Reasoning with Graph Operations

Roger T. Hartley, Michael J. Coombs
Computing Research Laboratory
New Mexico State University
Las Cruces, NM 88003

ABSTRACT

Problem solving is an analog to scientific method, wherein abduction and deduction operate in a cyclic fashion to generate and refine a series of hypotheses that purport to explain the observed data. Model Generative Reasoning implements this cycle through a family of operations on representations based on conceptual graphs. Specialize, the operator that implements abduction generates alternative hypotheses. Fragment removes potential incoherences from hypotheses, while preserving coherence with the observations. This is seen as a form of deduction with the aim of allowing more hypotheses to be generated in the next cycle.

1. Problem solving and noisy data

In essence, problem solving is a two-part process. There is a data-driven side, where making observations, filtering them and generating hypotheses from them is important. There is also another side, that is more goal-driven; evaluation of hypotheses against new data can, and does lead to reformulation or even rejection of these hypotheses. Typically problem solving machines are good at the first but less so at the second part. Heuristic search enables this sort of efficient generation, leading to excellent systems for planning, where the prime aim is to reach some desired goal without interference from the environment (Chapman, 87). When, however, data are incomplete, incoherent (i.e. unexpected), or just irrelevant, such techniques will fail.

In these difficult cases, what is needed is a principled way of handling noisy data. The gaps in incomplete data should be filled, the incoherencies removed and the irrelevancies ignored. We will present, in this paper, mechanisms for doing all of these, and place these mechanisms in a problem solving framework. In order to do this we will contrast our approach with that based on theorem proving. In particular the nonmonotonic approach to noisy data through the ATMS (Reiter and de Kleer, 87) can be seen as only solving one of the problems associated with noisy data: incoherence that presents itself through logical inconsistency.

1.1. Hypotheses, explanation and abduction

All problem solvers generate hypotheses, and in general we can classify all such mechanisms as abductive. However, the use to which these subsequent hypotheses are put separates what we might called logical abduction from the more pragmatic use of the term in scientific reasoning (Peirce, 57).

Many authors have pointed out that abduction is an unsound logical inference from consequent to antecedent, as in:

$$\frac{\begin{array}{l} A \rightarrow B \\ B \end{array}}{A}$$

One mechanism that can implement this inference is used extensively in backwardchaining expert systems and is the basis of Prolog's proof technique. We distinguish here between the inference technique and the mechanism used to implement it. Prolog is deductive, but uses a backward-chaining mechanism. Most of these, however, set up the A's as intermediate goals to be carried on further in the method i.e. they are not asserted as true, or as possibly true. Another way to look at such a inference is that if B is true, and A -4 B, then A explains B. This is the basis of the methods described by Levesque (Levesque, 89) and Poole (Poole, 89). Explanations are hypothetical structures generated to fit some set of observations, (not just one, as the above simplification implies) and have only the status of 'possible' in the system. However, if they were true, then the consequent would follow naturally by deduction. Finding an explanation in a logical system then amounts to finding an expression, that if it were true would imply the input (the axioms). In order to find out how this works, we need to analyze the use of rules (logical implications) in such systems.

There are four main uses of a rule:

1. as a selectional constraint on types, e.g. all U's are V's:

$$\forall x\ U(x) \rightarrow V(x)$$

2. as an Aristotelian definition, e.g. if something has properties A, B, C etc. then it is a V

$$\forall x\ A(x) \wedge \mathrm{B(x)} \wedge \mathrm{C}(x) \ldots \rightarrow V(x)$$

3. as a contingent, or schematic definition, e.g. if something is a V, then it has properties A, B, C etc.

$$\forall x\ V(x) \rightarrow A(x) \wedge \mathrm{B(x)} \wedge \mathrm{C}(x) \ldots$$

4. to express causality, e.g. if P, Q, R all happen, then X, Y, Z will happen as a direct consequence

$$P \wedge Q \wedge R \ldots. \rightarrow X \wedge Y \wedge Z \ldots$$

To illustrate these types of rule, consider the following abductive inferences:

$$\frac{\begin{array}{l} car(a) \\ \forall x\ ford\ (x) \rightarrow car\ (x) \end{array}}{possible\ (ford\ (a))}$$

In other words, if a car a is observed, then it is possibly a ford. Of course, this is not an explanation of why *a is* a car, but it does shed a little more light on the subject. There may well be other types of car (chevrolet, subaru etc.) and these would be equally likely inferences. The question of the goodness of an explanation is one for the pragmatics of abduction. Poole has pointed out, however, (Poole, op. cit.) that there are several possible accounts of what constitutes the best explanation.

2.

$$\frac{\begin{array}{l} car(a) \\ \forall x \ has\text{-}wheels(x) \wedge has\text{-}engine(x) \wedge has\text{-}drivers\text{-}seat(x) \rightarrow car(x) \end{array}}{possible(wheels(a) \wedge has\text{-}engine(a) \wedge has\text{-}drivers\text{-}seat(a))}$$

This abductive inference stands a little better as an explanation; it at least shows why *a* is a car, based on the limited knowledge to hand about cars. Note that if both of the above rules were available and a criterion of goodness was expressed as the simplest explanation is the best, then the first would be preferred over the second.

3.

$$\frac{\begin{array}{l} at(a,b) \\ \forall xyz \ person(x) \wedge car(y) \wedge drive(XY) \wedge location(z) \rightarrow at(x,z) \wedge at(y,z) \end{array}}{possible(\exists y \ person(a) \wedge car(y) \wedge drive(a,y) \wedge location(b))}$$

Here we infer an existential result (as pointed out by Poole) because we have no evidence about the car if we assume a is a person. The other possible inference is:

$$possible(\exists y \ person(y) \wedge car(a) \wedge drive(y,a) \wedge location(b))$$

It is also possible that the observations are existentially quantified. e.g. with the fact: 4.

$$\exists xy \ at(ax) \wedge drive(y,b)$$

This, with the same rule as in 3, gives the inference:

$$possible(\exists x \ person(a) \wedge car(b) \wedge drive(a,b) \wedge location(x))$$

Finally an example that gets closer, we believe to the reason why abduction is important. If we have the facts:

5.

at(a,b) and

has -engine(c)

which seem to be unconnected, the job of abduction is to *glue* them together in a single hypothesis. The inference we might look for, and one that is clearly an explanation of why these two pieces of data are observed together is:

$$person(a) \wedge car(c) \wedge drive(a,c) \wedge location(b) \wedge a(c,b)$$

This hypothesis glues the facts together, through the definition of a car and the driving rule.

