The MGR algorithm and its application to the generation of explanations for novel events

M. J. COOMBS AND R. T. HARTLEY

This paper presents an algorithm for reasoning about novel events. Termed Model Generative Reasoning (MGR), we replace deductive reasoning with an abductive procedure based on the generation of alternative, intensional domain descriptions (models) to cover problem assumptions, which are then evaluated against domain facts as alternative explanations for queried events. The algorithm is principally illustrated using a problem from process control.[1]

1. Intelligent decision support for process control

A major concern in the application of computers to the control of physical and mechanical processes is the degree of autonomy to give to automated control systems. Although this has long been an important research area, it has received additional, and urgent attention since the year 1980, which has been named the "Year of Mishaps" for American technology. As pointed out by Kopec & Michie (1983), this year saw: (a) numerous near-misses at Kennedy and other busy national and international airports; (b) technical problems with the operation of the Hercules Sea Station helicopters which resulted in aborting the rescue mission to free American hostages in Iran; (c) the NORAD military computer falsely signaling a Soviet nuclear attack on three separate occasions. Moreover, these followed the incident at Three Mile Island nuclear power station in March 1979 when inappropriate operator action almost led to a melt-down of the core. A series of foolhardy operator actions actually led to such an accident at Chernobyl in 1986.

An approach to improving automated control frequently favored by analysts is to increase the intelligence of control devices through the use of expert systems. The argument runs that artificially intelligent controllers will have all the advantages of a human operator (for example, the ability to reason using a mixture of qualitative and quantitative information), while having none of the disadvantages (for example, reduced performance due to fatigue or information overload). However, a review of the reasoning abilities of current expert systems made by the first author (Coombs, 1986) found that, although the technology could make a significant contribution by ensuring that established operating procedures were rigorously applied, expert systems were unreliable when faced with the

[1] This work has been supported by the Computing Research Laboratory with funds provided by the New Mexico State Legislature, administered by its Scientific and Technical Advisory Committee as part of the Rio Grande Research Corridor. Additional support for the development of the CP programming environment has been provided by Texas Instruments (grant w-7344403) and Sandia National Laboratories (grant 95-3951).

interpretation of novel, unanticipated events. This is a serious weakness, since it seems that in such situations human reasoners are also most vulnerable, and automation would be of greatest value (see Kopec & Michie, 1983).

Although the quality of human problem-solving is often unreliable, especially when under stress, humans do have a significant ability to speculate. Using only minimal factual data, human operators arc often able to create plausible interpreta- tions of novel events out of knowledge selected from a wide range of domains and experiences. These interpretations of novel events are usually based on conceptual views formed during normal functioning, thus resulting in novel events being understood as perturbations of the normal. This can be extremely dangerous because a small local change in a physical system can result in a large change in the system's global dynamics (see Moray & Muir, 1986), and hence in the mental models required to understand and predict its behavior. A stuck inlet valve in the cooling system of a nuclear reactor, for example, can set off a sequence of events which results in the reactor going unstable.

The irony is that current expert systems not only suffer from the same bias towards normality as humans, but do so to an even greater extent. A major reason for this is that, although much effort is expended by system developers to represent faithfully the minutiae of expert heuristic knowledge, inferencing over that knowledge is usually based on classical (first order) logic. This imposes the formal constraints that: (a) all propositions map to either true of false, and (b) the problem domain be closed under a defined set of inference rules, i.e. that the domain being reasoned over be complete in the sense that those relations required to answer queries be derivable from local information in the system (Hewitt, 1985). These constraints are unacceptable for reasoning about novel events because it is not possible ahead of time to ensure that complete factual knowledge is available to cover the events or to determine the relevance of the facts that are available.

The artificial intelligence community is acutely aware of the weaknesses of classical formal inference. Indeed, considerable effort has been invested recently in trying to extend the expressive power of formal reasoning systems. One extension deals with non-monotonic reasoning,[2] which aims to provide rules for propagating the effects of new information in an inferentially incomplete knowledge-base (e.g. Davis, 1980; McCarthy, 1980; McDermott & Doyle, 1980; Reiter, 1980). Substantial work is also being done in the area of truth maintenance which involves the maintenance of the required consistency between propositions in a knowledge-base for truth-functional inference (e.g. de Kleer, 1986; Doyle, 1979; McAllester, 1982). This research is, however, still very much at the exploratory stage. This is well illustrated in the paper by Hanks & McDermott (1986)

---

[2] Classical logic is termed "monotonic" because conclusions never need to be revised with the addition of new facts. Conclusions thus increase monotonically with the addition of new facts. This feature is, however, unacceptable for much real-world reasoning, where we often have to modify or withdraw conclusions as new facts become available. Conclusions may thus vary "non-monotonically" with facts.

which demonstrated that application of the most mature theories of non-monotonic reasoning could lead to unexpected (and undesired) conclusions: [3]

There are advantages to taking a formal approach to reasoning, including the existence of a well-defined semantics and the use of a small number of representational structures. However, the difficulties presented in Hanks & McDermott (1986) become central when problem-solving about novel events because it is likely that some of the knowledge required for solution will not be explicitly present in the knowledge-base. There is, therefore, good reason to seek some alternative approach—one which preserves as much as possible of the power of a formal system but which does not sterilize natural forms of inference.

The Knowledge Systems Group at the Computing Research Laboratory, New Mexico State University, is currently developing such an alternative approach to problem-solving—Model Generative Reasoning (MGR)—to support reasoning about novel events. MGR replaces logical proof in problem-solving with a method based on the construction and empirical evaluation of models which represent alternative world views. Within MGR, models are generated from problem descriptions and general domain knowledge. These models are then evaluated against sets of domain facts to determine their status as explanations.

In the following sections, the MGR algorithm will be described, along with an illustrated discussion of the main data structures and operations used for its implementation .

2. An algorithm for reasoning by model generation

2.1 THE PROBLEM

Intelligent decision aids in process control mostly employ deductive expert systems technology (see Gallanti & Guida, 1986). Decision support functions thus essentially rely on the diagnostic capabilities of deductive inference. There are two difficulties with this approach. The first is a pragmatic problem involving computational resources; the second is a deeper theoretical problem.

The first concerns the suitability of deductive reasoning for diagnostic problem-solving in situations where the solution requires the identification of some complex of target events from a pre-defined set of system data. Nau & Reggia (1986) argue that, although it is theoretically possible to achieve such identification through deduction from data to events, the approach is computationally unattractive for many real problems. The difficulty is that simple deduction using modus ponens on domain specific inference rules of the form "IF conjunction of symptoms THEN disorder" will produce the set of all

---

[3] In a logical system, inference rules can be applied in any order. However, Hanks & McDermott (1986) demonstrate that, given two well established theories of default reasoning, a simple and seemingly transparent theory of time and a set of axioms concerning a shooting incident, it was possible to prove that a character was both alive and dead at the end of the shooting event.

disorders (process events) capable of causing any of the symptoms (data items). It will however fail to identify situations where:

(a) every set of disorders capable of causing the set of symptoms presents an equally viable diagnosis;

(b) all minimum sets of disorders capable of causing the set of symptoms should be considered as viable;

(c) all independent minimal sets of disorders capable of causing the set of symptoms should be considered viable.

The latter two situations are important in real diagnosis because they provide useful simplifying assumptions concerning the relationship between disorders and symptoms (Reggia et al., 1984). They are therefore known as principles of "parsimonious explanation" .

A solution to the above problems proposed by Nau & Reggia (1986) is to exploit the abductive features of diagnostic reasoning. In abduction, reasoning proceeds from hypothesized disorders to the possible symptoms indicative of the disorders, rather than from symptoms to logically consistent sets of disorders. The focus is thus on the construction of sets of symptoms which may he caused by given disorders. Abductive inference starts with some defined explanatory framework (of which (a)-(c) above are examples), which then guides the search for solutions. This potentially reduces search, as well as ensuring that solutions are of the desired form for the application.

The importance of abduction in problem-solving was first discussed by Peirce (1957) in the context of scientific discovery but has subsequently been found to apply widely in human practical reasoning. These cognitive processes follow the general argument pattern illustrated in Fig. 1., which has received significant attention as a reasoning structure within artificial intelligence research in areas ranging from diagnosis (dc Kleer & Williams, 1987; Pople, 1982; Reggia et al., 1984; Reiter, 1987) to text understanding (Charniak, in press; Schank & Abelson, 1977).

The second problem with deductive reasoning for diagnostic problem-solving, i.e. the deeper theoretical problem, concerns the answering of queries about novel process events. By novel events, we mean events that are not explicitly represented in the system as a whole (or as a simple conjunction of parts), and as such form the acid test of a decision aid. The requirements of problem-solvers with this capability stand in contrast to the extensional view of knowledge commonly employed in decision support systems. According to the extensional view, reasoning depends on the truth-functionality of knowledge objects (i.e. the truth or falsity of a knowledge object with regard to the world). Great care is therefore taken over the veracity of device and process descriptions represented in the system, and the accuracy of logical inferences made over them. However, the aspect of novelty implies that appropriate pre-defined knowledge objects will not necessarily be available to the system to act as the objects of deduction. Moreover, even if they can be constructed from existing knowledge, the knowledge-base will not necessarily provide a complete set of inference rules for the required deductions

on these objects, and the system will not necessarily know the relevant classes of data to be used as axioms.

An extension approach is inappropriate because there is a need to reason about alternative interpretations of propositions, to synthesize such interpretations from term definitions, to utilize partial descriptions, and to be tolerant of possible conflicts between interpretations. These require a system that is strongly intensional (i.e. strongly dependent on the non-truth functional semantics of descriptive terms,

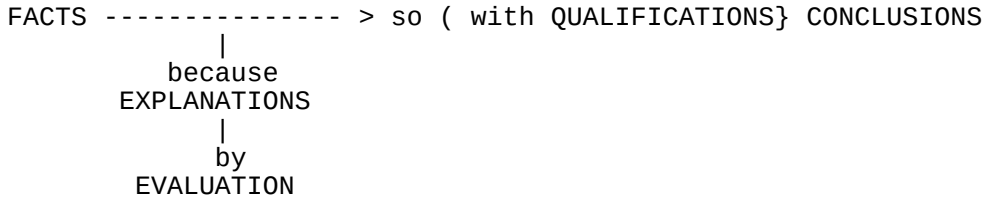


Fig. 1. Human practical reasoning (after Toulmin 1958).

rather than upon the actual existence of knowledge objects those terms represent world objects). Before it is possible to argue about the relevance of a knowledge object to some novel situation, the system will often have to construct a semantically coherent object to reason about. Moreover, work on real human reasoning in process control (e.g. Woods et a1., 1986) indicates that the actual existence of an object may be less important than the relationships it exemplifies and its role in helping an operator partition a complex system. An extensional approach would disallow such helpful fictions (see Maida & Schapiro (1982) for further discussion of the advantages of intensional representation).

