e-MGR: An Architecture for Symbolic Plasticity

M. J. Coombs, H. D. Pfeiffer and R. T. Hartley

Computing Research Laboratory
Box 30001/3CRL
New Mexico State University
Las Cruces, New Mexico 88003-0001

[mcoombs,hdp,rth]@nmsu.edu

ABSTRACT

The e-MGR architecture was designed for symbolic problem solving in task environments where data are noisy and problems are ill-defined. e-MGR is an operator-based, shared memory system which integrates problem solving ideas from symbolic artificial intelligence (AI) and adaptive systems research.

1. Introduction

A widely reported result concerning knowledge-based, symbolic problem solvers in artificial intelligence (AI) is that they are brittle in the sense that they only respond as intended in narrow, well-specified domains (Coombs and Alty, 1984; Fields and Dietrich, 1987; Hewett, 1985; Holland, 1986). This is a serious weakness because systems tend to become more brittle as they become larger. Few prototypes, therefore, survive the process of being upgraded to a full-scale, real-world problem solver. Moreover, the brittleness problem has been experienced in many of the foundation domains for AI research, including: (i) real-time diagnosis of complex, quasi-open systems (Coombs and Hartley, 1987; Woods, 1986); (ii) military data fusion and information integration (Thompson *et al.,* 1986); (iii) reactive planning (Georgeff and Lansky, 1987); (iv) scientific data analysis (McWilliams *et al.*, 1989).

Despite the importance of solving the brittleness problem, there has been little systematic exploration of the causes of brittleness in symbolic systems. There has also been little work on alternative problem solving approaches, other then simulated neural networks, for tasks currently undertaken by symbolic systems. However, while neural networks may provide an effective solution for tasks involving the interpretation of low level data, they appear not to be appropriate for traditional AI application domains; in particular, those domains that have proved susceptible to expert systems techniques when the problems are well structured (Barnden and Pollack, 1989).

Ongoing research in the Knowledge Systems Group at the Computing Research Laboratory (CRL) has identified the presence of noise and novelty in the problem solver's task environment as a primary source of brittleness (Coombs and Hartley, 1987; Fields and Dietrich, 1988). Stated informally, a task environment is considered to be noisy when the data available to the problem solver are inaccurate, fragmentary, incoherent, or unrelated to the task in hand. A task environment is considered to contain novelty when the problem solution requires the system to relate objects in ways unanticipated by the system designer. Similar characterization of the sources of brittleness have been given by Hewitt (1985) and Holland (1986).

Symbolic problem solvers are constructed following the standard practices of software engineering. The programmer codes into the system a fixed set of relationships between data and knowledge that have been determined by experiment, or experience, to be relevant to desired system goals. We term these *relevance* relations. Input is then screened to ensure that these relations are preserved. However, while this strategy will maintain the internal consistency of the system, its products will not present reliable interpretations of data when input moves outside the system's pre-specified operating range.

In noisy and novel task environments it is not possible in principle to specify a complete set of relevance relations in advance of a problem solving event. The main approach to this relevance problem in AI is to incorporate various relevance logics or nonmonotonic logics in the control structure of the problem solver, thus enabling it to change what information in its knowledge base is considered relevant for interpreting data input (see Doyle, 1979; the 1980 Special Edition of *Artificial Intelligence* on Nonmonotonic Reasoning; de Kleer, 1986). The trouble with such approaches is that the weakness is still there since definitions of relevance and irrelevance are pre-coded;

the system will still be unable to distinguish between unexpected but relevant data and expected but irrelevant or misleading data.

The conceptual alternative to incorporating explicit relevance relations in the program code is to leave them implicit in the control structure. Relevance relations can then be constructed by optimizing some fitness function between data and knowledge, where the patterns of input that are interpreted as data depend on the knowledge structures used for interpretation, and the knowledge structures available to be used for interpretation depend on the patterns perceived. Instead of exploiting pre-coded relevance relations in a "generate/evaluate" control cycle, where problem solving operations are determined by the evaluation stage, it is necessary to remove "evaluate" from the control cycle in order to achieve plasticity. Problem solving is driven *within* the generation stage through feedback from characteristics of the current set of proposed solutions to the operators that generate new solutions. Evaluation is thus reduced to a decision procedure for terminating generation. This approach is, of course, related to problem solving using artificial neural networks, and also classifier systems, where novel solutions are created by the superposition of previous solutions, rather than by the method of discrete conjunction largely used in traditional symbolic AI.

At CRL, we have been exploring architectures for implementing this view of symbolic problem solving under the Model Generative Reasoning (MGR) project. Our early experimentation (Coombs and Hartley, 1987; 1988; Fields *et al.*, 1988) convinced us that four features were required to achieve the requisite plasticity to cope with noisy and novel task environments. These features are supported by the evolutionary-MGR (e-MGR) architecture, and include:

(i) object-based knowledge representation based on the contextual definition of concepts;
(ii) primitive operators capable of creating transient knowledge structures that have the combined status of being interpretations of available data, and expectations used to determine the data that might become available for interpretation in the next cycle;
(iii) operators capable of coping with novelty by decomposing knowledge structures and recombining fragments to create new structures;
(iv) an optimizing approach to system control that uses coherence as the strongest criterion for the acceptability of an explanation.

2. Problem Solving in e-MGR

Problem solving in symbolic AI has conventionally been conceptualized as a search through a space of problem states (Nilsson, 1980). These states are defined over a representation language composed of concept and relation terms, where possible combinations of terms are restricted to allow the expression of propositions that are considered to be relevant to the task domain of interest, i.e., in an animal identification program, a "bird", but not an "insect", is allowed to have AS-PART a "beak". The search space thus becomes dimensioned by some set of explicit relations, which may themselves be subject to some higher-level dimensioning. This is achieved typically through the imposition of a partial ordering of terms (e.g., through an ISA relation), or by their organization as rules (e.g., expressing a CAUSE relation). These dimensions can then be exploited by the problem solver to control the search of the problem space.

Given the multi-dimensional nature of problem domains typically chosen by AI research, e.g., the planning and scientific data analysis problems mentioned above, the search space tends to grow exponentially. This results in theoretically possible solutions becoming computationally intractable. A principal concern in the design of AI symbolic problem solving architectures is therefore the incorporation of control procedures to restrict the search space to a manageable size.