2. An operator for abduction: specialize

We will describe an single operator, *specialize,* which mechanizes the process of abduction illustrated in the above examples. It is composed of two more primitive operators, *cover* and join that operate on conceptual graphs (Sowa, 84). This leads to the slogan:

Abduction = cover + join

Conceptual graphs are connected, directed, bi-partite graphs where the nodes are labeled with either a concept type or a relation name. There are restrictions on the edges, however, that are used to preserve semantic coherence (Sowa calls this *canonicality). A* relation node may have only one ingoing edge, but any number of outgoing edges. A concept node may have any number of edges, in or out. For the purposes of this paper, however, this family of graphs will be reduced to a *concept* graph by eliminating all relation nodes (thus reducing the number of colors to one) and the directionality of the edges. This can clearly be done for any conceptual graph (there is a small problem with relations of degree greater than two, but this need not concern us here. Figure 1. shows a conceptual graph and its reduction to the corresponding concept graph. The reason why we can work with the simplified graph is that specialize and its dual, frag*ment* only work on the concept nodes of the conceptual graph, through the operation *join.* The relational nodes, since they do not take part in join, can effectively be omitted. This is in contrast to the logic-based methods which almost always match predicates (relations) first, and leave object matching to some form of unification.

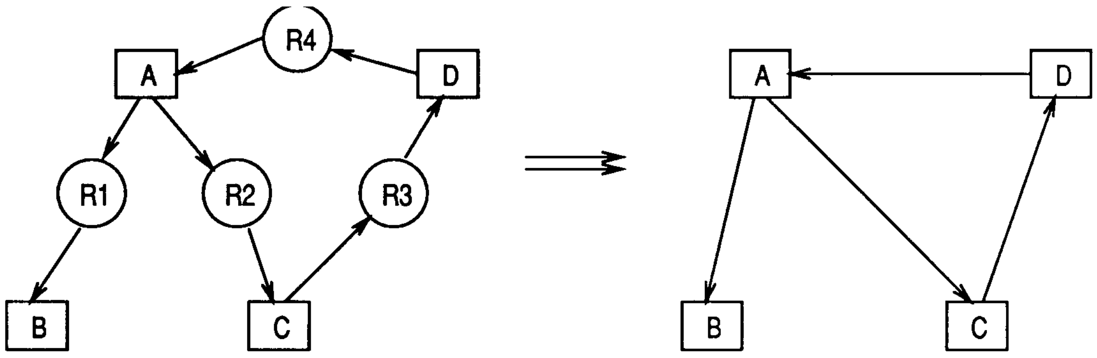

Figure. 1. A conceptual graph and its concept graph reduction

The functionality of specialize is:

$$specialize: 2^F \times 2^D \rightarrow 2^H$$

where F is a set of input graphs, D is a set of *definitions* (conforming to the rule types *2,* 3 and 4 above, i.e. Aristotelian, schematic, or causal) and H is the resultant set of hypotheses produced by cover and join. It is the composition of the two operations join and cover:

$$specialize = join \circ cover$$

2.1. The operator cover

It is the job of cover to choose an appropriate subset of a set of stored graphs.&, that cover all of the concepts in a given subset of graphs taken from a set 9. If the conceptual content of a graph ?l is given by C(g) and the maximal common subtype of two concepts c I and C 2 is given by *Mb (c* 1,c 2) then the functionality of cover is given by:

$$\textit{cover:}\ \mathrm{F} \times 2^{\mathrm{D}} \rightarrow 2^{\mathrm{D}}$$

$$\textit{where for } \mathrm{f} \in \mathrm{F}, \forall \mathrm{c} \in \mathrm{C(f)}\ \exists \mathrm{c_d}, \mathrm{d} \mid \mathrm{c_d} \in \mathrm{C(d)} \textit{ for some } \mathrm{d} \in \mathrm{D}$$

$$\textit{and } \mathrm{Mb(c, c_d)} \textit{ exists}$$

In other words, every concept in f must have at least one concept in the set of graphs D_C, where their maximum common subtype exists i.e. is not bottom. There are problems with graphs containing duplicate labels, but these can be solved by ensuring that there are sufficient quantities of covering concepts from graphs in D for the concepts in f.

The choice of an appropriate subset, since there can be many which satisfy the above condition is a matter for the pragmatics of the problem. The Maryland group (Nau and Reggia, 86) have used this idea of set covering (as have many others) in their diagnostic work, but deal with expressions at the propositional level rather than at the object level as we do here. They point out that although a parsimonious cover may be appropriate when simplicity is called for (cf. Occam's razor) there may be cases when less than parsimonious cover is safer, or simply better as an explanation.

Parsimonious cover may be produced my minimizing the boolean expression:

$$\wedge_c \vee_i d_i, \textit{ where } c \in C(f), C(d_i)$$

As a simple example:

$$\textit{let } C(f) = \{a, b, c\}$$
$$C(d_1) = \{a, b\}$$
$$C(d_2) = \{c, d, e\}$$
$$C(d_3) = \{a, c, d\}$$

Parsimonious cover is then the minimization of.

$$(d_l + d_3)\cdot(d_1)\cdot(d_2 + d_3)$$

$$\textit{or, } d_1 d_2 + d_1\, d_3, \text{ eliminating } d_1 d_2 d_3$$

2.2. The operator join

Cover just produces an appropriate subset of D. The job of producing an explanatory hypothesis is left to the binary operation join (actually *maximal* join). As an operation on single concept nodes, join merges two graphs at a single point where both graphs contain the same concept label. *Maximal* join (we will usually refer to this as just *join)* will not only allow restrictions in that a concept label can be replaced by a label of any subtype but also will merge the two graphs on the maximum number of nodes (see Sowa, op cit). An example is given in Figure 2. The functionality of join is:

$$\text{maximal join: } G \times G \rightarrow 2^G$$

There can be more than one maximal join, hence the powerset notation on the set of all graphs G. Join is a binary operation but multiple graphs can be joined by composing it with itself. Unfortunately, there is good reason to believe that join is not commutative when semantic considerations come into play (Pfeiffer and Hartley, 89), but for now we will assume there is no problem.

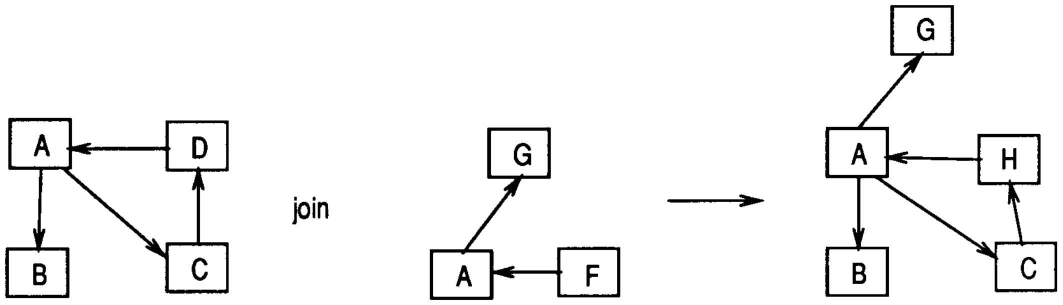

Figure 2. An example of maximal join (M_b (D,F) = H*).*

Since restrictions are allowed, it is clear that two nodes are joinable as part of a maximal join operation if they contain types that have a maximal common subtype. So PET can be restricted to DOG, and so can MAMMAL. Thus nodes containing PET and MAMMAL join to produce DOG. If two concepts have only $\perp$(bottom) as their common subtype, then the maximal common subtype is not considered to exist. The reason why the same constraint was placed on the operator cover was so to ensure that the covers returned are maximally joinable i.e. that the fact graph f is joinable to the graphs that cover it. That this is an abductive inference in the logical sense may be seen from the following equivalent presentation:

$$\begin{array}{l} PET(a) \\ MAMMAL\ (a) \\ \forall x\ DOG\ (x) \rightarrow PET\ (x) \\ \underline{\forall x\ DOG\ (x) \rightarrow MAMMAL\ (x)} \\ \qquad\qquad DOG(a) \end{array}$$

If we now look at example 5 above, the facts might be represented as in Figure 3, and the covering graphs in Figure 4. These graphs add information that the logical representation leaves out, however these are mandated by the need to form canonical graphs. The equivalent in a logic would be a full intensional logic with type restrictions on the place-holders, but the graphs have not prejudiced the argument that the appropriate hypothesis is obtained by joining all four graphs together, as shown in Figure 5.