The MGR paradigm has evolved to address both the pragmatic and theoretical difficulties discussed above. However, realizing that we are entering much unknown territory, we have concentrated alone on the task of developing an algorithm capable of solving problems containing novel events. Related formal and psychological questions concerning the automated construction of explanations from intensional knowledge objects have been reserved for the second stage of research. It is the basic MGR algorithm, and its realization as a prototype problem-solver, that is reported here.

2.2. THE MGR ALGORITHM

The MGR algorithm provides a general framework for computing explanations of a queried event. At the current stage of the research, we use a simple common-sense definition of explanation, the MGR goal being to account for an event by reproducing symbolically the mechanism(s) by which a coherent relationship could exist between the event and subsets of available facts, using the most parsimonious set of linking concepts.[4]

[4] The parsimony principle currently used in MGR is related to criterion (b) given in paragraph 2 of section 2. 1., i.e. all minimum sets of disorders capable of causing the set of manifestations. In MGR this becomes "all minimal sets of concepts from definitions capable of jointly covering all Of the concepts in the assumptions". The relationship becomes evident in an abductive reasoning context given an association

MGR explanations are achieved by a generate-evaluate cycle, in which intensional models of the phenomenon are constructed from definitions of domain entities and processes, and then evaluated against system facts to determine their extensional validity (i.e. the relevance of domain facts is evaluated in terms of their ability to explain facts). Queries are then answered over the set of explanations. A high-level specification of the basic algorithm is given in Fig. 2.

The algorithm starts with a statement of the queried phenomenon (a query) and an initial set of assumptions. With the basic algorithms the query will be of the form "Is phenomenon Q possible, given the assumptions?", meaning that Q only needs to be supported eventually by one of the set of generated explanations. The initial assumptions are taken from a set of available factual information about the problem world, and are selected by the user for their ability to provide a desired conceptual foundation for the query statement. At present, the selection of initial assumptions is left to the user, the only constraint being that the set of concepts within the assumptions have one mutually coherent interpretation. However, this aspect of MGR clearly influences model generation and is thus a prime candidate for automation following further work on the relationship between assumptions and explanations.

```
where r
     F = {facts}, A = {assumptions}, Q = query,
     D = {declarative definitions}, C = {contexts},
     PO = {procedural overlay}, P = {programs},
     M = {models},
     CE = {candidate explanations}, E = {explanations}
     <- means assignment

define function model-build(A):
     C <- interpret(A,D)
     P <- proc-overlay(C,PO)
     M <- execute(P)
     return M

define procedure MGR:
     A <- select(F)
     M <- model- build(A)
     loop
       CE <- evaluate(M,F)             ;evaluation consists of
       E <- query-match(CE,Q)          ;'evaluate', 'query-match'
       if acceptable(E,A,Q) then       ;and 'acceptable'
             return SUCCEED
       else
         if A' <- extract(CE) then
            A <- A + A'
            M <- model-build(A)
         elseif M <-  merge(CE) then loop
         elseif M <- generalize(CE) then loop
         else M <- empty set
     until empty(M)
     return FAIL
```

Fig. 2. The MGR algorithm

---

between definitions and the construction of potential "disorders", and between MGR assumptions and the set manifestations which the disorders explain.

The initial assumptions are used to seed model generation,[5] where model generation is a three stage process. First, assumptions are interpreted using general declarative definitions of domain entities to form a set of structural descriptions, which we term contexts. Contexts are then further developed by the addition of procedural information related to declarative structures (procedural overlays) to form programs. Finally, programs are executed to produce models. As discussed earlier in this section, models represent hypothetical world descriptions, which have not yet been evaluated against any factual information about the "state-of-affairs" in the world, and are thus purely intensional structures. Looked at another way, models represent internally coherent interpretations of assumptions which serve as structures to be passed on for evaluation.

We now come to the evaluation stage. Here, models are interpreted using factual data to determine their validity in terms of their ability to distinguish conceptually coherent subsets of current factual knowledge. Where a fact is found to be conceptually coherent with a model,[6] the model is augmented by the fact, and the combined structure is passed on as a candidate explanation. Finally, candidate explanations are matched against the query to determine their value as explanations of the queried phenomenon. Typically, a solution is not achieved in one pass through the interpreter, thus requiring further cycles.

The next cycle begins by augmenting the set of assumptions with any new facts which are found to match elements of the candidate explanations. The generation process is then continued by production of a new set of models from the augmented assumption set. Formally, this results in the specialization of the previous candidate explanations to include new descriptive features. However, if there are no new facts to augment the assumption set, then an attempt is made to merge any models which are individually coherent but provide only partial explanations of the original assumptions and facts. Furthermore, if a merge fails, then a generalization procedure is invoked in an attempt to remove those features supported by facts which block integration of the over-specialized models. Generalization thus imple- ments a form of nonmonotonic reasoning in MGR closely related to de Kleer's ATMS (de Kleer, 1986). The cycle is then repeated with the objective of generating a single model which covers, and thus explains, all of the assumptions. This may not be possible, of course, in which case the cycle continues until either all assumptions are covered by the minimal number of explanations or some user-

---

[5] The query is not included as an assumption for the seeding of model generation This is to avoid a conceptual incoherence occurring between assumptions and contexts or programs during the first pass through the interpreter which would prevent a potentially useful model from being formed By "conceptual incoherence" we mean a relationship between two concepts where they both cannot be true of a designated individual, but they both can be false with respect to that individual This is not the notion of logical contradiction commonly used in automated problem-solving, but is closer to the notion of "contrariness" The use of the query as a seed can thus over-constrain the development of model structure, blocking the addition of features which have explanatory value, and which would resolve any incoherence arising with a more sparse model This is a similar problem to that addressed by Stefik (1980) in planning, where the "least commitment" principle is used to reduce the need for backtracking

[6] Two conceptual structures are considered to be "conceptually coherent" if they share concepts which, when integrated, do not introduce any incoherencies (see previous footnote).

defined default is reached. While the existence of single explanation is all that is required for a positive answer to be given to a query in the basic algorithm presented here, we anticipate that complex queries will require qualified answers which will involve the evaluation of all available explanations (and possibly all candidate explanations as well).

2.3 AN EXAMPLE OF HUMAN MODEL GENERATIVE REASONING

The interplay between processes of model generation and evaluation captured by the MGR paradigm can be observed in human reasoning about suspected malfunctions in physical devices (e.g. Woods et al., 1982, 1986). We will therefore complete this high-level description of the MGR algorithm by presenting an illustrative protocol of human model generative reasoning.

In our example, an operator is alerted to a possible malfunction in some portion of a water cooling system by the observation that a flow-sensor, A, which has always previously given a flow-direction reading of "In", currently registers a flow-direction reading of "Out". Moreover, the expectation that sensor A should read "In" is initiated by six assumptions which, on investigation, are found to be still true, i.e. they are facts. These facts are that: (a) the pipe contains water; (b) the pipe has a given location within it marked "Top"; (c) the pipe has another given location within it marked "Bottom"; (d) flow-sensor A is positioned at the "Top" location; (e) How-sensor B is positioned at the "Bottom" location; (f) flow-sensor B reads "Out". This knowledge is represented in the initial model of Fig. 3.

A common strategy used by human operators, and one which we endorse within MGR, is first to seek an explanation of the queried event by building the simplest model out of assumptions which initiated the problem. Model 1 (Fig. 3) meets this criterion, being seeded by the six assumed facts given above, and composed from a simple interpretation of the two concepts "pipe" (i.e. a simple conduit) and "flow". Having visualized this configuration, the operator next conducts a thought experi- ment to see if the structure could produce the novel sensor readings. It can be clearly seen that it could not. The effect of water flowing from sensor A to sensor B through a simple enclosed pipe would give a flow-direction of "In" for A, and not "Out" as observed.

The configuration involving a simple pipe clearly does not explain the queried event. The operator is thus required to look for some alternative structure. A common strategy at this point is not to start building a new model from scratch, but to use the failed model to extract additional facts and to use these in conjunction with the initial set to seed a more specialized structure. This is achieved from the mapping of Model 1 to a database of facts about the pipe system. Given our concern with a water cooling system, two facts likely to be accessed are that the pipe may be a compound structure (possibly a "water-jacket") and that there may be a temperature difference between sub-bodies of water in the pipe. Accordingly, Model 2 is generated with the pipe specialized to "water-jacket". However, on evaluation it is found that even though the model coherently accommodates the new assumptions, it still does not explain the queried sensor reading, i.e. that sensor A reads "Out" .

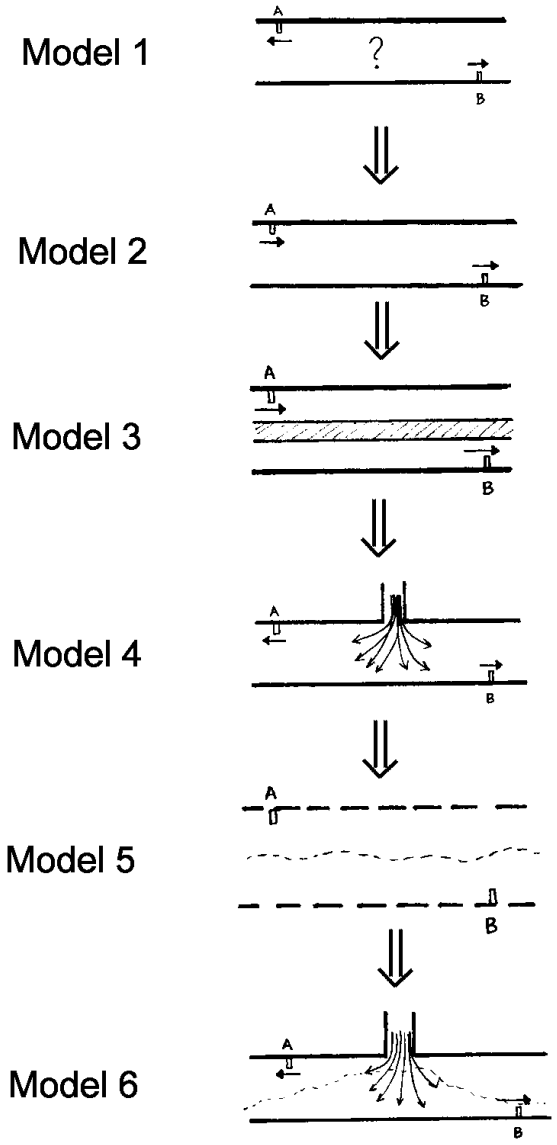


Fig. 3. Sequence of models generated during the solution Of the "flow-sensor" problem.

Given that the specialization of the pipe concept to a water-jacket fails to generate an explanation, the operator tries an alternative form of compound pipe—an "input T-junction". The evaluation of a model built around a T-junction (Model 3) indicates that it

can, indeed, explain the sensor effect, although it does so without incorporating the idea of possible temperature differences within the body of water. We thus have an explanation, but one which does not cover all current assumptions.

Reasoning could end at this point with additional facts being discovered but unincorporated. However, the protocols given in Woods et al. (1982) indicate that when explanations are required for real incidents, model building is likely to continue until most of the assumptions generated during problem-solving have been incorporated. The fact concerning a possible temperature difference in the water, it may be recalled, was covered in Model 2. The operator therefore attempts to incorporate all assumptions by merging Model 2 with Model 3 to complete the explanation. For the sake of our example, this fails due to a conflict between the defined characteristics of the two types of compound pipe—we will not allow the main pipe of an input T-junction to contain a hot water pipe.