The intractability problem arises from the high dimensionality of explicit relevance relations typically present within AI problem domains. As a result, AI problem solving architectures may be characterized in terms of the specific method used to reduce dimensionality and factor the search space into sub-spaces. These methods frequently exploit three classes of perceived property attributed to objects and events. These include: (i) classification by *types; (ii) causal* relations; (iii) *means-end* relations. Architectures based on types are usually the most efficient but require a stable, well-defined problem domain (e.g., the spectroscopy domain of the classic expert system DENDRAL - Buchanan and Feigenbaum, 1978). However, as it becomes less clear how to dimension the problem space, where one may rely on pre-defined types, propagation architectures using causal or means-end relations become necessary (e.g., the least-commitment approaches explored in MOLGEN for genetics experiment design Stefik, 1981).

In noisy and novel task domains the dimensionality of relevance relations will be unknown, and thus unavailable to be built into a problem solver's control structure. EMGR attacks this problem by taking a procedural approach to relevance, allowing relevance relations to emerge implicitly from the optimization of a fitness function between data and knowledge, where the basic function is defined in terms of structural mappings between concepts rather then relations. The semantic relations necessarily present in the representation language to allow for the expression of domain knowledge are not the fundamental source of control, as in conventional AI architectures. However, it may be necessary in some applications to incorporate measures derived from semantic features into the fitness function.

Reasoning in e-MGR is implemented through a process of building sets of hypothetical conceptual structures to explain the concepts in available data. Since all objects in e-MGR are represented as graphs, it is possible to define "explanation" in terms of set relations between concept nodes in the graphs representing data and concept nodes in the graphs representing knowledge; more specifically, in terms of the *set covering* relation between data concepts and knowledge concepts; in e-MGR predefined knowledge structures are termed *definitions,* data are termed *facts* and explanations are termed *models.*

In this respect, reasoning in e-MGR is related to the generalized set covering (GSC) view of abductive problem solving developed by Reggia *et al.* (1985) where, given data, the task is to find the best set of hypotheses to explain the data in terms of the most parsimonious cover of the data by this set. However, whereas GSC deals with atomic explanatory hypotheses and pre-defined relevance relations between hypotheses and data, the requirement that e-MGR should function in noisy and novel task environments makes it necessary for the system to be capable of: (i) creating hypotheses autonomously from knowledge fragments, and (ii) autonomously identifying relevant data from the set of available observations.

Explanations, e-MGR models, are generated through a set of graph transformation operations: *(i) specialize,* which builds new graphs from graph fragments, "gluing" together facts with definitional material to generate models; (ii) *fragment,* which decomposes graphs into fragments, "ungluing" models to extract fragments worth preserving as assumptions to be passed on to subsequent stages of processing; (iii) *classify*, which tags a graph with a pre-computed marker graph, using assumptions to tag new facts to be submitted for processing. The critical problem solving notions here are that: (i) e-MGR interprets facts by gluing them together with definitional material to form models; (ii) models are unglued to form assumptions (proto-facts); (iii) assumptions are used to extract new facts from the world, which then become interpreted to form new, more complete models.

To conclude this conceptual summary of e-MGR problem solving, it should be noted that the three operations can be interpreted in terms of Peirce's (Peirce, 1934) explanation cycle → *induction* → *abduction* → *deduction* →. Classify implements the induction of relevance relations between assumptions and facts, specialize implements the abduction of interpretive contexts for tagged facts, and fragment implements the deduction of new assumptions from models (Hartley and Coombs, 1989). That e-MGR explanations are truly abductive, and will contain information of a hypothetical nature (i.e., that is not contained in the facts), is evident from the operation of the primitive procedures *cover* and *uncover* that implement the gluing and ungluing of graphs. Cover interprets tagged facts by first finding some subset of definitions that subsume the facts, and then fusing facts and definitions by coalescing on common concepts. The resulting explanations will therefore contain facts joined by non-factual material. Uncover cleaves an explanation into one or more fragments around the images of facts projected onto it by removing links between projections. Links may not necessarily be cut exactly at projection boundaries, thus leaving nodes that originate from definitions attached to the fragments. Uncover is not simply the inverse of cover.

3. The e-MGR Architecture

3.1. Overview

Evolutionary (e-MGR) is logically a multi-instruction, multi-data parallel virtual machine (MIMD) that accepts input from the databases F,D, an M. F is a fact database that receives all input from external agents; D is a definition database that contains all of the system's pre-computed explanations, and serve as an initial set of concerning the relatedness of facts; M is a model database that contains all explanations currently under development. In contrast to earlier descriptions of the architecture (Fields *et al.,* 1988), in e-MGR we partition M into three sections: (i) A, which is an assumption database containing system constructions deemed to be sufficiently fit to regard as factual "for all practical purposes"; (ii) T, which is a database of tagged items selected from A and T to be presented to e-MGR for interpretation during the current cycle; (iii) M itself, which as in MGR is the database of current interpretations. All objects in the databases are represented as connected, multi-labeled, bipartite, oriented graphs. A data flow diagram of the e-MGR architecture is given in Figure 1.

Three operators, classify, (*Cl*), specialize, (*Sp*), and fragment, (*Fr)* act on the databases in an autonomous fashion. The functionality of these operators is specified completely

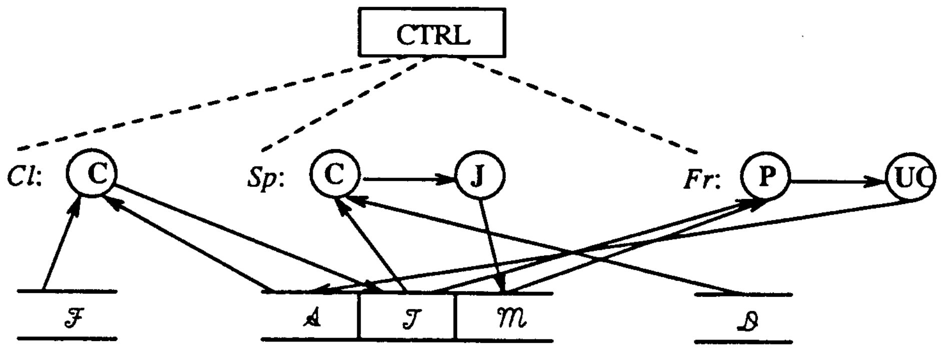

Figure 1. A data-flow diagram of the e-MGR architecture. Detailed description of the lower-level operators join, **J**, cover, **C**, project, **P** and uncover, **UC**, are given in Hartley and Coombs (1989) and in section 3.3. below. Informally, **C** identifies a subset of definition graphs that has some pre-defined set cover relation to all of the labeled nodes in a given subset of graphs; **J** merges two graphs at a single point where both graphs contain related node labels; **P** is the inverse of join in that it seeks to identify related labels between graphs; **UC** is the inverse of cover in that it partitions graphs in the neighborhood of sub-graph boundaries.

