6.

<u>Lemma</u>5: If y is a node of T and x is a node of G not in T, and $G \nvdash a \sqsubset y$ then:

(a) $G \vdash y \sqsubset x$ iff $G \vdash a \sqsubset x$

(b) $G \vdash x \sqsubset y$ iff $G \vdash x \sqsubset a$ and $G \vdash a \sqsubset y$

(c) $G \vdash y \not\sqsubset x$ iff $G \vdash \cap xa = \theta$

(d) $G \vdash x \not\sqsubset y$ iff $G \vdash x \not\sqsubset a$

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

(f) $G \vdash \cap xy \neq \theta$ iff $G \vdash a \sqsubset x$ and $G \vdash y \neq \theta$

<u>Proof</u>: Straightforward.

<u>Theorem</u>: This theorem relaxes the restriction that the subgraphs T,S and G without T be closed, in lemma 5.

<u>Proof</u>: This makes reference to the theorem 2 from [5].

For the following corollaries to the theorem let G be a fully consistent semi-closed P-graph with subgraphs Ta, S, Tb such that Ta and Tb 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 Ta, Tb respectively, such that $G \nvdash a' \sqsubset a$ and $G \nvdash b' \sqsubset b$.

<u>Corollary</u>1: $G \nvdash a \sqsubset b$

<u>Corollary</u>2: $G \vdash a \not\sqsubset b$ iff $G \vdash \cap a'b' = \theta$

<u>Corollary</u>3: $G \vdash \cap ab = \theta$ iff $G \vdash \cap a'b' = \theta$

<u>Corollary</u>4: $G \nvdash \cap ab \neq \theta$

The correctness and completeness of the proposed algorithms follows from the above theorem and its corollaries.

## VII <u>CONCLUDING REMARKS</u>

In (6) it 19 shown that the problem of answering questions about part-of and disjointness relations between nodes of a general P-graph 1s co-NP-complete. This motivates the search for algorithms which answer these questions 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 *9rm* 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 m a knowledge assimllatmg' 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 <u>main</u> root to one of its nodes could yet be relaxed, leading to a larger class of graphs

Another area of further investigation is the applications of P-graphs to propositions! 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.


## <u>ACKNOWLEDGEMENTS</u>

We wish to thank Leo Hartman for the help he has offered through discussions of this work and for his assistance in preparing the final draft of this paper.


## <u>REFERENCES</u>


[1] 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.

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

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

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

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

[6] Schubert, L.K. (1981), "Representing and using knowledge about parts", in preparation as Computing Science Tech. Note, University of Alberta, Edmonton, Alberta.

[7] Papalaskaris, M.A. (1981), Master's thesis in preparation, University of Alberta, Edmonton, Alberta.

[8] Boolos, G., Jeffrey, R. (1974), "Computability and Logic", Cambridge University Press, London, pp.99-101.