The major addition to the driving graph is the actor node in the diamond-ended box. In a extension to conceptual graphs (Hartley, forthcoming), these nodes can express the temporal relationships between states and events in order to represent procedures qualitatively. In the example, the causal relationship between the two states CAR→LOC→PLACE and PERSON→LOC→PLACE is made through the actor, with the DRIVE event as mediator. The temporal relations between the state relations (the LOC nodes) and the event concept (DRIVE) designate a particular relation between the interval over which the event takes place, and the interval over which the state holds. CEFC (Continuous Emabling Finish Condition) and AS (Assert State) together mean that the end-point of the event interval coincides with the start of the state interval, which is open-ended. In other words, the state comes into existence when the event stops. However, these extra nodes play no part in cover or join, and will not be discussed further. The join will only occur if the following type relationships hold:

$$M_b(CAR, PHYS\text{ -}OBJ) = CAR$$
$$M_b(CAR, TRANSPORT) = CAR$$

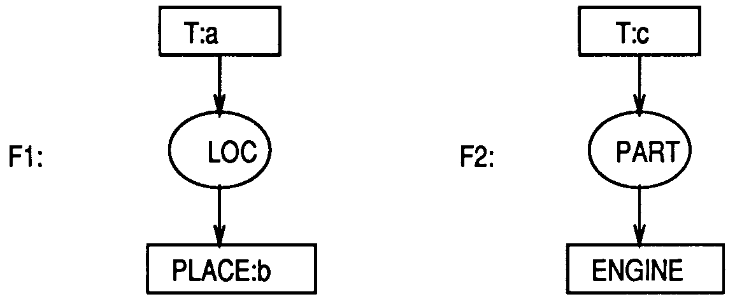

Figure 3. The ’driving’ facts.

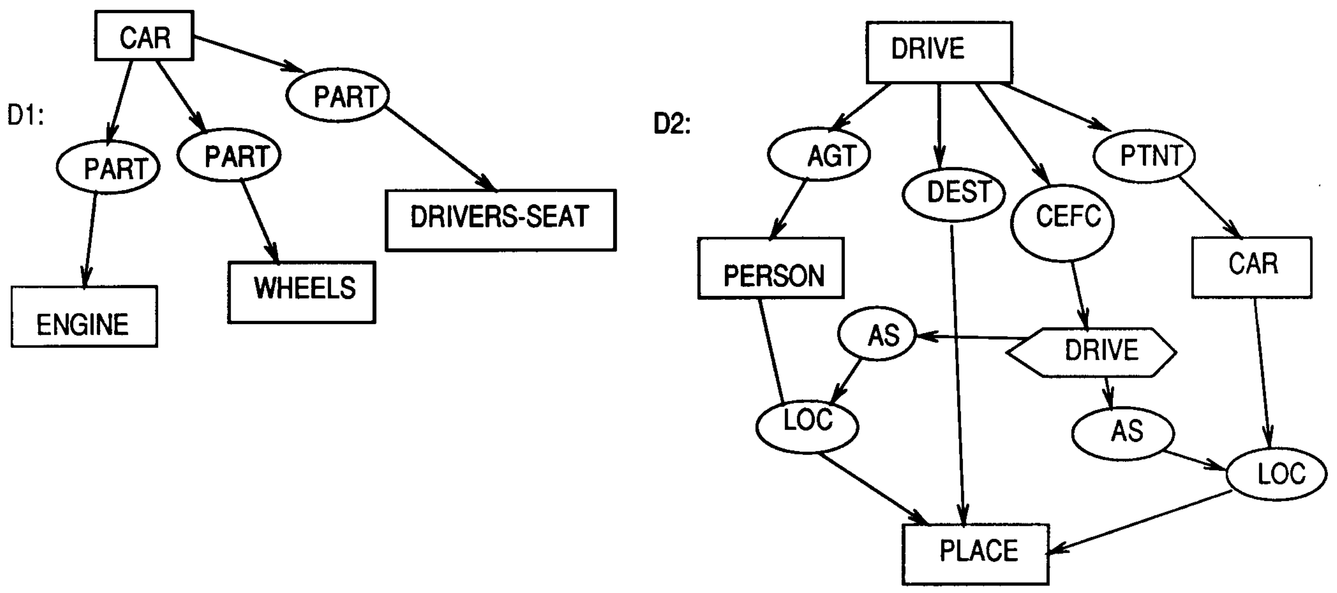

Figure 4. The covering graphs for Fig. 3.