by the architecture. Operator actions may be described informally as follows: (i) *Cl* selects tagged facts T for interpretation from processing of A and F; (ii) *Sp* generates model M interpretations by fusing items from T using definitional "glue" taken from D; (iii) *Fr* generates new assumptions by cleaving models through the removal of "glue" around the items currently in T. The e-MGR operations can be represented as a closely coupled set of functions, with coupling at T, M, and A. In the worst case:

Classification

$$Cl: \mathcal{A} \times 2^{\mathcal{T}} \rightarrow 2^{\mathcal{T}}$$

Specialization

$$Sp: 2^{\mathcal{T}} \times 2^{\mathcal{D}} \rightarrow 2^{\mathcal{M}}$$

Fragmentation

$$Fr: \mathcal{M} \times 2^{\mathcal{T}} \rightarrow 2^{\mathcal{A}}$$

The activity of the operators is governed by the control level, which determines when the operators act, but not their functionality. Strategy in MGR thus consists largely of scheduling these three operators, along with the additional activities of selection over T and D, and evaluation of A and M in order to determine halting conditions. Control strategies are formally optimizations, represented either as algorithms or adaptive systems.

The precursor to e-MGR, along with the supporting CP environment to be discussed below, is implemented in CommonLisp on Symbolics Lisp Machines. e-MGR is currently being developed on the Sun 4 (Sparc) running SunOS 4.0.3. The e-MGR precursor has been used to produce a number of proof-of-concept demonstrations in the information integration area. The most substantial project is concerned with intentions analysis in high-intensity conflict (Coombs et al., 1990).

3.2. Representation

3.2.1. The Conceptual Programming (CP) Environment

The conceptual representation package for e-MGR is provided by the Conceptual Programming (CP) environment (Hartley and Coombs, 1988). This package uses knowledge structures based on Sowa's conceptual graphs (Sowa, 1984), and supports three levels of representation. Two classes of declarative concept are represented at the base level: *facts* and *schematic definitions.* The concept labels form a vocabulary which is partially ordered to form a type lattice.

Facts correspond to reports of objects in the world and their interrelationships; schemata correspond to definitions of objects and actions that operate on them. The schematic definition for the act [MURDER-METHOD] taken from a cluster of murder methods is illustrated in Figure 2. This schema shows the relationships between the action of an illegal killing. and the concepts related to the act. For example, an unknown [PERSON] is the agent of the [KILL] act, a [VICTIM] is the object of the act, and a [WEAPON] is the instrument. Schemata assert the set of possible structural features that may exist without reference to time or space, where relationships are not necessary but represent a point of view of the object. Each object can have more than one schema associated with it, in which case we refer to the schematic cluster of an object.

The second CP level represents procedural or temporal information. The procedural/temporal level is built on the declarative level and shows the temporal relationships between each action in a schema. Procedural overlays make use of a distinction between states and events. If a state is thought of as a collection of predicates, then events are relations on states. The procedural overlay for killing in Figure 2 is represented by the diamond-shaped node (actor) linked to relations by solid lines. Actors serve essentially as confluence nodes, taking states as inputs and events as outputs or, vice versa, inputs and states as outputs. In Figure 2, the act of <KILL>ing is shown to require the "Temporary Enabling Condition" (TEC) that both killer and victim be in the same place, and that <KILL>ing is a "Continuous Enabling Finish Condition" (CEFC), i.e., the body does not get up and walk away.

The third level of CP adds numerical constraints to graphs, which give power and flexibility to the representation of actions by the attachment of functions. In CP, these functions serve as more than just procedures in that they can serve as true constraints, i.e. a missing parameter to the function can be constrained to a value that is dependent on the other parameters. Each function has a number of parameters which may be specified as either inputs, outputs or both. If some of the parameters to the function are both inputs and outputs, the function serves as a constraint function. If all parameters are specified as inputs or outputs the function serves only as a computational procedure.

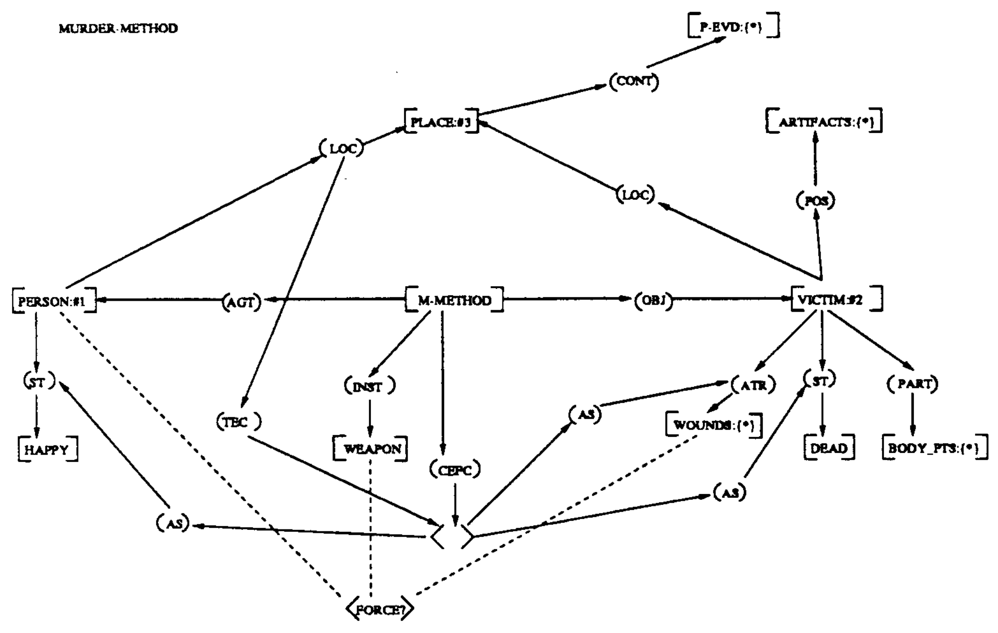

Figure 2. A CP schematic definition representing the concept of [MURDERMETHOD].