Having identified the conflicting characteristics, the operator attempts to generali- zation them away, i.e. attempts to replace the blocking features with others that are more general but which maintain the coherence of the model. In our example, this is achieved by replacing the inner pipe in Model 2 with a "thermocline" to produce Model 4, which is both coherent (a thermocline being capable of incorporating the assumptions about water temperature) and also happens to generate the queried flow-sensor reading for sensor A.

We now have two alternative explanations for the sensor phenomenon: Model 3 and Model 4. However, both models individually fail to explain all of the assumptions. Model 3 does not account for the existence of a temperature difference, and Model 4 has generalized the pipe concept from a compound pipe back to a simple conduit. An attempt is therefore made to merge the two explanations in order to achieve the most comprehensive explanation by covering the maximum number of facts. The result is Model 5, which concludes the investigation.

3. MGR knowledge structures and basic operations

3.1. THE CP ENVIRONMENT

A programming environment has been developed to provide the correct epis- temological mechanisms for the MGR algorithm. This software, which is available on a Symbolics 3670, embodies a conceptually based theory of knowledge representation based on proposals by Hartley (1986) named CP—(C)onceptual (P)rogramming. This section will present the main features of CP in the context of a simple application of model generative reasoning.

The present example involves a different domain from the water cooling system illustration presented informally above. The water cooling problem will be considered again in Section 4 when we discuss the full MGR interpretation cycle. For the present, however, we will concentrate on details of knowledge representation and model generation. These aspects of MGR are illustrated more clearly by an example involving only a small number of entities and processes, and in which a solution can be reached

after only a single pass through the interpreter. We have such an example in a problem concerning the choice of paints for protecting the external surface of a cold water pipe. Given a tendency for condensation to form on such pipes, we want to know whether there is some paint which is suitable for painting a wet surface.

Two classes of information are represented in the CP system: definitions and situations. Definitions contain both declarative and procedural aspects of meaning and are contained in a definitional component of the MGR architecture. Situations can be either given externally, in which case they reside in a factual component, or are internally generated by procedures, in which case they reside in an assertional component. Our approach varies from other existing knowledge representation schemes (e.g. KL-ONE—Brachman & Schmolze, 1985), in that it allows for explicit representation of procedures, rather than procedural interpretation of conceptual structures. A semantic network formalism based on conceptual graphs (Sowa, 1984) is employed for both classes of information. Thus the four main MGR structures (contexts, programs, models and explanations) are all formed as compositions of conceptual graphs.

In his book "Conceptual Structures" John Sowa describes a knowledge repre- sentation scheme, the atoms of which are concepts and conceptual relations. These are arranged into conceptual graphs according to a set of formation rules, which govern their ultimate structure. Presenting these rules axiomatically, Sowa gives the following canon. Graphs formed according to the rules are termed canonical. Its parts are:

(1) a type hierarchy which relates concepts according to the principles of specialization and generalization;

(2) a set of individual markers which are the internal map of real-world objects;

(3) a conformity relation which ensures that individual markers are tokens of the right concept type;

(4) a finite "starter" set of graphs assumed to be canonical;

(5) a set of formation rules with which new canonical graphs can be derived from existing ones, the main rule being the conceptual join (to be discussed later in this section).

Graphs acquire their meaning through the notion of canonicality, and through definitional mechanisms for new concept types. Here, the type hierarchy is considered to be the backbone of the representational system, providing the internal mechanism which propagates meaning from concept to concept, as with other network-based formalisms (e.g. KL-ONE—Brachman & Schmolze, 1985; SNePS Shapiro, 1979). Also, like other knowledge representation systems, a clear-cut method for determining the denotations of the internal symbols is necessary. The individual markers (which are attached to real-world individuals by ostension) and the conformity relation ensure this.

Following Sowa (1984), there are two mechanisms for definition. Concept types can be defined in an Aristotelian way, i.e. through a set of necessary and sufficient conditions by which the type may be distinguished from all other subtypes of its parent type; they can

also be defined using schemata, which express alternative contexts in which a term can gain meaning. Schemata are, however, given a position within the type hierarchy by being placed under an empty type label which locates them with reference to some fully defined supertype.

An example of an Aristotelian definition is of a paint as a "a fluid covering for a surface". It should be noted that this definition is not only very sparse, but also fails to mention many of the attributes people would consider when reasoning about "paint". There is no mention, for instance, of color or covering properties. This is a significant weakness because, in order to support naturalistic expert reasoning, it is important to capture the meaning of terms in a form that reflects their use in the problem domain. Schematic clusters provide ideal structures for this. Figures 4 and 5 give two alternative schema for "paint" which will be used in our example, one for "oil-paint" and one for "acrylic-paint". Given that the focus of the current problem is on the interaction between paint and wet

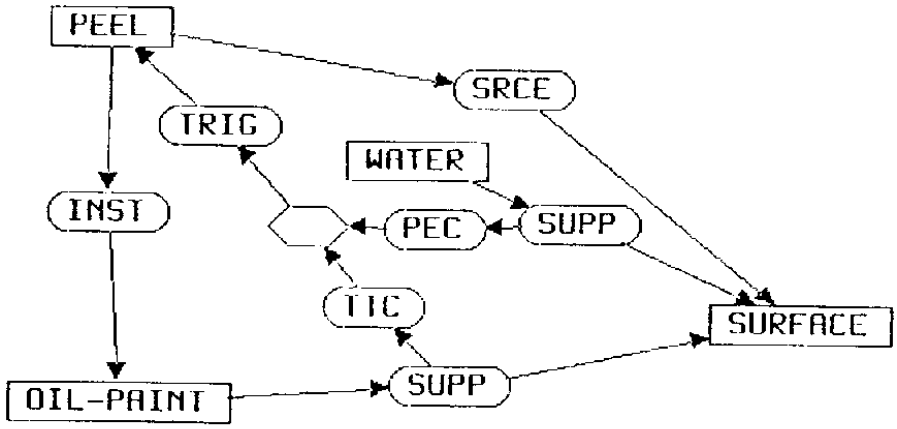


Fig. 4. Definition of [OIL-PAINT] with procedural overlay.

surfaces, the schema are built around concepts that are important in this context: [ABSORB] and [PEEL].[7] Oil paint is defined as peeling off a wet surface, while acrylic paint is defined as absorbing surface water. The diamond nodes represent actors which implement the procedural aspects of definitions. In reality, these procedural components are stored separately in CP as procedural overlays, but are here given along with the declarative component for the sake of clarity. Actors function like "active concepts" which accept states as preconditions and events as triggers. Having been triggered, they

[7] Square nodes "[ ]" indicate concepts; round nodes "( )" indicate relations; angled nodes "<>" indicate actors. In later graphs which require the designation of individuals, # will be used to indicate a named individual (e.g. [PERSON: #John]), and * will be used to indicate a generic concept (e.g. [PERSON: *x]). By default, concepts having no explicit designation for individuals are taken to be generic.

assert states and enable further acts as by-products of their activity. These actors may be used to express causality, involving states and events, and inferences, involving propositions.

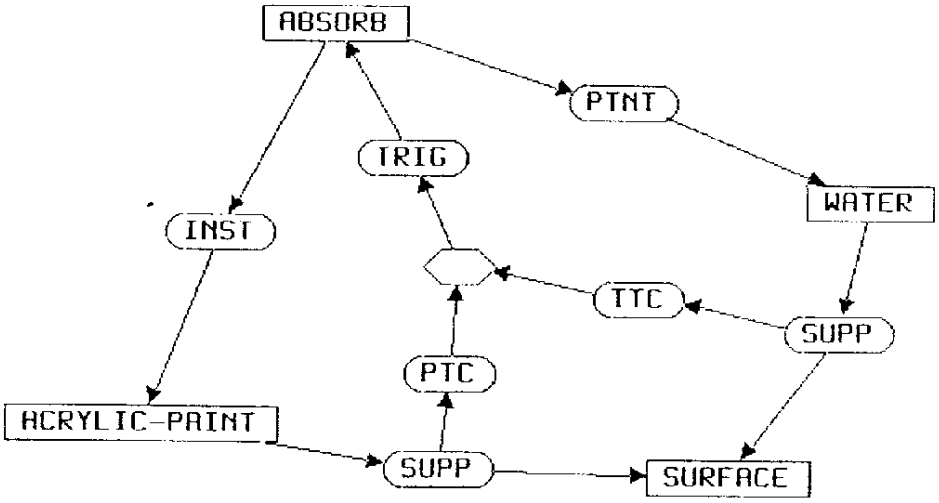


Fig 5. Definition of [ACRYLIC-PAINT] with procedural overlay.

In Fig. 4, the actor implements the peeling of oil-paint when it is applied to a wet surface. The actor is linked to its inputs and outputs via special "actor-relations" which collectively express a factorization of the combination of states and events typically found in a theory of action (cf. Rieger's similar treatment in commonsense algorithms: Rieger, 1976). To represent the peeling of oil-paint, two inputs are needed: a (P)ersistent (E)nabling (C)ondition, where the surface to be painted is supporting water, and will still be supporting water at the completion of the painting event (expressed via the 'PEC' relation); a (T)ransitory (T)riggering (C)ondition consisting of the application of the oil-paint to the wet surface (expressed via the relation "TTC"). Together, these inputs (TRIG)ger as output the act of peeling.

The graphs involving actors are actually abbreviated from graphs containing embedded graphs of type STATE and EVENT. However, the embedded level has been removed, unambiguously, since the nature of actor inputs and outputs arc restricted to use with only one of these types. Any actor linked to a conceptual relation (via one of the actor-relations shown in Fig. 6) has a STATE as input or output; an actor linked to a concept has an EVENT as input or output. The meaning and use of the actor relations in Fig. 6 are described more fully in Hartley (1986). Brief captions have been provided for each of the relations in the figure to give the reader a sufficient intuitive appreciation of their meaning for an under- standing of the example.

All inference procedures used by the MGR algorithm are based on the single formation rule known as the conceptual join. Join takes two conceptual graphs and unifies them by

replacing any pairs of concepts having a common specialization with their specialization so that both sets of relational attributes are attached. Concepts are regarded as candidates for joining if:

1. they have identical labels and compatible designations for individuals (if any), (e.g. [PERSON: #John] join [PERSON: #John], [PERSON: #John] join [PERSON: *y]);

2. they have a common specialization as indicated by the type hierarchy in the Definitional Component, in which case the two concepts are replaced by their specialization (e.g. assuming 4*4 < TRUCK < PHYSICAL-OBJECT and 4*4 < TRUCK < MOBILE-ENTITY, then [PHYSICAL-OBJECT] join [MOBILE-ENTITY] = > [TRUCK] . [8]

As stated above, join is used in CP as the basic operation, providing the means of building models from definitions and assumptions and of mapping models to facts during evaluation. It also offers some measure of control over the growth of conceptual structures. Joined concepts may be specialized by different degrees, pairs of concepts being replaced by their "greatest common specialization" at one extreme ([4*4] in 2. above)* and by their "least common specialization" at the other ([TRUCK] in 2. above). The most important operation which uses the conceptual join for MGR is the "maximal join" which is the result of joining two graphs on all possible concepts. A further related operation provided by CP is project, which abstracts common generalizations from pairs of conceptual structures and can be used to derive their points of discrepancy.