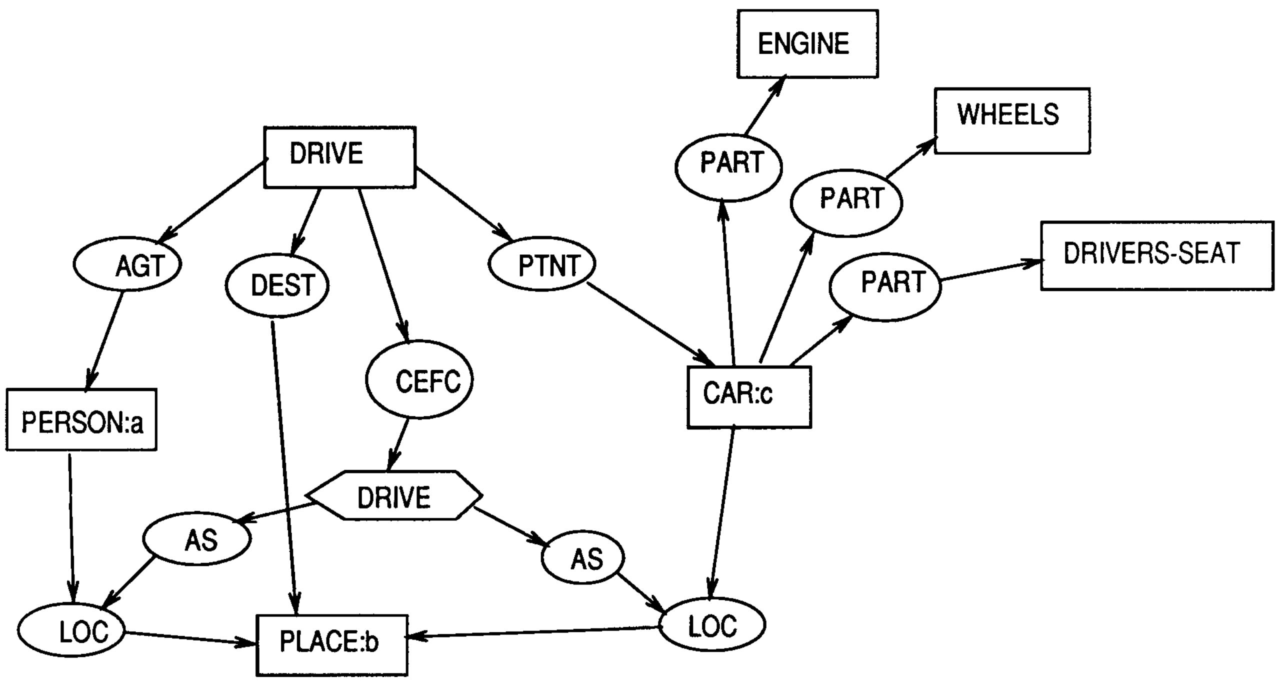

Figure 5. The maximal join of Figs. 3 and 4.

It should be noted that a person, an engine and a drivers-seat are all physical-objects, in addition to the car. These relationships potentially give alternative joins. Thus instead of placing the person a at the place *b*, the join could place any of the other objects, for instance the engine, at *b*. This sort of thing may produce a violation of canonicality (e.g. giving a pipe three ends instead of two ends and a middle), but may also be prevented by knowledge of the conformity of individuals to types. Again, the FOPC form does not contain this information, but a may conform to PERSON, but not to ENGINE.

In essence, therefore, the resultant graphs produced by join can be seen as abductive inferences from the facts and those definitions, causal or Aristotelian that cover them. The result is hypothetical in nature because the maximal common subtype restriction of two types leads to the same unsound inference rule that an logical abductive rule makes. Additionally, however, constraints stemming from canonicality and conformity increase the likelihood of the inference. Figure 6 contains a more intuitive Venn-like diagram of specialize where each enclosed region contains at least one concept node. *F1* and *F2* are two fact graphs, and *DI* and *D2* are two covering definitions. *D I* covers all of *F1* and one node of *F2, D2* covers all of F2 and two nodes of *F1.* Together they cover all of the nodes in both facts.

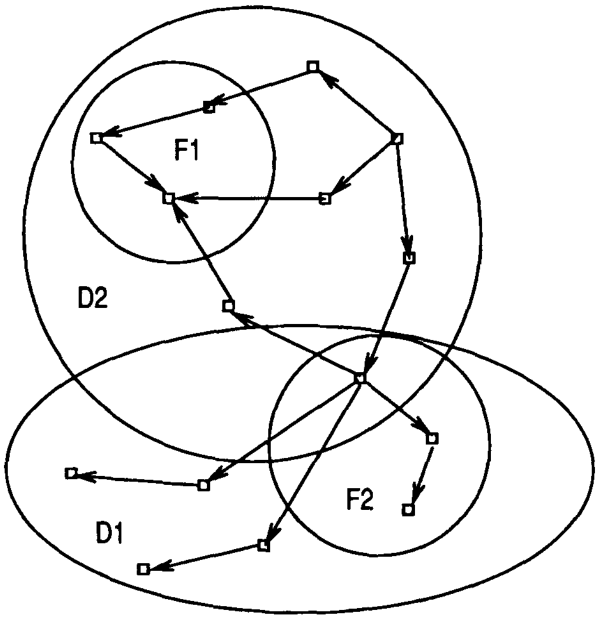

Figure 6. Specialize in a set diagram form.

3. An operator for deduction: fragment

In section 4 we will show how problem solving can be seen as a cycle of abduction and deduction. Any operator used for deduction had better be logically sound (although human problem solvers use a variety of unsound inferences, typically based on analogy). The inverse of the operation join is *project,* based on the same idea of merging two graphs, but this time taking the minimal common supertype of the concepts *(M_p).* Just as join is maximal, so project is as well. However, we will always mean maximal projection unless explicitly stated. The other difference is that all nodes that do not have a common supertype are dropped for the resultant graph. Again just as bottom was not allowed with join, so top is not allowed with project. If join is likened to set union, in that all nodes not joinable are just left alone, and come along for the ride, then project is like set intersection. All nodes that are not projectable are simply dropped from the resultant graph, along with their associated relation nodes.

All the abductive inferences in section I can be **reversed** into deductive inferences if the appropriate axioms are available. The conclusions are, of course, truth preserving. Project, however, is not a substitute for forward chaining through a rule. For instance, given the theory:

$$\forall x \; mammal\,(x) \rightarrow warm\text{-}blooded\,(x)$$
$$\forall x \; cat\,(x) \rightarrow mammal\,(x)$$
$$cat\,(a) \wedge has\text{-}tail\,(a)$$

we could conclude *warm-blooded (a).* However, if the schema definition for warmblooded mammals is projected with the fact about *a*, the result is the single node graph *mammal (a),* as Figure 7 shows; the other nodes are dropped out.

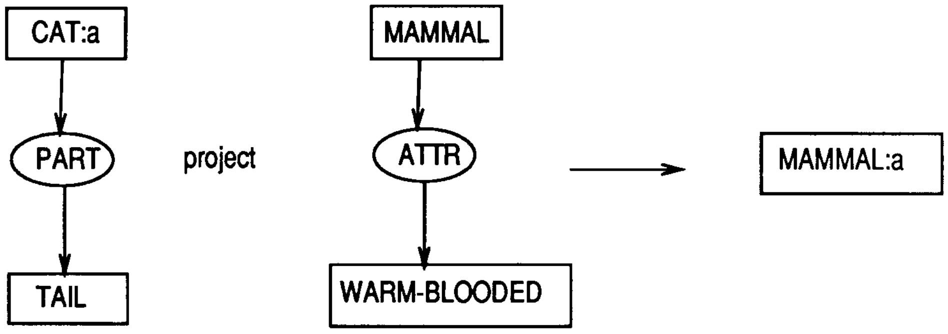

Figure 7. An example of projection

This is a valid conclusion, but it merely follows the subtype rule about all cats being mammals; it does not follow the schematic rule that all mammals are warm-blooded.

However, with the addition of an inverse operation to cover, in specialize, project can be turned into a useful operation. The inverse, which we call *uncover,* has the job of preserving factual information at the expense of definitional information. It breaks apart a hypothesis graph into unconnected pieces, or fragments which are by definition unjoinable. Hence the name for the operation: *fragment.* If specialize glues facts together with the minimum number of definitions, then fragment removes the minimum amount of glue to separate them again. It is important to note that fragment is not the true inverse of specialize. As we shall see in section 4, a problem solver searches a space of potential solutions. An operator that merely retraces the steps of another is useless. The functionality is:

$$fragment\!: 2^F \times H \rightarrow 2^H$$

Here a set of fact graphs taken from F are projected into a single hypothesis $h \in H$ to produce a set of fragments H. It is not necessary for the subset F to be the set of facts that were originally covered to produce h.