In Figure 2, the constraint overlay associated with the [MURDER-METHOD] schema is shown as a diamond-shaped node linked to its parameters - [PERSON], [WEAPON] and [WOUND] - with dashed lines. It asserts that the [PERSON] must have been capable of using the [WEAPON] to apply sufficient force to cause the victim's [WOUND].

3.2.2. Reduced Graph Notation for Operators

As mentioned in the overview, conceptual graphs are connected, directed, bipartite graphs where the nodes are labeled with either a concept type or a relation name, and concept labels are partially ordered to form a type lattice. There are restrictions on the edges, however, that are used to preserve semantic coherence. 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 and the directionality of the edges. This can clearly be done for any conceptual graph, although there is a small problem with relations of degree greater than two which will not concern us here. Figure 3 shows a conceptual graph and its reduction to the corresponding concept graph.

3.3. Primitive Operators in e-MGR

3.3.1. The Cover Operator C

The cover operator acts over two sets of graphs **A** and **B**. Its job to choose an appropriate subset of a set of graphs **B**, that cover all of the concepts in a given subset of graphs taken from a set **A**. If the conceptual content of a graph g is given by CC(g) and the maximal common subtype of two concepts c_1 and c_2 is given by $M_b(c_1, c_2)$ then the functionality of cover is given in the worst case by:

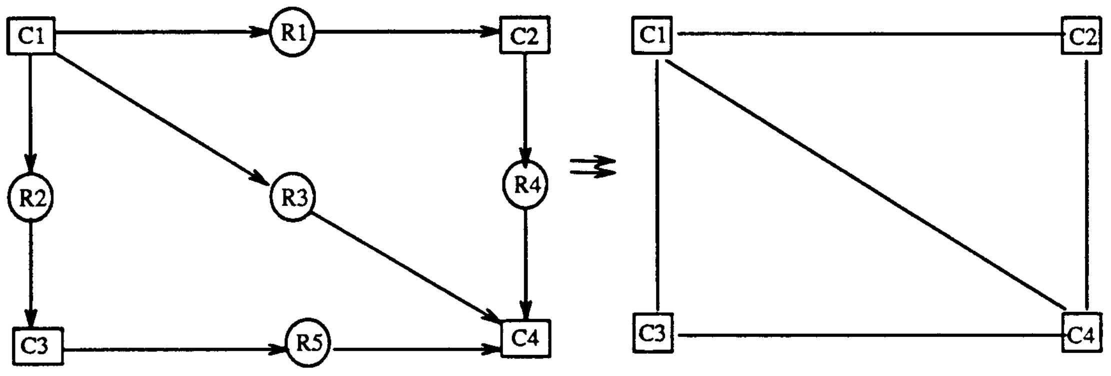

Figure 3. A conceptual graph and its concept graph reduction.

$$\mathbf{C} : \mathbf{A} \times 2^{\mathbf{B}} \rightarrow 2^{\mathbf{B}}$$

$$\textit{where } \text{for } \mathbf{a} \in \mathbf{A}, \forall c \in CC(\mathbf{a}) \ \exists c_{\mathbf{b}}, \mathbf{b} \mid c_{\mathbf{b}} \in CC(\mathbf{b}) \text{ for some } \mathbf{b} \in \mathbf{B}$$

$$\textit{and } M_b(c, c_{\mathbf{b}}) \text{ exists}$$

In other words, every concept in a must have at least one concept in the set of graphs **B**cc, 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 **B** for the concepts in **a**.

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, and the e-MGR operator it is serving. For illustration, we use the parsimonious set cover definition from Nau and Reggia (1986), which is produced my minimizing the boolean expression:

$$\bigwedge_c \ \bigvee_i \mathbf{b}_i, \text{ where } c \in CC(\mathbf{a}), CC(\mathbf{b}_i)$$

$$\textit{let } CC(\mathbf{a}) = \{l, m, n\}$$

$$CC(\mathbf{b}_1) = \{l, m\}$$

$$CC(\mathbf{b}_2) = \{n, o, q\}$$

$$CC(\mathbf{b}_3) = \{l, n, o\}$$

Parsimonious cover is then the minimization of:

$$(\mathbf{b}_1 + \mathbf{b}_3).(\mathbf{b}_1).(\mathbf{b}_2 + \mathbf{b}_3)$$

$$\textit{or}, \ \mathbf{b}_1\mathbf{b}_2 + \mathbf{b}_1\mathbf{b}_3, \quad \text{eliminating} \quad \mathbf{b}_1\mathbf{b}_2\mathbf{b}_3$$

3.3.2. The Uncover Operator UC

The job of uncover is to undo some of the work that cover has done, but not all of it. It is best illustrated with a set diagram form. In Figure 4 the areas labeled **AI** and **A2** (forming the set **A**) are distinguished regions of a graph B. Uncover splits apart these regions by breaking links between nodes spanning these regions to form multiple new distinguished regions. These regions may, and usually will, contain additional nodes derived from the spanning region.

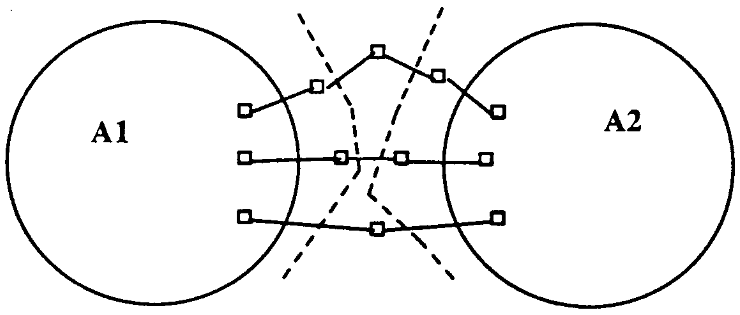

Figure 4. Set diagram of the uncover operator **UC**.

In Figure 4, uncover removing the links shown with a bar through them and splits the graph into two fragments. Which nodes to remove is a matter of searching out from the initial distinguished regions, adding nodes to each region where they do not lie on a path to another region, and removing one node that lies in the middle of each path to another region.

The functionality of uncover is:

$$\mathbf{UC}: \mathbf{B} \times 2^{\mathbf{A}} \rightarrow 2^{\mathbf{B}}$$

where $\forall$ b $\in$ $\mathbf{A}_{\text{out}}$, b is connected, and
no concept in any **b** connects with any concept in another **b.**

3.3.3. The Join Operator J

The job of the binary operation join is to merge two graphs at a single point where both graphs contain the same concept label, or a subtype. In the current version of the e-MGR architecture, join is always maximal, i.e., labels may be restricted by replacement with a label of any subtype, and graphs will be merged on the maximum number of nodes. However, there are plans to implement a partial join in order to support information integration and planning applications. An example of maximal join is given in Figure 5.