3.2. THE BASIC MGR REASONING CYCLE

In order to illustrate the implementation of the MGR algorithm in CP, we will return to the problem of choosing a paint to protect water pipes. This problem has been especially constructed so that it can be solved in a single pass through the algorithm, involving a single act of specialization. Such simplification is desirable because it enables us to focus on the process of model generation without having to be concerned with the complexities of evaluation against multiple facts.

The task is to explain whether "paint will remain on some surface", given the *assumption* that "the paint is applied to the surface when it is coated with water". Both the assumption and query are given in graphical form in Fig. 7.

Before starting our example, we need a definition for the process of painting a surface, in addition to definitions of oil-paint and acrylic paint as given in Figs 3 and 4, respectively. This is given in Fig. 8 as a schema which mentions a brush as an instrument .

[8] X < Y states that X is a subtype of Y, or alternatively that Y is a supertype of X.

The first step is to apply the *interpret* procedure (see the MGR algorithm in Fig. 2). *Interpret* interprets the assumption set by joining it to the set of definitions (using maximal joins), so that the number of concepts in the assumptions that are covered by concepts in the definitions is maximized, and excess concepts in these definitions (i.e. not mentioned in the assumption set) is minimized. A join can be rejected when either count is inadequate. The result of *interpret* is a set of contexts, each covering the assumption contexts with a mutually exclusive set of definitions. Contexts are located in the *assertional component* of the system. In the example, coverage of the assumption

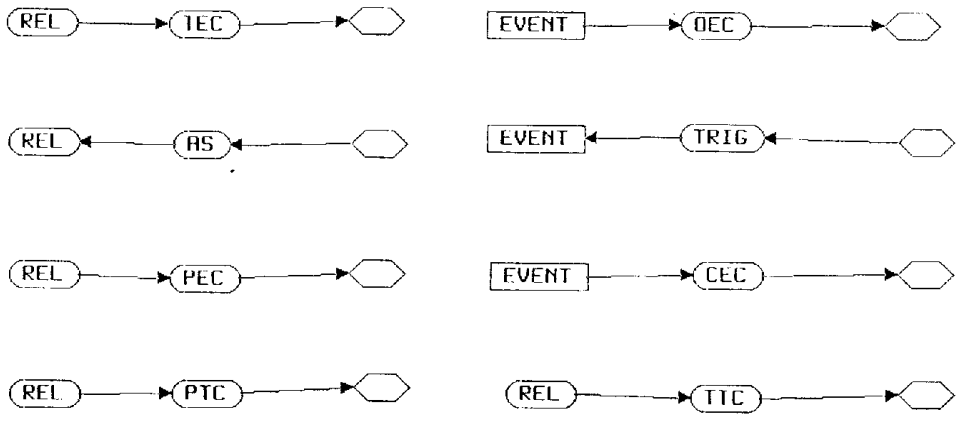
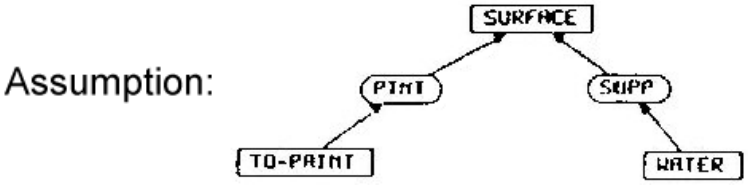


Fig. 6. Actor inputs and outputs.

concepts ([TO-PAINT], [SURFACE] and [WATER]) requires joins to the definitions of



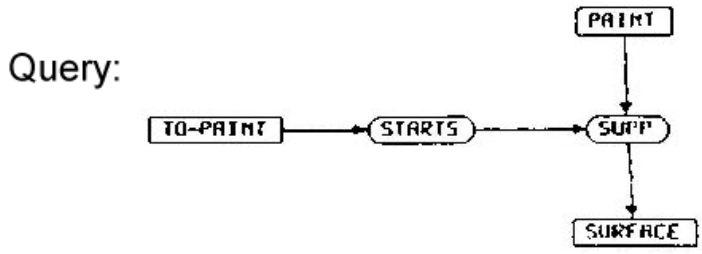


Fig. 7. Assumption and query.

[TO-PAINT] and either [OIL-PAINT] or [ACRYLIC-PAINT]. The contexts resulting from these interpretations are given in Fig. 9.

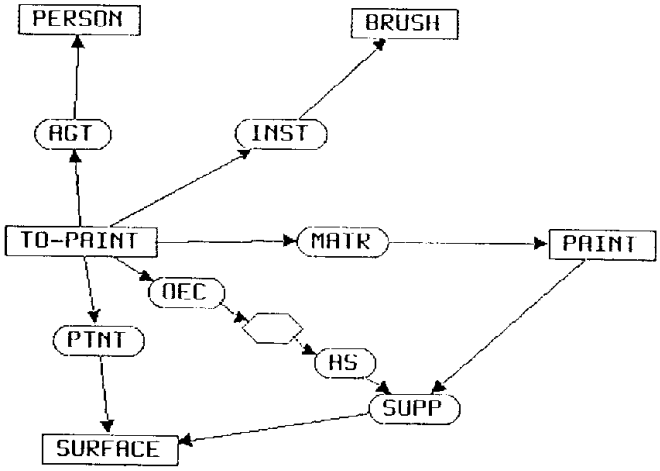


Fig 8. Definition of [TO-PAINT].

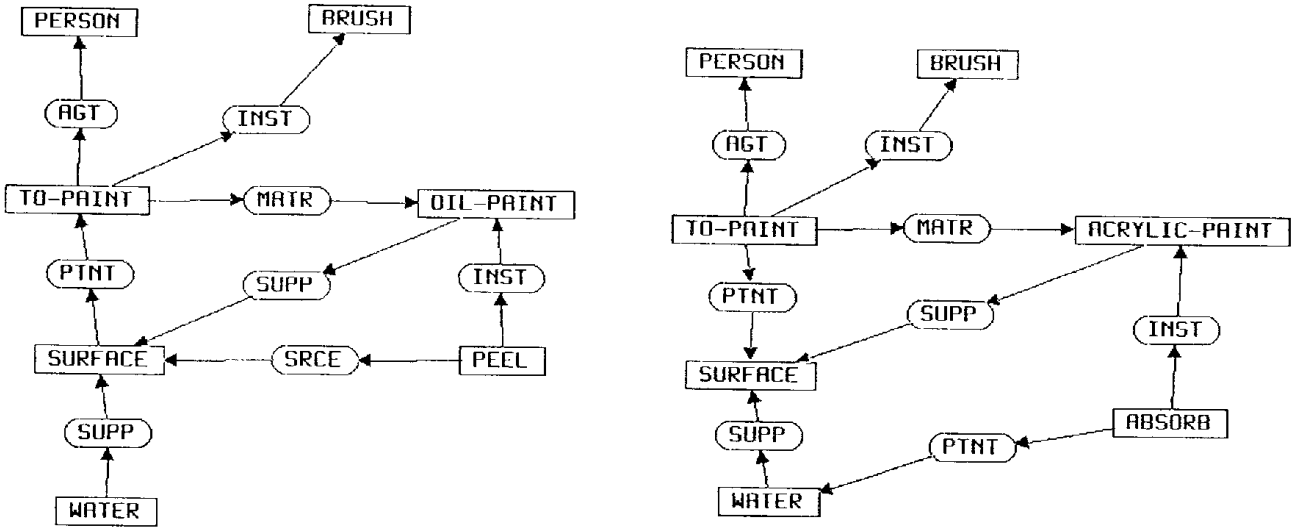


Fig. 9. Contexts for [OIL-PAINT] and [ACRYLIC-PAINT].

Proc-overlay is next applied to join each context to procedural overlays for the three definitions to produce 'programs' which will generate models when the actors contained in them are initiated. In general, each definition can have many procedural overlays, each representing a different type of strategic operation. For each context, there may therefore

be several programs. The single program generated for [ACRYLIC-PAINT] in the example is shown in Fig. 10. The program for [OIL-PAINT] is similar.

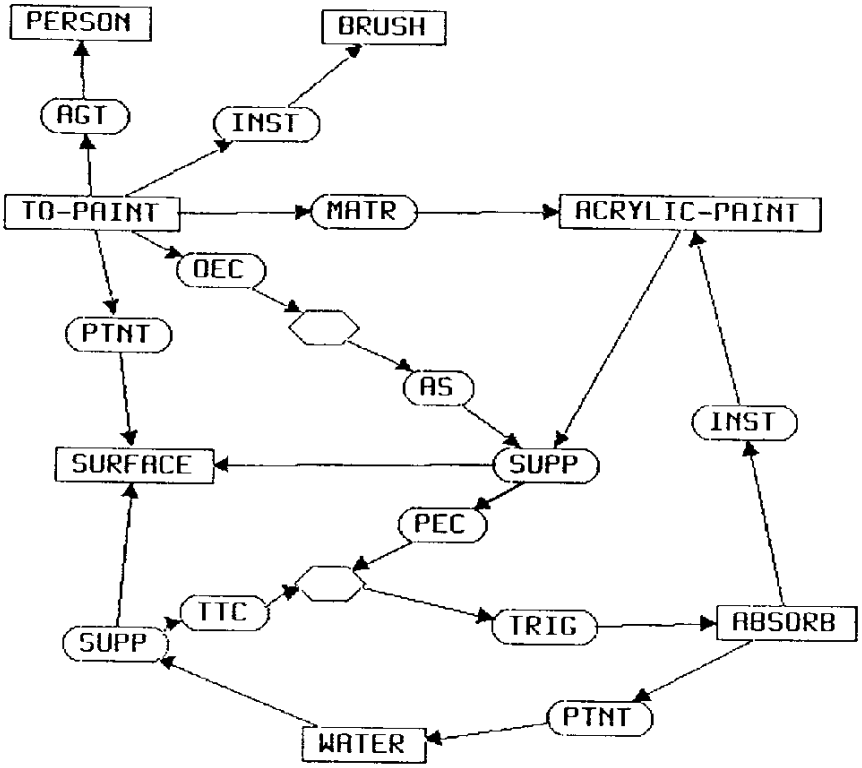


Fig. 10. The program for [ACRYLIC-PAINT].

The programs are now executed to produce models. Programs are essentially local rule-systems which express causality through a theory of action similar in spirit to Rieger's commonsense algorithms (Rieger, 1976), or any other sort of inference with an epistemology involving independent, parallel actors, with typed pre- and post-conditions. The results of running a program are models in which causal components are replaced one by one by temporal relations following a scheme developed by Allen (1983), together with a program "trace" in the form of a time chart of execution. The model's time chart is similar to the time maps of Dean and McDermott (1987) which represent patterns of co-existing objects and processes. The reader should consult this paper for details of this approach.