3.1. The operator project

Project's functionality is the same as join:

$$\textit{maximal project: } G \times \mathrm{G} \rightarrow 2^{\mathrm{G}}$$

The example in Figure 8 shows projection with the reduced form of graphs. These operand graphs are the same as in Figure 2.

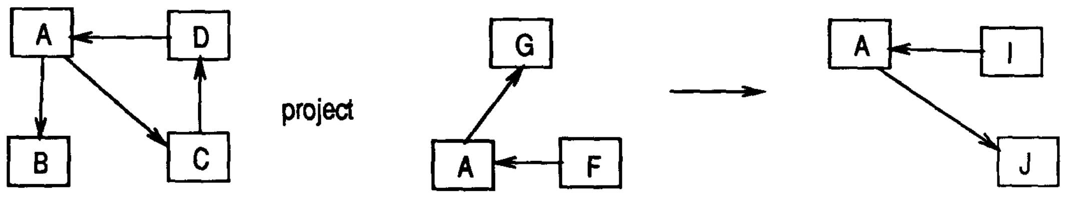

Figure 8. An example of projection
$(M_p\ (D,F)$ = I and $M_p\ (G,C) = J)$

3.2. The operator uncover.

The job of uncover is to undo some of the work that cover has done, but not all of it. It is best seen first in the set diagram form. In Figure 9 the regions labeled F1 and F2 are the regions of the graph that contain concept nodes to be projected onto by a given set of fact graphs. *D1, D2,* and *D3* are the definition graphs that originally made up the hypothesis graph. In fact these may already be fragments of definitions from previous fragmentations, but for the purpose of illustration, we will assume that the definitions are, as yet, intact within the hypothesis. Removing the links shown with a bar through them, and the node n1, splits the graph into two fragments. Which nodes to remove is a matter of searching out from the fact projections F1 and F2 (each fragment is thus guaranteed to contain at least one whole fact projection), adding nodes to each projection where they do not lie on a path to another projection (i.e. that connect *within* definitions), and removing one node that lies in the middle of each path to another projection. It can be seen from this example that uncover cuts links and removes information from definition graphs. Another way to say this is that facts are trusted, but definitions may have parts that are suspect. Fragment, through uncover, removes those untrustworthy parts. The fragments are then facts with good intensional information added (they could be called assumptions) and are available for further refinement through specialize. Moreover, the set of fragments produced by the operator are guaranteed not to join. They therefore correspond to separate hypotheses, and potentially to separate lines of thinking.

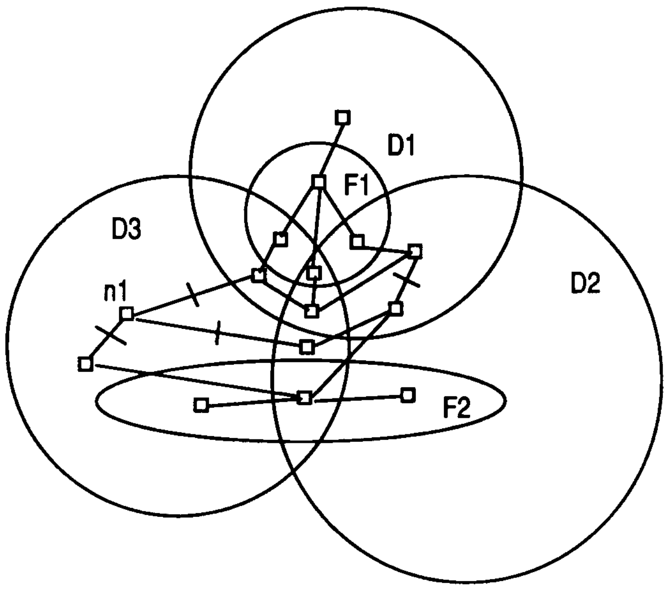

Figure 9 A set diagram showing uncover.

The functionality of uncover is:

$$\textit{uncover: } H \times 2^F \rightarrow 2^H$$

where $\forall h \in H_{out}$, h is connected, and no concept in any h connects with any concept in another h. and every h is of maximal size (i.e. $|C(h)|$ is maximal.)

3.3. An example of fragment producing new definitions.

The example shown above is a good case where the fragments produced are of different sorts. The upper fragment only contains nodes from one definition and one fact. However, the lower one contains nodes from two definitions and one fact. If these fragments were to be available as definitions for use in specialize, then the lower fact could be covered by one definitions, instead of two as before. Furthermore, the number of nodes has been reduced, specifically the glue removed by fragment. Fragment can thus be seen as a way of reorganizing knowledge deductively such that future abductions can be made more efficiently. This seems to a different sort of learning from explanationbased methods which typically are generalizations of successful solutions (de Jong and Mooney, 86).

Figure 10 shows, in concept reduction form, an instantiation of the parts in the example in Figure 9. The seemingly unconnected facts can be glued together by the three murder schema definitions. DRIVE is covered by DI, PLACE and CAR are covered by D1, D2 or D3 and MURDER, VIOLENT and 1ST-DEGREE by D2 *and* D3. All three definitions are thus needed to cover both facts. The five graphs join (there are no restrictions to sub-types to do) to produce Figure 11 which is the hypothesis that matches the form of Figure 9. Fragment removes the node PERSON (connected to PLACE), when used with the original facts, leaving the fragments shown in Figure 12. Here we used the original facts for fragment, to show that it is not the true inverse of specialize, although any facts which project into the hypothesis graph could have been used.