The functionality of maximal join over a set of graphs G is:

$$max\ \mathbf{J}: \mathbf{G} \times \mathbf{G} \rightarrow 2^{\mathbf{G}}$$

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, 1989), but for now we will assume there is no problem.

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 NARCOTIC can be restricted to OPIUM, and so can POISON. Thus nodes containing NARCOTIC and POISON join to produce OPIUM. If two concepts have only $\perp$(bottom) as their common subtype, then the maximal common subtype is not considered to exist.

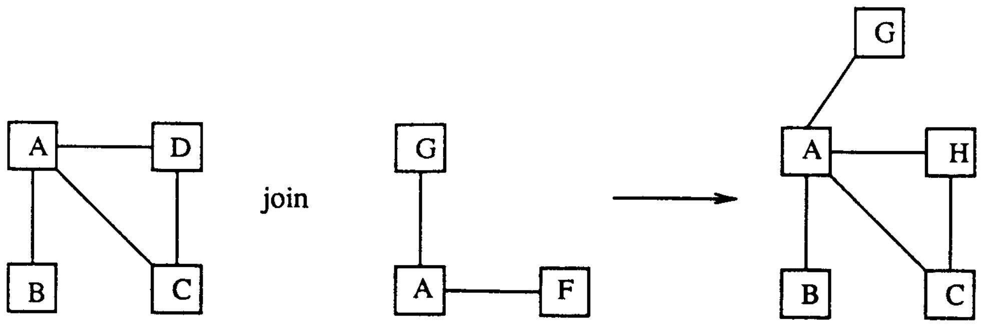

Figure 5. An example of maximal join (M *b (D,F) = H).*

$$\frac{\begin{array}{l} NARCOTIC(a) \\ POISON(a) \\ \forall x\ OPIUM(x) \rightarrow NARCOTIC(x) \\ \forall x\ OPIUM(x) \rightarrow POISON(x) \end{array}}{OPIUM(a)}$$

3.3.4. The Project Operator P

P is the inverse of the operation join. It is based on the same idea of merging two graphs, but this time taking the minimal common supertype of the concepts *(MP).* Just as join is maximal in the current e-MGR architecture, so is project. Another similarity is that, 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.

Project's functionality is the same as join:

$$max\ \mathbf{P}:\ \mathbf{G} \times \mathbf{G} \rightarrow 2^{\mathbf{G}}$$

The example in Figure 6. shows projection with the reduced form of graphs. The operand graphs are the same as in Figure 5.

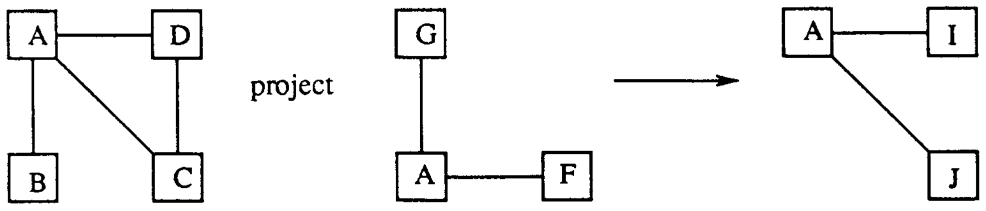

Figure 6. An example of projection (M p (D *F)* =I and M p (G,C) = J).

3.4. The e-MGR Operators

3.4.1. The Functionality of the Classify Operator Cl

Cl is implemented by the single primitive operator cover **C**. Its task is to retrieve the maximal set of facts from F that mention a concept present in the current set of assumptions A. *Cl* therefore uses a default of maximal cover, which is the opposite setting of cover to the example given in section 3.3.2. The set of facts retrieved, and the assumptions employed to retrieve them, are then passed along to *Sp* as the set of tagged facts T (or more correctly the set of assumption tags A and facts in T covered by graphs in A).

The functionality of classify is:

$$Cl\colon \mathcal{A} \times 2^{\mathcal{F}} \rightarrow 2^{\mathcal{T}}$$

Classification may be seen as a form of induction operation, in which assumptions are used to induce a tentative set of relevance relations over facts, and to abstract those facts that conform to those relevance relations.

3.4.2. The Functionality of Specialize Operator Sp

The functionality of specialize is:

$$Sp\colon 2^{\mathcal{T}} \times 2^{\mathcal{D}} \rightarrow 2^{\mathcal{M}}$$

where T is a set of tagged facts, D is a set of *definitions,* and M is the resultant set of models.

Sp is composed from the two primitive operators cover, **C**, and join, **J**, where

$$Sp = \mathbf{C} \cdot \mathbf{J}$$

In the *Sp* context, **C** selects a subset of stored graphs D that cover the concepts subsets of graphs from the set T. In applications implemented in e-MGR, a default setting of minimal cover for Sp has found to be effective (e.g., Coombs *et al.,* 1990). This contrasts with the default of maximal cover setting for *Cl*. Models are then generated by the primitive operation maximal join, **J**, which merges definitions and tagged facts on common labels. During this operation, all labels with common subtypes will be specialized to their maximal common subtype.

As a reasoning process, specialize generates contextual interpretations of tagged facts by covering them with stored definitions. Since in e-MGR the covering definitions are viewed as explanations that provide the system with expectations concerning relations between facts, and join restricts labels to their maximal common subtype, the resultant model graphs have the status in logic of abductive inferences, i.e., they should be regarded as hypotheses rather than sound deductive inferences.

Figure 7. contains a more intuitive Venn-like diagram of specialize where each enclosed region contains at least one concept node. *T1* and *T2* are two fact graphs, and *D1* and *D2* are two covering definitions. *D1* covers all of *T1* and one node of *T2*, *D2* covers all of *T2* and two nodes of *Fl*. Together they cover all of the nodes in both facts.