In the paint problem, there is only one actor with no pre-conditions which are dependent on other actors. It immediately instantiates the state of a surface having paint on it. This in turn causes the "absorbing" actor to fire, and it "absorbs" the water on the surface. The whole model is a declarative structure showing the temporal relationships between events and states each having reference to surfaces, people, paints and brushes. The [OIL-PAINT] program goes through the same sequence, but the model shows how the paint on

the surface peels off, allowing the water to remain. The model for [ACRYLIC-PAINT] is shown in Fig. 11. Temporal relations should be self-explanatory.

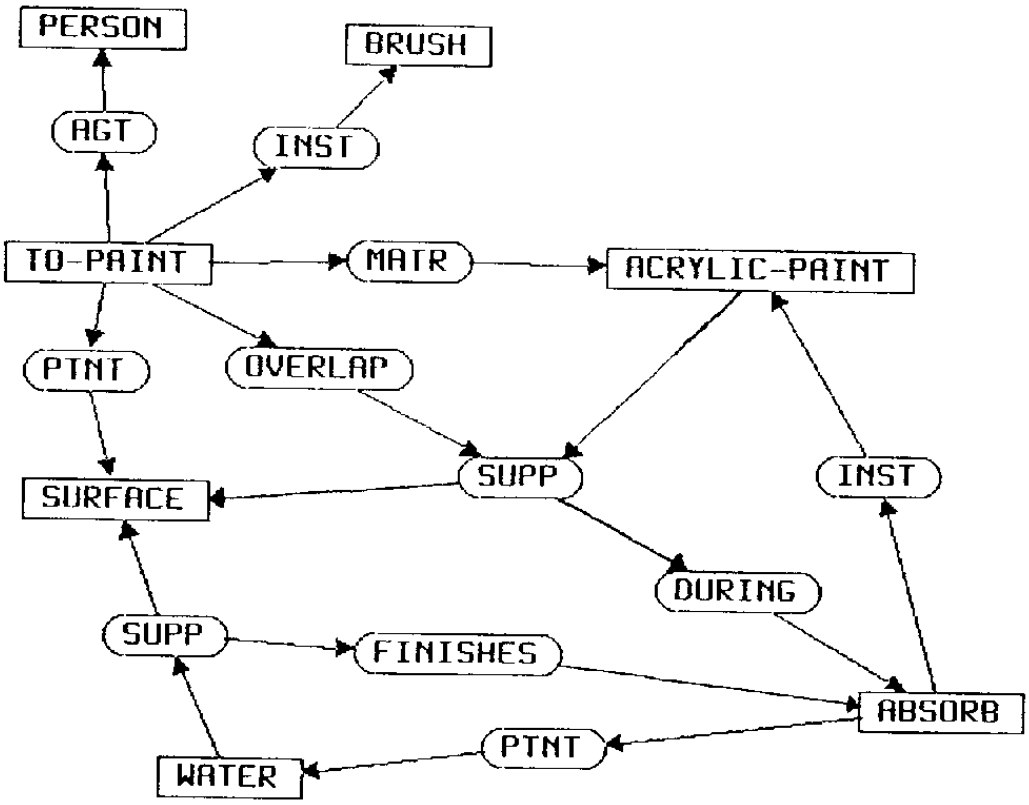


Fig. 11. The model for [acrylic-PAINT].

The models are now *evaluated*. Since models contain more concepts than those present in the initial assumptions, they may gain further support from the facts. The extent of factual support may be determined by a further join between models and facts. Within MGR facts are considered as unconnected and thus a powerset. Pairs of facts may also be conceptually incoherent (see previous footnote), each fact supporting a different model and thus being separately true, while being conjointly incoherent. Incoherence is determined by reference to the type hierarchy. Two concepts are considered to be implicitly incoherent if they refer to the same individual and are located on different branches of the hierarchy, or if they are located on the same branch but refer to different individuals and are jointly dominated through the same relation by a common, individualized concept.[9] The notion of "domination" concerns the direction of relations,

[9] Assuming a type hierarchy in which [PEEL] < [ACT] and [ADHERE] < [ACT], then the facts Fl. [PEEL: #12] and F2. [ADHERE: #12] are incoherent, and the facts [PEEL: #12]-> (LOC)-> [PlPE:#7] and [PEEL: #12]-> (LOC)-> [PIPE: #9] are incoherent. This definition has been found to be too restrictive for many applications. In the later case, [PIPE: #7] and [PIPE: #9] may in some instances be best assumed to be

the concept [A] is said to dominate another concept [B], if [A] is related to [B] with the directionality [A]-> (R)-> [B]. In addition, concepts are regarded as explicitly incoherent if they exceed a "relation restriction" placed on them. A [PIPE] for example, is only allowed to (POS) two [ENDS]. This notion is similar to that of a "range restriction" used in KL-ONE (Brachman & Schmolze, 1985).

In the example, evaluation is made with reference to the one known fact (Fig. 12) residing in the *factual component* of CP. Here, the fact which concerns the possession of acrylic-paint by an individual "Fred" joins with the [ACRYLIC- PAINT] model alone. However, the [OIL-PAINT] model is not incoherent with this fact; it simply fails to relate to it. It is perfectly possible for there to be two, totally unconnected situations in our domain, i.e. Fred's possession of acrylic paint, and a case of peeling oil paint. Processing therefore proceeds with a query-match to both of the models. Here the [OIL-PAINT] model again fails, and the [ACRYLIC- PAINT] model SUCCEEDS. Since there is at least one supported model, the match is found to be acceptable. The answer to the user is therefore "yes, as long as the paint is acrylic", the justification for this conclusion being the single explanatory model in which [PAINT] is specialized to [ACRYLIC-PAINT].

Fig. 12. The single problem fact.

4. The full MGR interpretation cycle

4.1. EXPLAINING UNEXPECTED "FLOW-SENSOR" READINGS

In order to illustrate the cyclical aspects of the MGR interpreter, we will present a solution to the flow-sensor problem described informally at the end of section 2. It may be recalled that the problem is concerned with explaining an unexpected flow-sensor reading in a water pipe, the sensor indicating that water is flowing out of the pipe at an end which the operator thinks of as an input. We assume that this is a novel situation for the plant operator, that he has limited factual information available to him about the state of the plant, and that he wants to make best use of this information in determining whether the situation is coherent (i.e. can be explained) .

---

different sections of some larger, unnamed pipe [PIPE: *x], in which case the two facts will not be incoherent.

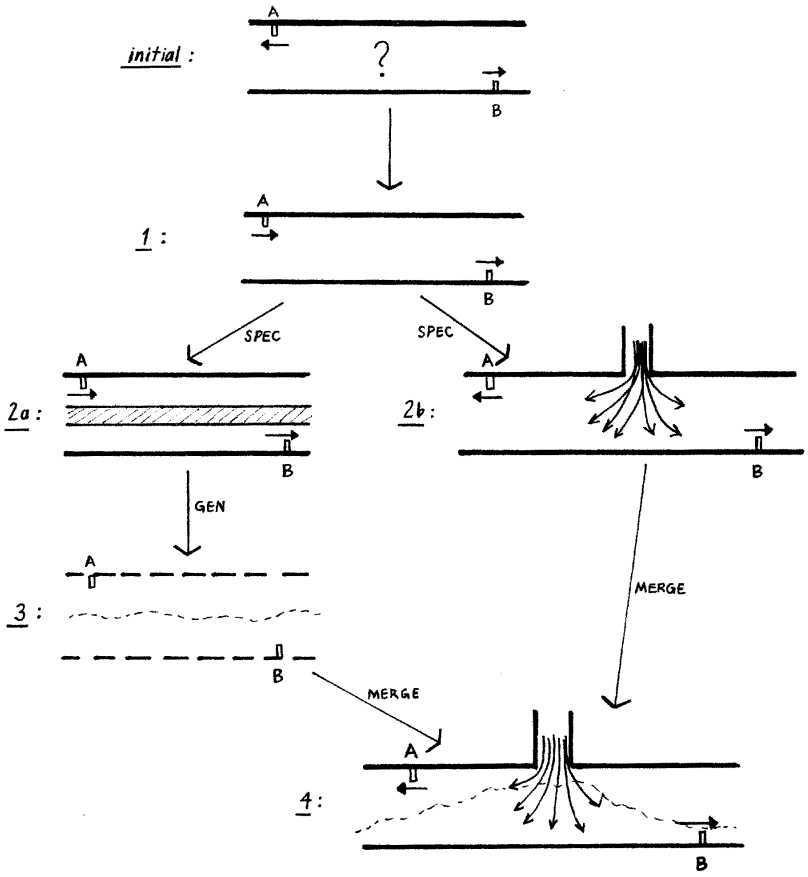


Fig. 13. Explanatory sequence for the "flow-sensor" problem.

The sequence of models offered as candidate explanations by the MGR algorithm is illustrated in Fig. 13. The order in which models are generated is the same as in the informal example presented in section 2.3, with the exception that models based on sets of schemata are constructed in parallel.

The hierarchy of concept types and embedded schemata[10]: presented in Fig. 14 gives all of the concepts available within the example as general domain knowledge on pipes and the behavior of fluids. This knowledge is located in the definitional component of the CP system. Key definitions to be used in the following discussion are given in Figs 15 and 19. It should be noted that many of these definitions have been much simplified for the sake of exposition. In particular, because the example focuses on the gross effects of water flow through pipe systems, we have combined the motions of pipe structure and water flow into a single concept (i.e. [PIPE- FLOW]). However, in a real application, the interpreter would be expected to construct such complex concepts for itself.

[10] The names of embedded schemata are enclosed within angle-brackets - (. . .) - and printed below their type labels. Relational restrictions are enclosed within slashes - /. . . /.

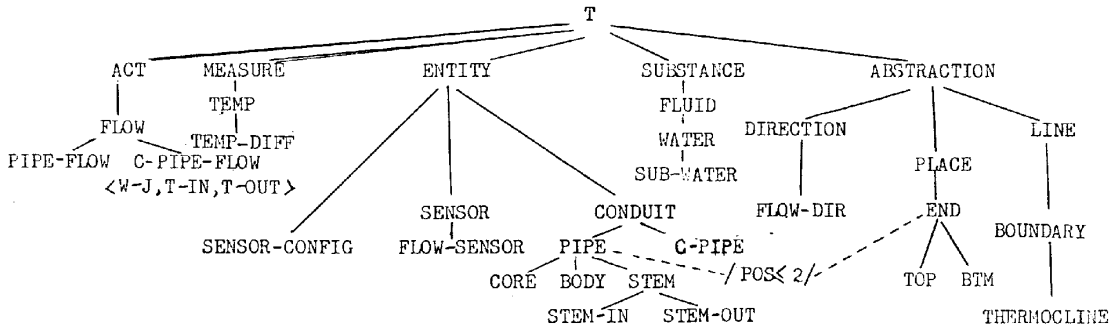


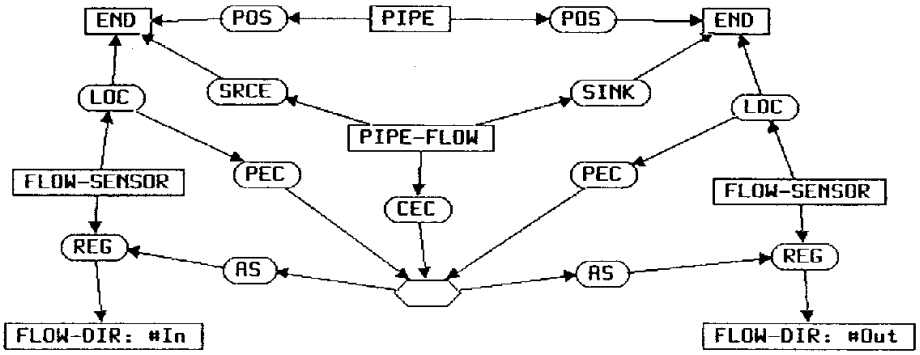
Fig. 14. Type hierarchy for the "flow-sensor" probelem.