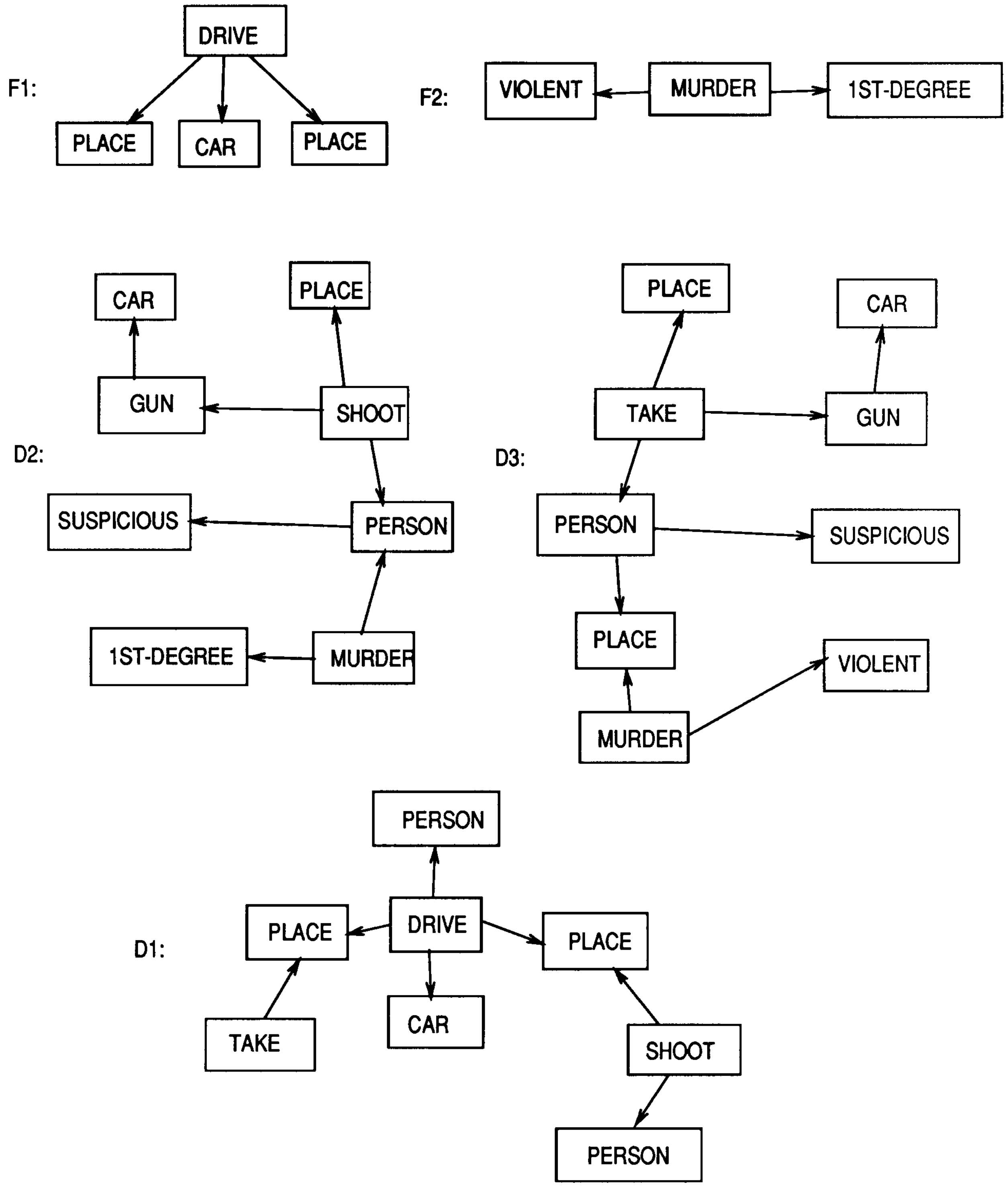

Figure 10. The ‘violent murder’ graphs.

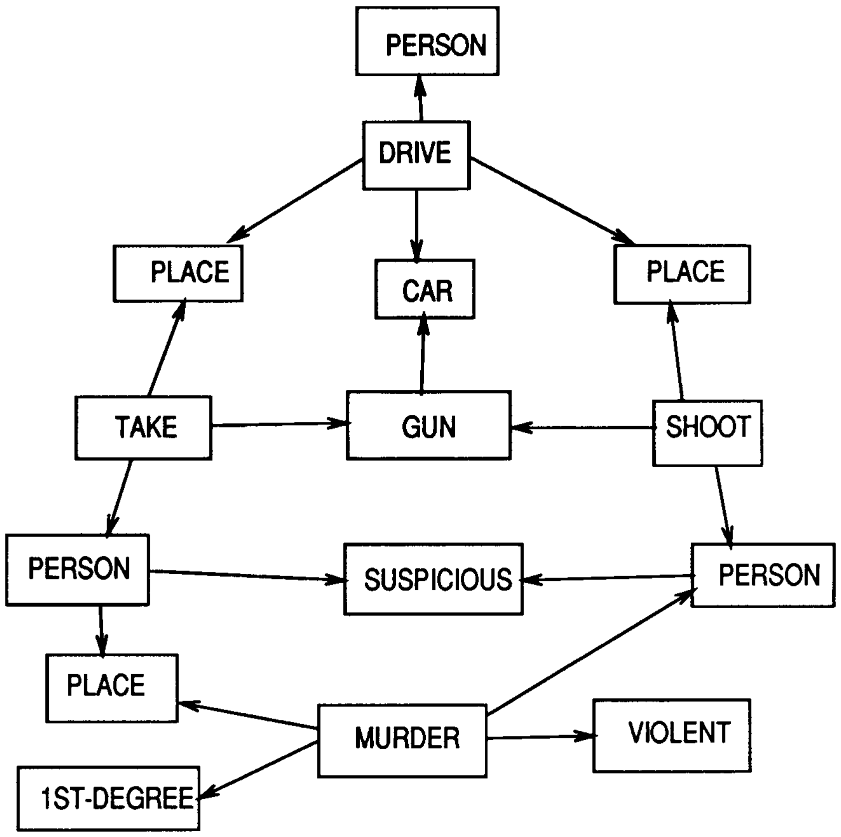

Figure 11. The 'violent murder' hypothesis.

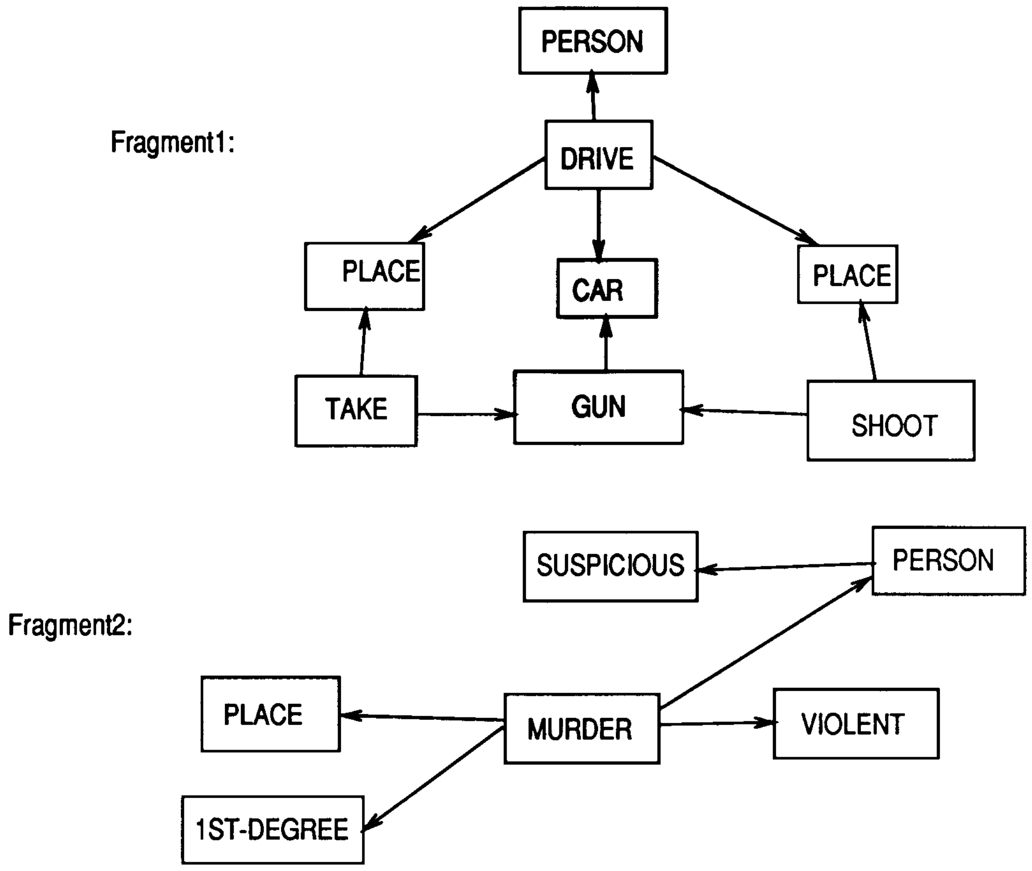

Figure 12. The 'violent murder' fragments.