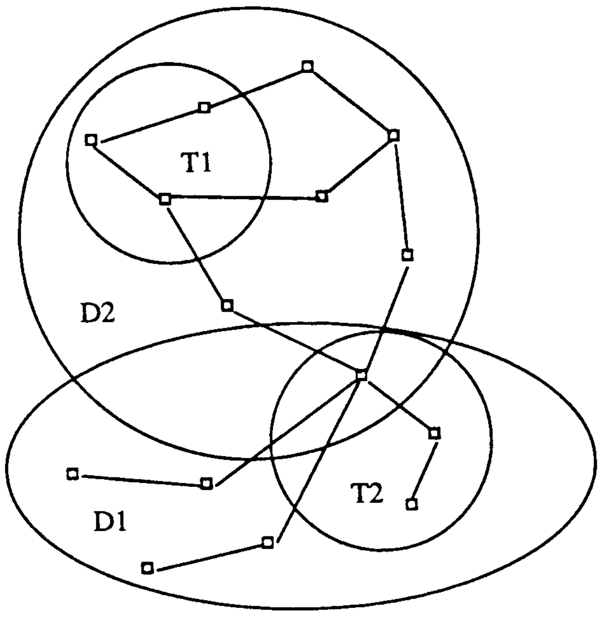

Figure 7. Specialize *Sp* in a set diagram form.

3.4.3. The Functionality of the Fragment Operator Fr

The functionality of fragment is:

$$Fr\colon \mathcal{M} \times 2^{\mathcal{T}} \rightarrow 2^{\mathcal{A}}$$

where a set of tagged fact graphs taken from T are projected into a single model $m \in M$ to produce a set of fragments A. The subset T need not be the set of tagged facts that were originally covered to produce m.

Fr is composed of the two primitive operators project, **P**, and uncover, **UC**. **P** identifies projections of tagged facts in a model, m. We then add to projection **UC**, the inverse of the cover operation. UC has the job of preserving subgraphs in m that are supported by facts at the expense of definitional information. It does this by breaking apart m into unconnected pieces, or fragments, which are by definition unjoinable. Hence the name for the operation: *fragment.*

Fr is used in two different contexts that require different setting for the amount of definitional glue removed by **UC**. From a reasoning perspective, *Fr* has two functions: (i) the extraction of assumptions from models, where the assumptions may be regarded as factual "for all practical purposes", and (ii) the removal of incoherences between a model and a tagged fact. In the case where *Fr* is used to extract assumptions, fragments may be seen as logical deductions from the facts in T. **UC** thus needs to remove sufficient glue from models to leave facts minimally connected with hypothetical material. The e-MGR architecture thus employs maximal uncover and the default for the deductive use of fragment. In contrast, the removal of incoherences from models may be seen more as a process of adjusting abductions, than of making deductions. E-MGR therefore uses the default **UC** of minimal uncover, which preserves the hypothetical nature of fragments.

Figure 8 illustrates minimal uncover in the form of a Venn-like diagram. The regions labeled *T1* and *T2* 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 model 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 model. 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 T1 and T2 (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.

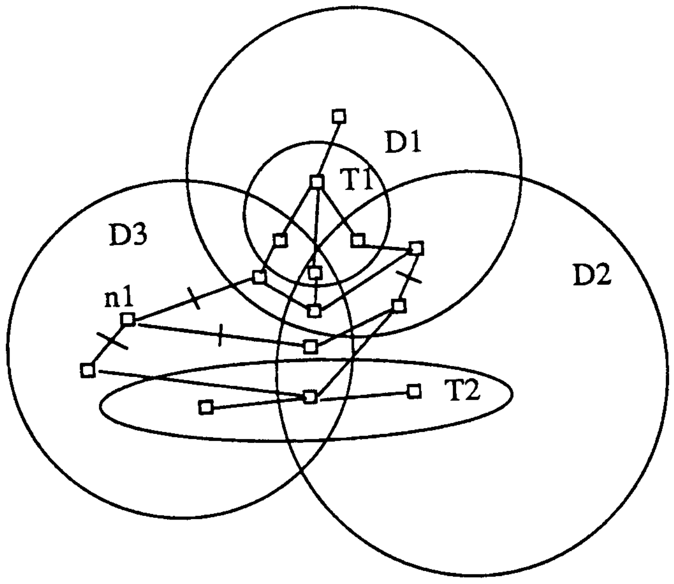

Figure 8. Fragment *Fr* in a set diagram.

4. Baseline Experiments in Control

4.1. The Approach

Control in e-MGR is the process of regulating the execution of the e-MGR operators. Operator execution may be regulated by: (i) adjusting the order and rate of firing, (ii) adjusting the setting of the primitive operator parameters, (iii) applying filters to a selected operator's input from the five databases F, D, A, T and M, and (iv) restricting operator output by evaluation.

As discussed in section 2, e-MGR supports an optimization approach to control. More specifically, e-MGR seeks to optimize the final model population using feedback from some function defined over a set of relations between assumptions and models. The function manipulates the gluing and ungluing of graphs through adjusting parameters to the primitive operators **C** and **UC**. The manipulation of these operators thus provides the basis for the controlling process. A set of experiments have been designed to baseline the effects of given parameters to C and UC within the e-MGR operator framework.

Control is being studied through a simulator written in C and running on a Sun 4. The simulator has all the essential functionality of the e-MGR architecture, although it sacrifices representational expressiveness with regard to both the concept language and the allowed structural relations between the transient structures A, T, and M In particular, the simulator imposes the following set of subsumptive relationships:

$$Cl: \mathcal{A} \subset \mathcal{T} \subset \mathcal{F}$$

$$Sp: \mathcal{T} \subset \mathcal{M} \subset \mathcal{D}$$

$$Fr: \mathcal{T} \subset \mathcal{A} \subset \mathcal{M}$$

Although the simulator is much reduced, it do have significant advantages with regard to speed and procedural clarity.

The halting rule used in the experiments is an e-MGR variant of the parsimony principle employed in set covering approaches to reasoning (Reggia *et al.*, 1985). The rule sets the goal of generating a set A, such that the set:

i) maximizes the facts covered, while

ii) minimizing the definitional material used in order to achieve maximal cover.

This system goal is open to a number of operational interpretations, depending on the operational definitions of "definitional material". The initial definition employed in the precursor MGR system was made in terms of the cardinality of discrete definitions used to achieve the cover. However, e-MGR introduces an additional definition made at the conceptual level, in which cover is evaluated both parsimoneously and in terms of the conceptual content of definitions, i.e., CC(d).

The fitness function accepts as input measures defined over the five e-MGR databases, and outputs parameter values for the primitive operators to drive the next cycle of interpretation. It is anticipated that, given the importance of the concept of "gluing", the function will have as its core the relationship between the proportion of the available facts covered and the conceptual glue used to achieve the cover. In order to develop a baseline for constructing a fitness function, the present experiments operate on the starting parameter set of the primitive operators, and no fitness function is used to modify these operators on each additional cycle of interpretation. Therefore, all graphs are considered fit in the experiments.