4.2. CYCLE 1

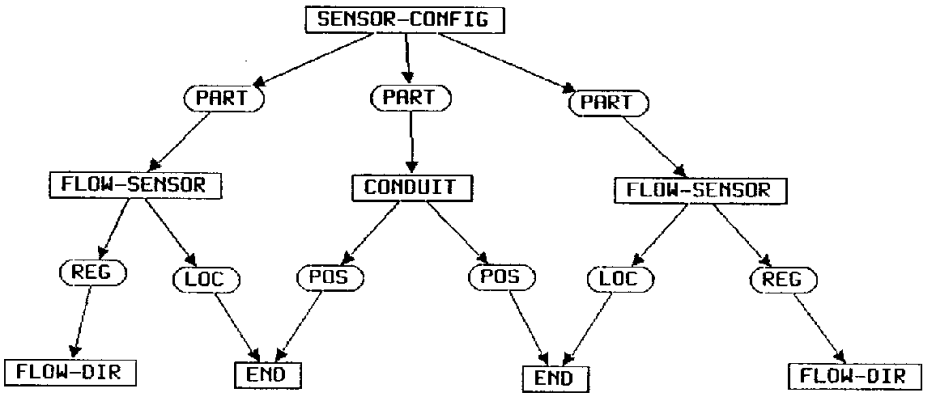
We start processing with a statement of the problem query and the initial assumptions. These are presented in Fig. 16. In the current implementation the initial assumptions are required to join, so forming a single coherent structure with which to seed the MGR interpreter.

The query is represented as the simple statement that [FLOW-SENSOR: #A1 is reading [FLOW-DIR: #Out]. The assumption is composed from a set of related statements. These jointly describe a situation in which there exists a [PIPE] which: contains [WATER]; (b) has a marked position [TOP]; (c) has a marked position [BTM]; (d) has a [FLOW-SENSOR: #A] located at [TOP]; (e) has a [FLOW-SENSOR: #B] located at [BTM]; (f) has a reading of [FLOW-DIR: #Out] for [FLOW-SENSOR: #B]. It may be recalled that assumptions have the status of facts within MGR. However, at this stage of processing, the assumed facts cannot be considered to be necessarily relevant to any explanation of the queried event and thus may need to be "removed" at a later stage.

schema for PIPE-FLOW



procedural overlay for PIPE-FLOW



schema for SENSOR-CONFIG

Fig 15a. Key definitions for cycle 1 of the "flow-sensor" problem.

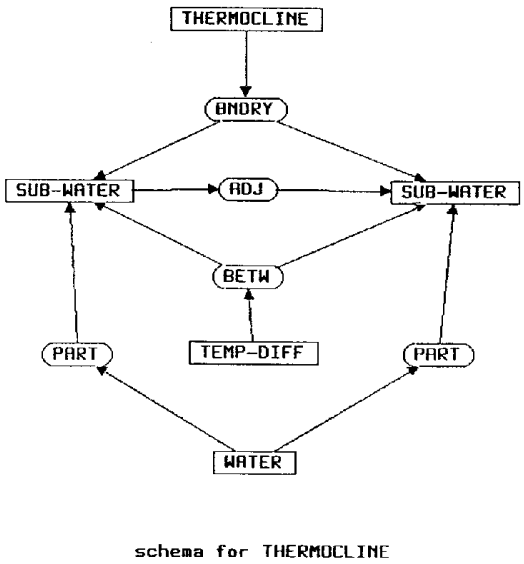


Fig 15b. Key definitions for cycle 1 of the "flow-sensor" problem.

Entering the model generation phase of the MGR algorithm (see Fig. 2), the system first builds contexts by interpreting concepts mentioned in the assumptions by replacing them with their definitions. In our example, this results in the following covers for the 6 assumption concepts:

Assumptions	Definitions
[PIPE]:	[PIPE-FLOW]
[WATER]:	[THERMOCLINE], [SENSOR-CONFIG]
[TOP]:	[SENSOR-CONFIG]
[BTM]:	[SENSOR-CONFIG]
[FLOW-SENSOR]:	[SENSOR-CONFIG]
[FLOW-DIR]:	[SENSOR-CONFIG]

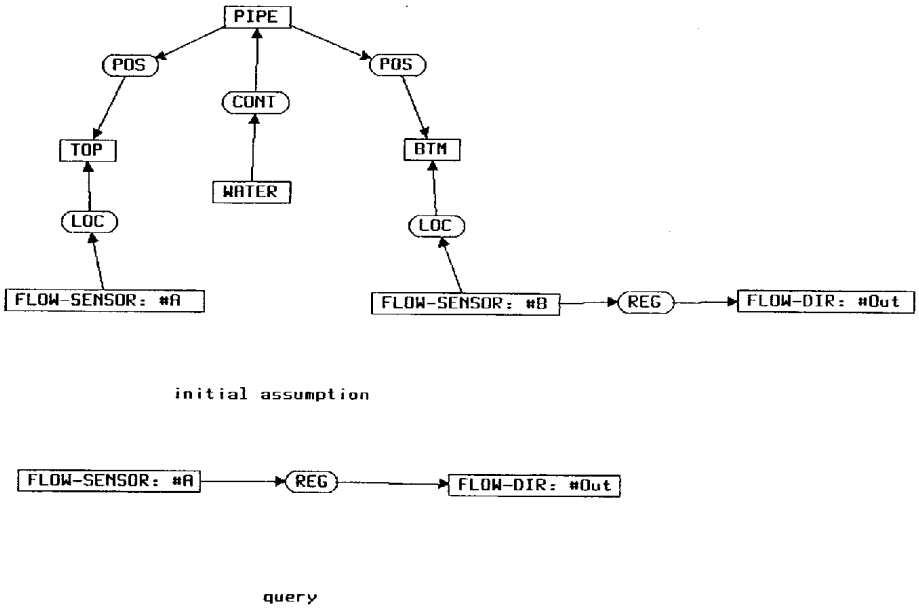


Fig. 16. Initial assumption and query.

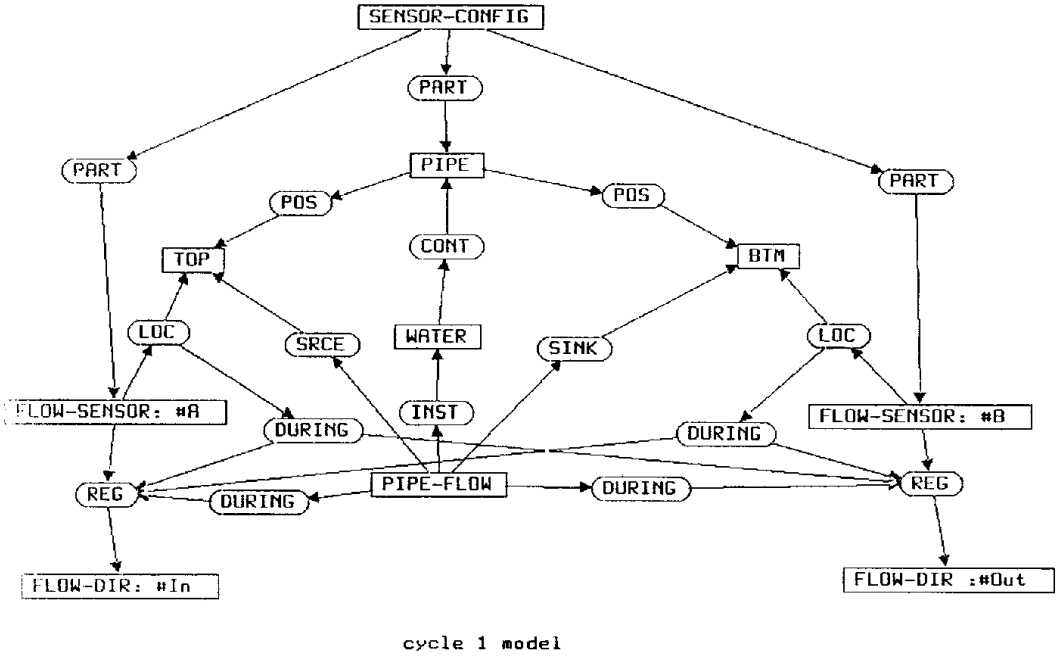


Fig. 17. Model 1 with a simple [PIPE] and [PIPE-FLOW].

All of the above covers are self-explanatory except those for [TOP] and [BTM]. [SENSOR-CONFIG] is a cover for these two concepts because it mentions the concept [END], and [TOP] and [BTM] are subtypes of [END] (see the type hierarchy in Fig. 14).

Following the conceptual parsimony principle for constructing explanations discussed in section 2 (see especially footnote on page 683), the interpret procedure next seeks the most parsimonious joint cover for all of the 6 assumption concepts. Out of 3 possibilities, there is a single smallest conceptual cover comprising the set {[PIPE-FLOW], [SENSOR-CONFIG]}. We thus have a single context, which evolves by the addition and execution of procedural overlays (see Fig. 15) into a single model. This is Model 1 of Fig. 13, which is also given in graphical form in Fig. 17. Note how [SENSOR-CONFIG] dominates [THERMOCLINE] according to the principle of parsimony.

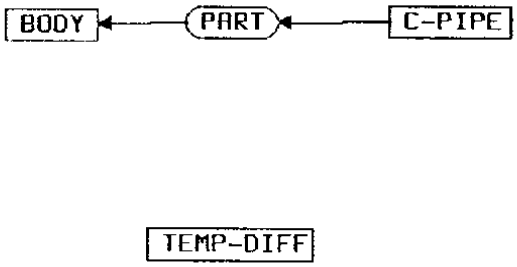


Fig. 18. Domain facts.