4. The problem solving cycle

The operators specialize and fragment can be seen as allowing search in a space of hypotheses. Since specialize is based on join and fragment on project, it is possible to consider the space as a generalization lattice, as in Figure 13. The relationships between the graphs in the lattice are specialize and generalize. A graph is more specialized than another if it restricts a node to a subtype, or adds a node. A graph is more generalized if it replaces a node with a supertype, or removes nodes. Even though the graph that is just a single node with label T can never be produced, it anchors the lattice at the top end. Similarly there is one single-node graph at the bottom of lattice which is the (impossible) maximal join of all known graphs. If there exists at least one graph in the lattice that is a 'correct' hypothesis, then the problem solver must drive the system into that area of the lattice. The subject of control is too lengthy for this paper, but abstractly the two operators act in a refinement cycle, together with a decision operator *evaluate* that stops the cycle when a satisfactory hypothesis is generated. Figure 14 shows a data-flow diagram where control is embodied in an operator *choose* that filters the three data-bases of facts, definitions and hypotheses for specialize and fragment. The data-type graph flows along all the arcs in the diagram.

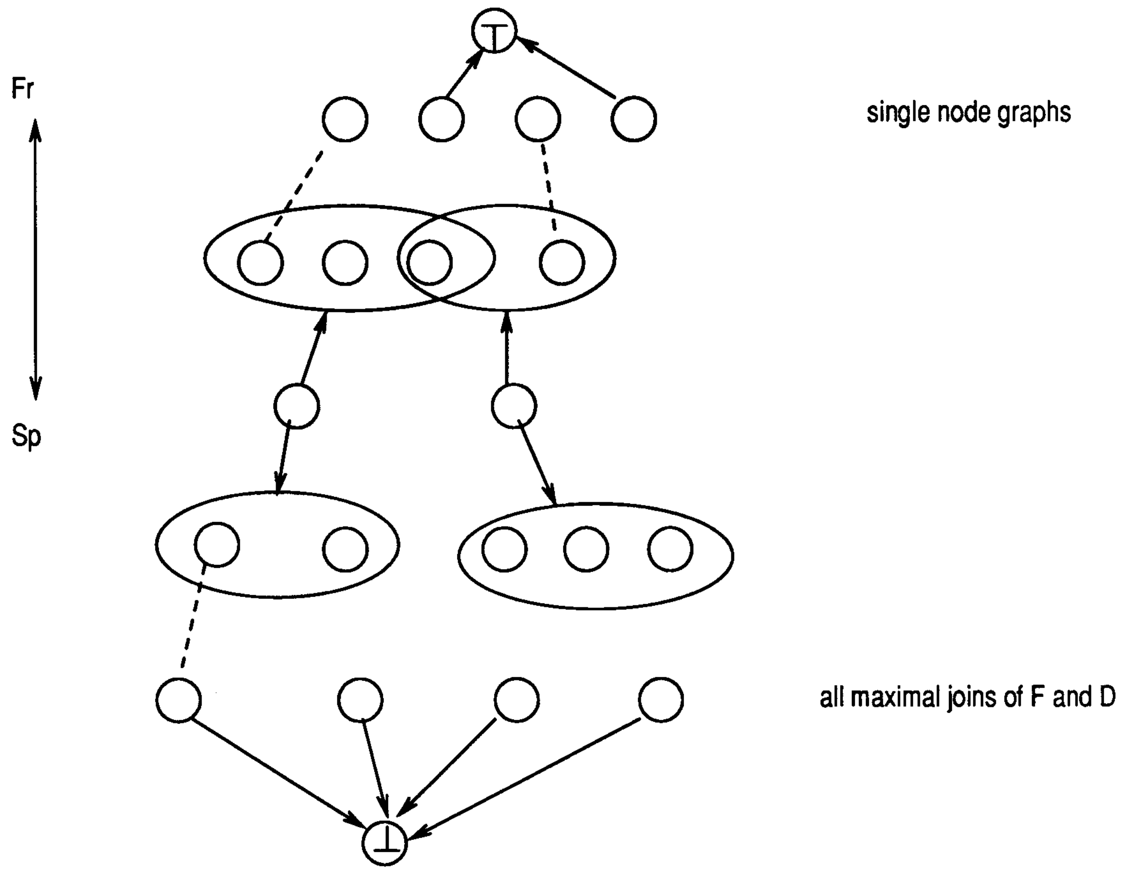

Figure 13. The specialize/fragment lattice.