4.2. Simulation Experiments

A simple graph transformation exercise was devised for the baseline experiments related to Fuller's "Four Triangles Out of Two" problem (Fuller, 1975). Fuller presents the problem as an illustration of the phenomenon of structural self-organization through the mechanism of cooperation between the parts within each structure. Such cooperative processes are termed *synergesis*, and provide a tool for understanding processes underlying the evolution of complex organizations.

Given two distinguished triangular structures (facts), a set of definitional fragments and a starting assumption (Figure 9), the e-MGR task is to hypothesize their possible relevant relationships. The interest of the task is that in Fuller's analysis, triangles can be depicted as flat helixes (Fuller, 1975, pp 4). Using this structural depiction, which Fuller argues is common in natural systems, pairs of triangles tend to join in ways that form a tetrahedron, which is considered by Fuller as a minimal solution. Pairs of triangles thus combine to generate four triangular faces, rather than two. By analogy with continuous dynamical processes, the tetrahedron should emerge as an "attractor" in the discrete e-MGR problem space.

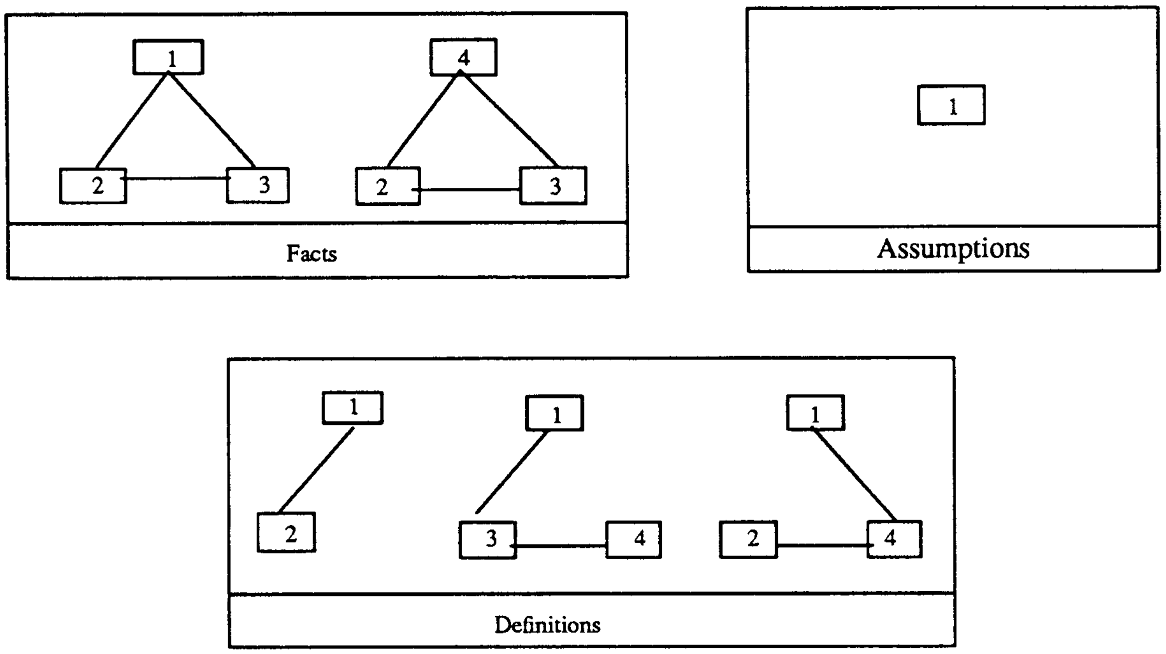

Figure 9. Given are the three sets F, A, and D of graphs that form the starting conditions of the "Four Triangles out of Two" problem.

The problem was run under the four groups of two primitive operator settings given in Table 1 below. The parameter setting for *Cl* is not shown because it is maintained as a maximal cover on all runs. The labels "least" and "most" given in parentheses after the minimal cover designation for *Sp* refer to heuristic functions applied on top of the minimal covering algorithm. These heuristics provide an additional constraint on definitional minimality at the conceptual content level, where "least" returns the set of graphs with the smallest number of concepts, and "most" returns the set of graphs with the largest number of concepts. The experiments thus include both operational definitions of "definitional material".

All of the final models in the experimental runs turned out to be "attractor" models composed of four triangular faces (Figure 10). Interesting elements to the results were as follows: i) maximal uncovering returned zero results; ii) minimal uncovering enhanced the chances of getting results; iii) minimum covering with no additional heuristic resulted in the largest number of models being generated; iv) minimum

Table 1.
Experimental Operator Settings and Results.

Run#	*Sp*	*Fr*	Total Models Generated	Final Models
1.	min	max	6	0
2.	min	min	6	1
3.	min(least)	max	2	0
4.	min(least)	min	2	0
5.	min(most)	max	6	0
6.	min(most)	min	6	1
7.	max	max	2	0
8.	max	min	2	1

covering with least number of nodes returned zero results; v) minimum covering with most number of nodes resulted in the smallest number of models being generated; vi) maximum covering resulted in the smallest number of models being generated with a successful result.

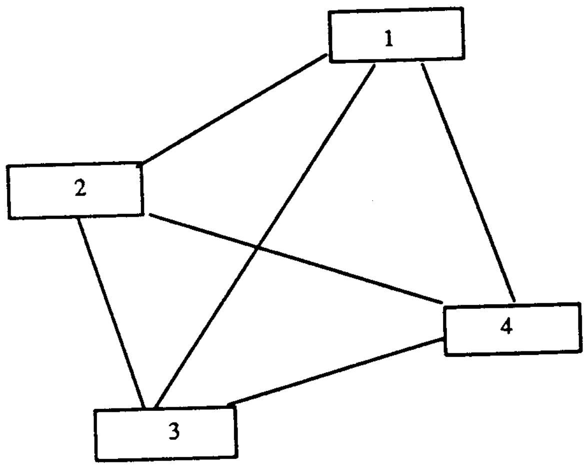

Figure 10. Given is the "attractor" model found in the "Four Triangle out of Two" problem.