The model generation phase is followed by model evaluation, in which Model 1 is evaluated against the Factual Component in order to identify any new supporting factual evidence. In the example we have only two facts (Fig. 18): (F1) the statement that there exists a compound pipe—[C-PIPE]—with a [BODY] element, and (F2) that there exists a temperature difference—-[TEMP-DIFF]. Moreover, [BODY] in (F1) joins with its supertype, [PIPE], within the model. Model 1 thus becomes a candidate explanation for the query. It is therefore passed to query-match to be tested against the query statement (Fig. 16). This, however, fails, given the conceptual incoherence between the [FLOW-DIR: #In] of [FLOW-SENSOR: #A] of the candidate explanation derived from Model 1 and the reading of [FLOW- DIR: #Out] in the query, thus resulting in Model 1 being rejected as an explanation.

4.3. CYCLE 2

The MGR cycle continues with the augmentation of the set of problem assumptions with the new fact [BODY->(PART)->[C-PIPE]. When the new fact is joined to the initial assumption, the concept [PIPE] becomes specialized to [BODY]. This effectively removes [PIPE-FLOW], which only mentions [PIPE], from consideration (see the initial assumption in Fig. 16 and the schema for [PIPE-FLOW] in Fig. 15a). Specialization thus

arises as the default direction for model development in MGR through the effects of join applied during the manipulation of assumptions.

Having created a revised assumption set, it is subjected to re-interpretation. This brings in some new definitions because the concept [BODY], which was introduced via fact (F1), joins to the set of schema related to the [C-PIPE-FLOW] subtype of [PIPE-FLOW]: ([T-IN]), and input T-pipe; ([T-OUT]), an output T-pipe; ([W- J]), a water jacket. The definitions of new concepts involved in this second cycle are given in Fig. 19.

Three [C-PIPE-FLOW] schemata give the possibility of three specialized models being generated from the new conceptual covers:

{[SENSOR-CONFIG], ([T-IN])}

{[SENSOR-CONFIG], ([T-OUT])}

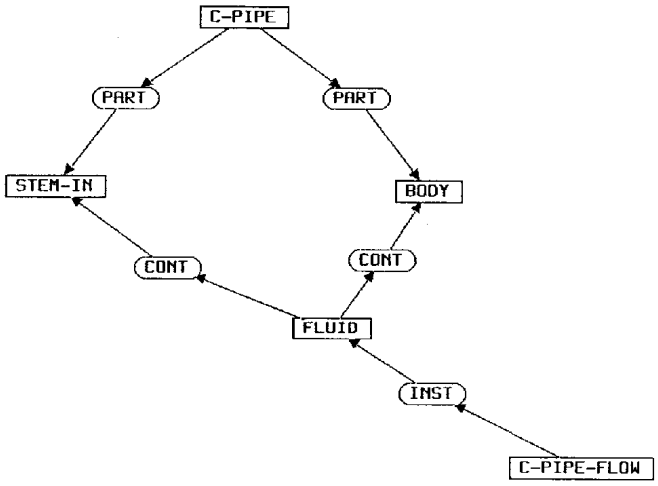
{[SENSOR-CONFIG], ([W-J]) } .

However, a conceptual incoherence between [FLOW-DIR: #Out] in the assump- tions and [FLOW-DIR: #In] for [FLOW-SENSOR: #B] arising during execution of the ([T-OUT]) program reduces this to two models. The models include one involving an input T-pipe and one involving a water jacket. These are pictured in Models 2a and 2b of Fig. 13, and presented as graphs in Fig. 20.

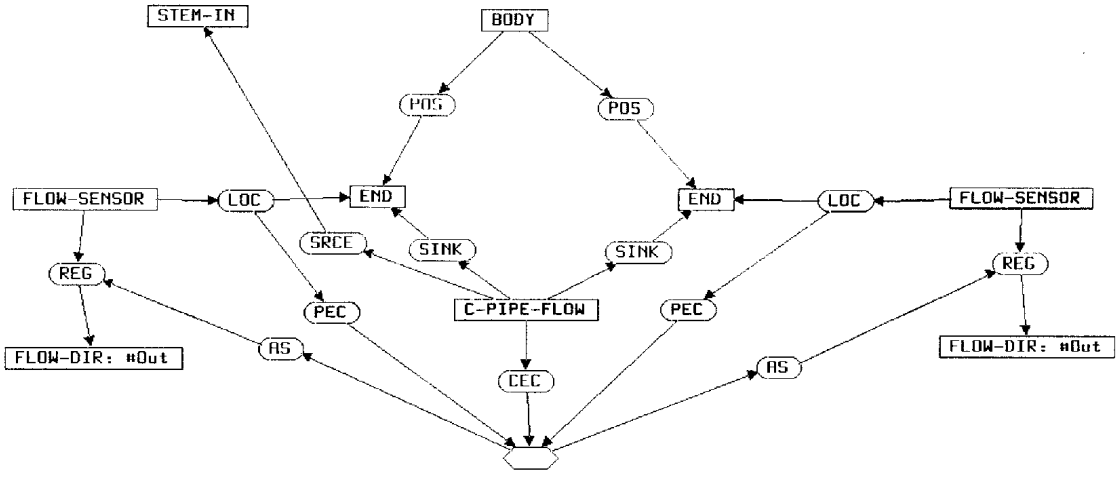
Model 2a and Model 2b are next subject to evaluation. This reveals a join between (F2)--[TEMP-DIFF]--and [TEMP-DIFF] in <[W-J]>. Both models therefore attain the status of candidate explanations. A call to the procedure query-match follows, which results in Model 2b being found to be an explanation with [FLOW-SENSOR: #A] reading lFLOW-DIR: #Out]. Model 2a, however, reads [FLOW-DIR: #In] for [FLOW-SENSOR: #A] and thus fails as an explanation of the query.

4.4. CYCLE 3

After two MGR cycles, the system has generated one model (Model 2b) which explains the query, but which does not cover all known facts—[TEMP-DIFF], and one model (Model 2b) which does not explain the query, but which does cover all facts. Following our algorithm, this situation triggers a merge procedure which seeks to integrate the two models into a single structure. Merge is essentially an operation of maximal join. In our example, this fails because of conceptual incoherence since a [PIPE] cannot have more than two [END]s). This incoherence arises from an explicit relation restriction recorded in the type hierarchy between the concepts [PIPE] and [END].

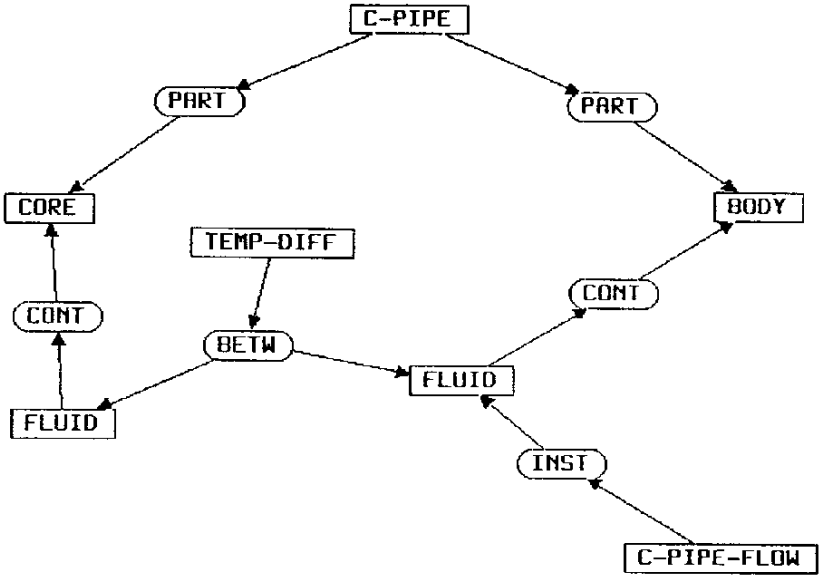


schema for C-PIPE-FLOW ("t-junction in")

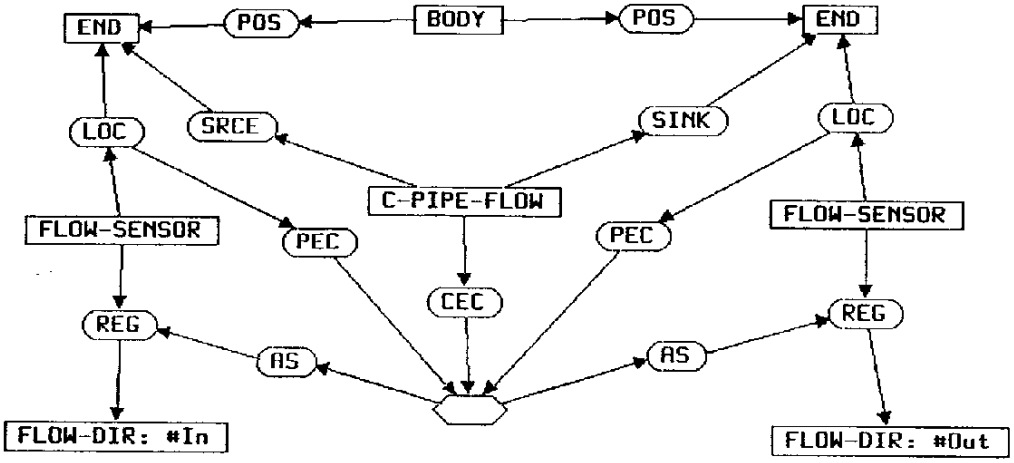


procedural overlay for C-PIPE-FLOW ("t-junction in")

Fig. l9a. Key definitions for cycle 2 of the "flow-sensor" problem.



schema for C-PIPE-FLOW ("water jacket")



procedural overlay for C-PIPE-FLOW ("water jacket")

Fig. 19b. Key definitions for cycle 2 of the "flow-sensor" problem.

Having been unsuccessful at merging the models, MGR applies its generalize operator in an attempt to remove the blocking concepts. Generalize implements a form of non-monotonic reasoning in MGR in that it removes those facts from the model context which have been brought in exclusively by the earlier cycle and which support the concept

generating the incoherence. In the example, the model is supported by the fact [BODY->(PART)->[C-PIPE]. The effects of removing this fact are then propagated through the context, all concepts being removed which are not supported by remaining facts. The effects of this procedure on the ([W-J]) context are presented in Fig. 21.