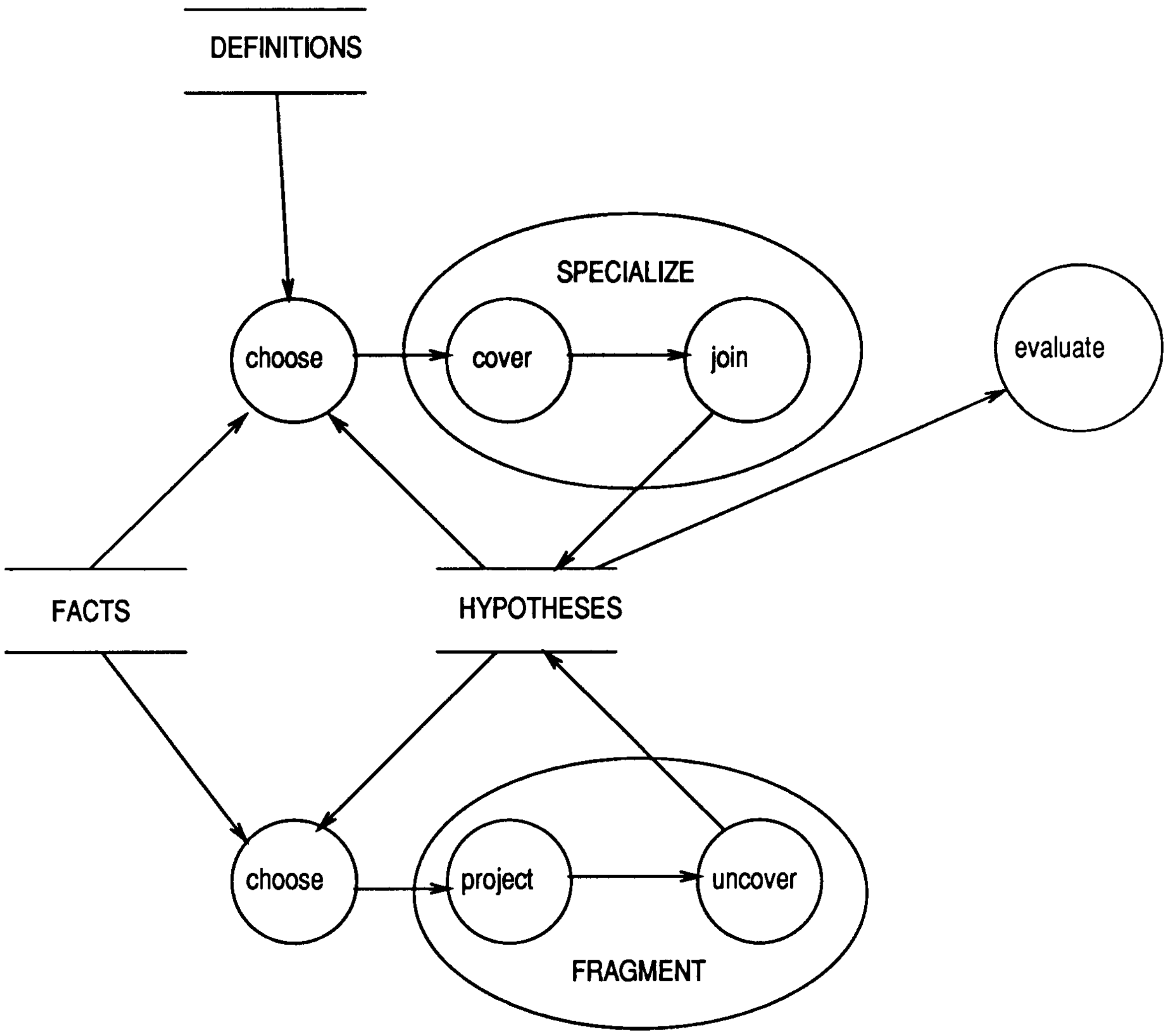

Figure 14. A data-flow diagram showing the relation of specialize and fragment.

Specialize produces alternative hypotheses for evaluation. When allowed, fragment breaks these unsatisfactory hypotheses apart, preserving factual connections, and allowing a new line of reasoning to emerge. The fragmented hypotheses become new input 'facts' for specialize. They contain a fact as a subgraph, but also contain pieces of definitional, and therefore intensional, information. Another way to view this is to call them assumptions, grounded in fact, but deduced by fragment from a hypothesis. There are degenerate cases where fragment merely inverts the effect of specialize. In this case the system will loop. However, if the stored definitional knowledge is rich enough, this will not happen. Cover, being associational in nature, win explore the interconnections in the knowledge base in a classical semantic network fashion. A useful conjecture is that specialize and fragment, in tandem, can generate any canonical graph, and to discover the conditions under which this might be true, but this has yet to be proved. (We have proved it for specialize and another, simpler operator *generalize* (Fields et al, 88) et al, 88), but not for fragment).
The cycle can also be seen as a weak problem-solving method, akin to the general techniques in SOAR (Laird et al, 87). Strength comes from good evaluation of hypotheses, particularly their rejection. We are also investigating the possibilities of controlling the search with some dynamic or even stochastic method, as in the genetic algorithm (Goldberg, 88). This is a view of problem solving that differs from the classical method based on a single line of reasoning, or even from the newer approach using multiple lines of reasoning such as found in the ATMS, wherein the system provides the problem solver with consistent alternative hypotheses but makes no attempt to evaluate their worth. Now we might view problem solving as refinement of a population of hypotheses where the evaluation has two components. One is a static measure of worth; the other is a dynamic measure of the population as a whole. These measures can be heuristically combined and fed back to the problem solver in order to guide the choice of an operator and on what it should operate. See (Fields et al, 88) for more detail.

5. Conclusion

We have presented two operators, specialize and fragment that when taken together mimic the abduction/deduction cycle of scientific method. The addition of an evaluation operator allows the cycle to be used a weak method for problem solving. Parsimony in the sub-operators cover and uncover ensure that potential solutions are as small as possible, and where a hypothesis is shown by evaluation to be inadequate, are modified as little as possible. Factual information is trusted over stored knowledge, allowing the breaking of associations in definitional knowledge to effectively form new, better definitions. The operator specialize has been shown to perform abduction, or hypothesis generation, and fragment to perform a limited form of deduction i.e. that necessary for hypothesis refinement.

6. References

Chapman, D. (19870 Planning for Conjunctive Goals, *Artificial Intelligence (32) pp.* 333-337.

DeJong, G., and R. Mooney (1986) Explanation-based learning: An alternative view. *Machine Learning* (1) pp. 145-176.

Fields, C. A., M. J. Coombs, E. S. Dietrich, and R. T. Hartley (1988b) Incorporating dynamic control into the Model Generative Reasoning system. *Proc. ECAI-88, pp.* 439-441.

Fields, C.A., Coombs, M.J. and Hartley, R.T. (1988) The MGR Architecture is Turing Equivalent. KSG working paper, Computing Research Laboratory, Las Cruces, NM.

Goldberg, D., *Genetic Algorithms in search, optimization, and machine learning.* Reading, MA: Addison-Wesley.

Hartley, R. T. and M. J. Coombs (1988) Conceptual programming: Foundations of problem solving. In: J. Sowa, N. Foo, and P. Rao (Eds) *Conceptual Graphs for Knowledge Systems.* Reading, MA: Addison-Wesley (in press).

Laird, J. E., A. Newell, and P. S. Rosenbloom (1987) SOAR**:** An architecture for general intelligence. *Artificial Intelligence* (33), pp. 1-64.

Levesque, H. (1989) A knowledge-level account of abduction. *Proc. Eleventh International Joint Conference on Artificial Intelligence,* Detroit, MI. pp. 1061-1073.

Nau, D. and J. Reggia (1986) Relationships between deductive and abductive inference in knowledge-based diagnostic problem solving, in L. Kerschberg (Ed.), *Expert Database Systems: Proceedings of the First International Workshop.* New York: Benjamin Cummings.

Peirce, C.S. (1957). *Essays in the Philosophy of Science.* New York: Bobbs-Merrill.

Pfeiffer, H.D. and Hartley, R.T. (1989) Semantic additions to conceptual programming. Conceptual Graphs Workshop, IJCAI89.

Poole, D. (1989) Explanation and prediction: an architecture for default and abductive reasoning. *Computational Intelligence* (5), pp. 97-110.

Reiter, R. and J. de Kleer. Foundations of assumption-based truth maintenance systems: Preliminary report. *Proc. AAAI-87.* Los Altos, CA: Kaufmann, 183-188 (1987).

Sowa, J.F. (1984). *Conceptual Structures.* Reading, MA: Addison Wesley.