The main generalization from the experiments is that problem solving in e-MGR requires cooperation between *Sp* and *Fr*. In particular, arrival at the attractor model depends on *Fr* leaving some conceptual glue, added by *Sp*, still attached to factual components during the generation of new assumptions. If all glue is removed, no results are obtained.

The above effect is enhanced by the amount of glue added during the *Sp* operation. If no glue is added, as in the the minimum cover (least) experiment, no glue can be removed; therefore, no results are obtained. On the other hand, when a maximum cover is used, a result is obtained very quickly. However, when the amount of glue added is moderated by the application of minimum cover or minimum cover (most), results are obtained but at a slower rate.

5. Conclusions and Future Work

In summary, the design of a problem solving architecture, e-MGR, fulfills the four requirements for processing in noisy and novel task environments outlined in section 1. Critical amongst these are the notions of: (i) employing transformable, object-based knowledge structures, and (ii) taking an optimization approach to control. Together, these features allow relevance relations to emerge from gluing (and ungluing) operations applied to knowledge structures in order to optimize relations between prior knowledge and data.

In section 2, it was argued that reasoning in noisy and novel task environments must have an abductive component for the generation of hypotheses, which in e-MGR is implemented by the *Sp* operator. The effects of abduction are clearly illustrated in the above experiments. The two triangles in F only become fused into a tetrahedron when *Fr* employs minimal uncover, and therefore preserves the attachment of definitional material to facts in assumptions.

The dynamics of abduction will be explored in future work. In the multidimensional search space of an AI problem, it is to be expected that the computation of globally optimal solutions will be difficult. The system will encounter "rugged" fitness landscapes in which a wide range of alternative optimal and sub-optimal solutions are distributed across the space. The problem solver must be capable of searching this space efficiently and effectively, while avoiding the traps of sub-optimal solutions. The fitness function must drive the system to a solution within the bounds of reasonable computational complexity. We have already seen potential difficulties here, since in the experiments the tetrahedron "solution" was identified faster when cover was maximal; a setting most likely to increase the problem space. However, e-MGR provides a flexible environment to balance the demands of symbolic representation against the need for adaptive control.

References

Artificial Intelligence, 13, (1980). Special Edition on Non-monotonic Reasoning.

Barnden, J. & Pollack, J. (1989). Introduction. In J. Barnden & J. Pollack, Eds. *Advances in Connectionism and Neural Computation Theory, vol. 1.* Norwood, NJ: Ablex.

Buchanan, B. G. & Feigenbaum, E. A. (1978). DENDRAL and Meta-DENDRAL: Their application dimension. *Artificial Intelligence, 11,* 5-24.

Coombs, M. J. & Alty, J. L. (1984). Expert systems: An alternative -paradigm. *International Journal of Man-Machine Studies*, 20, 21-44.

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

Coombs, M. J. & Hartley, R. T. (1988). Explaining novel events in process control through model generative reasoning. *International Journal of Expert Systems, 1,* 89-109.

Coombs, M. J., Hartley, R. T. & Pfeiffer, H. D. (1990). Developing Computing Technology for Modeling Enemy Intentions using Environmental and Doctrinal Information. *MCCS-90-184,* CRL, NMSU, Las Cruces.

de Kleer, J. (1986). An assumption-based TMS. *Artificial Intelligence,* 28, 127-162.

Doyle, J. (1979). A truth maintenance system. *Artificial Intelligence,* 28, 127-162.

Fields, C., Coombs, M. J. & Hartley, R. T. (1988). MGR: An architecture for problem solving in unstructured task environments. *Proceedings of the Third International Symposium on Methodologies for Intelligent Systems. pp.* 40-49, Amsterdam: Elsevier.

Fields, C. A. & Dietrich E. S. (1987). Transition virtual machines. I: The TVM problem-solving architecture. Technical Report # 3CR/Al-87-06, University of Colorado, Boulder, CO.

Fuller, R. B. (1975). *Synergerics: Exploration in the Geometry of Thinking.* New York: Macmillan.

Georgeff, M. P. & Lansky, A. L. (1987). Reactive reasoning and planning. *Proceedings of AAAI-87, pp.* 677-682, Los Altos, CA: Kaufmann.

Hartley, R. T. & Coombs M. J. (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.

Hartley, R. T. & Coombs, M. J. (1989). Reasoning with graph operations. In J. Sowa, Ed. *Formal Aspects of Semantic Networks.* Los Altos, CA: Morgan Kaufmann.

Hewitt, C (1985). The challenge of open systems. *Byte, 10,* 223-242.

Holland, J. H. (1986). Escaping brittleness: The possibilities of general-purpose machine learning algorithms applied to parallel rule-based systems. In R. Michalski, J. Carbonell, and T. Mitchell, Eds. *Machine Learning: An Artificial Intelligence Approach, vol. 111. Los* Altos, CA: Kaufmann.

McWilliams, G. S., Kirby, C., Fields, C. A., Coombs, M. J., Eskridge, T., Hartley, R. T., Pfeiffer, H. D. & Soderlund, C. (1989). Army requirements for an intelligent interface to real-time meteorological databases. *Pre-prints of the Fifth International Conference on Interactive and Information Processing Systems for Meteorology, Oceanography and Hydrology, pp.* 1-2. Boston: American Meteorological Society.

Nau, D. S. & Reggia, J. A. (1986). Relationships between deductive and abductive inference in knowledge-based diagnostic problem solving. *Proceedings from the First International Workshop on Expert Database Systems.* New York: Benjamin/Curnmings.

Nilsson, N.J. (1980). *Principles of Artificial Intelligence.* Palo Alto: Tioga Press.

Pierce, C. S. (1934). Scientific method. In A. W. Burks, Ed. *Collected Papers of Charles Saunders Pierce,* Harvard: Harvard University Press.

Reggia, J., Nau, D. & Wang, P. Y. (1985). A formal model of diagnostic inference. 1. ~ Problem formulation and decomposition. *Information Sciences,* 37, 227-256.

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

Stefik, M.J. (1981). Planning with constraints. *Artificial Intelligence,* 16, 111-140.

Thompson, J. R., Trout, R. & Landee-Thompson, B. (1986). *Artificial Intelligence Applications for Sensor Data Fusion.* Perceptronics and Knowledge Systems Concepts, Report A002, Contract # F30602-85-C-0107, Rome Air Development Center.

Woods, D. D. (1986). Paradigms for intelligent decision support. In E. Hollnagel, G. Mancini and D. D. Woods, Eds. *Intelligent Decision Support in Process Environments.* Heidelberg: Springer-Verlag.