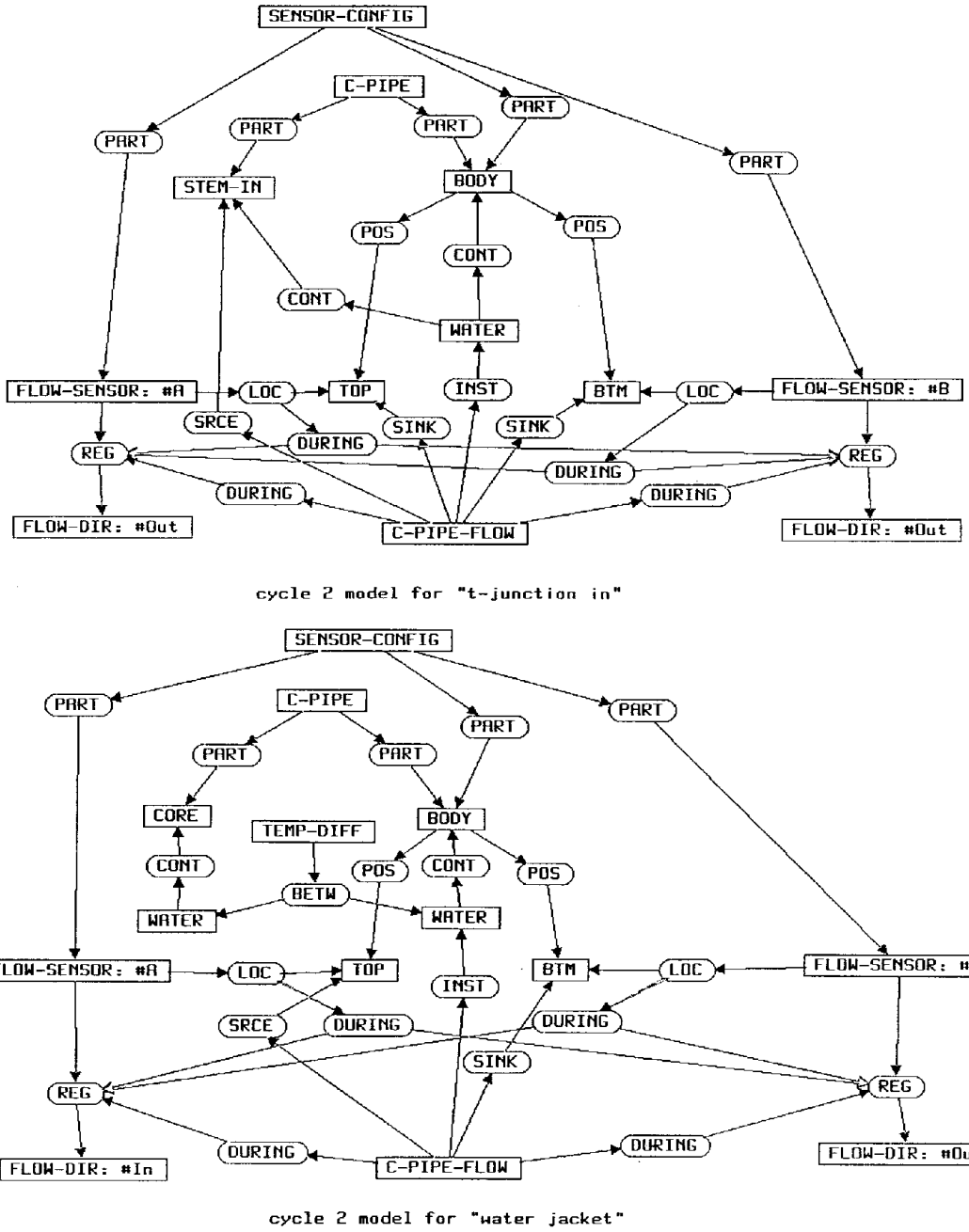


Fig. 20. Models for <[T-IN]> and <[W-J]>.

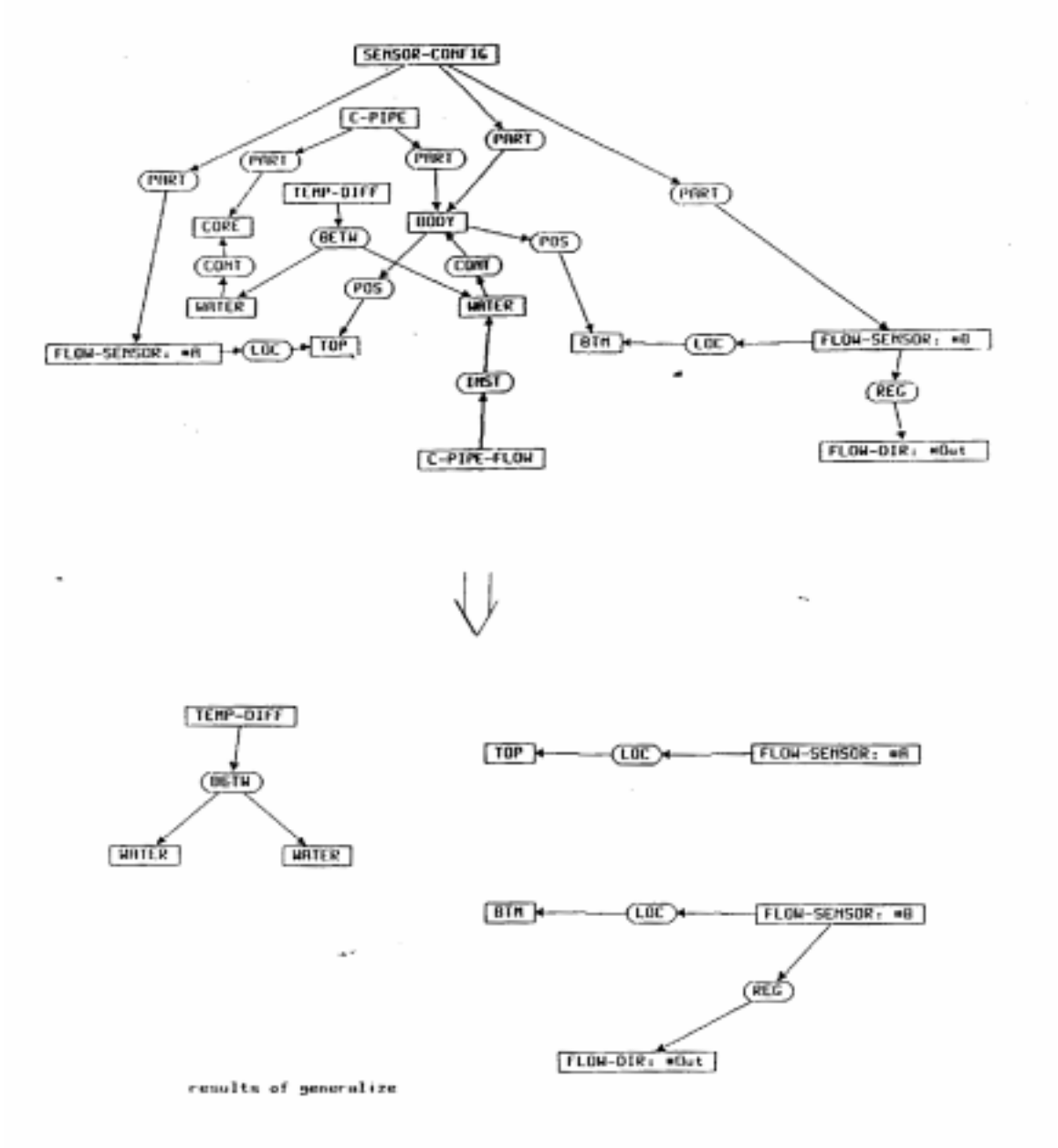


Fig. 21. Concepts remaining after the application of generalize to the ([W-J]) context.

The generalized facts now become a new set of assumptions to be submitted to interpret, the objective being to seek alternative support for remaining concepts of a form which will enable the merge. In the example, this results in a new set of covers:

Assumptions	Definitions
[TEMP-DIFF]:	([W-J]), [THERMOCLINE]
[WATER]:	[SENSOR-CONFIG], [THERMOCLINE]
[END]:	[SENSOR-CONFIG]
[TOP]:	[SENSOR-CONFIG]
[BTM]:	[SENSOR-CONFIG]
[FLOW-SENSOR]:	[SENSOR-CONFIG]
[FLOW-DIR]:	[SENSOR-CONFIG]
[PIPE-FLOW]:	[SENSOR-CONFIG]

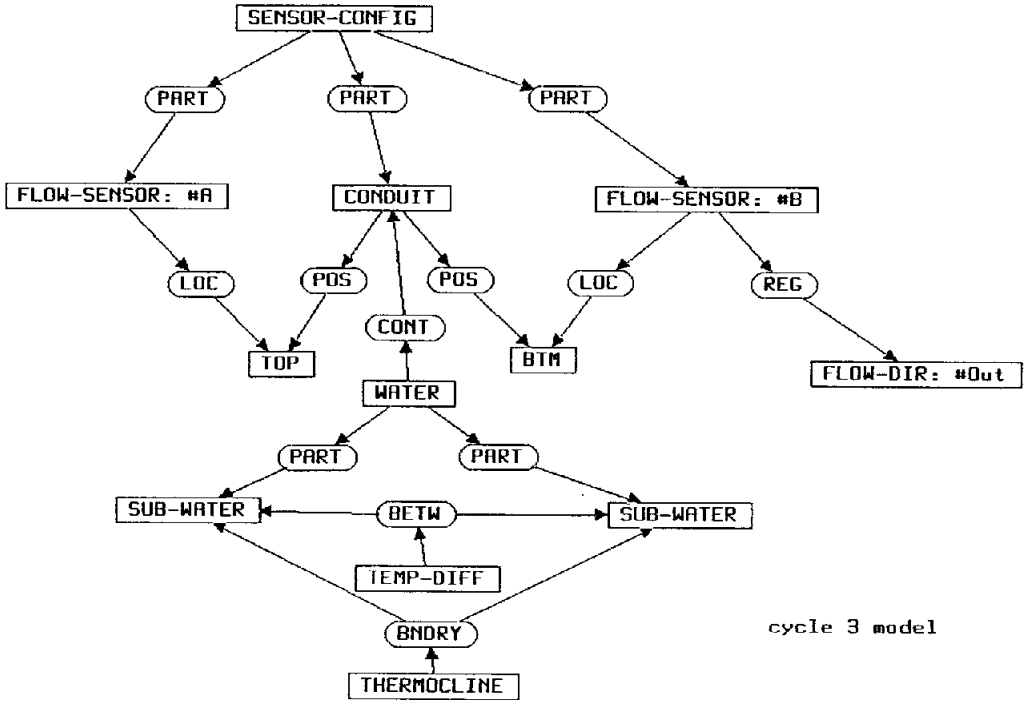


Fig. 22. Model 3 showing [WATER] with a [THERMOCLINE] but no [PIPE] specification.

There are two parsimonious covers in the above set: (a) {[SENSOR-CONFIG], <[W-J]>}; (b) {[SENSOR-CONFIG], [THERMOCLINE]}. It is the second that is taken as the context for the new model, the first having already been shown to fail.

4.5. CYCLE 4

Model 3 in Fig. 13 pictures the generalized pipe-flow structure, which is given in graphical form in Fig. 22. This close to Model 4 of the human model generative reasoning example described in section 2.3, although it is not identical, there being no indication of "flow-direction" in the present case.

On evaluation no further facts are found to join with the model. However, it is again found that query-match fails, so a further attempt is made at a merge with Model 2b. On this occasion it succeeds with the generation of Model 4 which covers all known facts and is an explanation of the query (Fig. 23).

The model represents a compound pipe ([T-IN]) with a [THERMOCLINE between two bodies of [WATER] which is contained in the [BODY] of a [C-PIPE].

5. Conclusions

This paper has presented an algorithm for automated reasoning about novel events. The research which led to the algorithm was motivated by our experience that current expert systems technology is very brittle when applied to problems which either are ill-formulated, or for which it is not possible ahead of time to determine the knowledge that will be required for solution. Designers of decision support systems for process control face both these conditions.

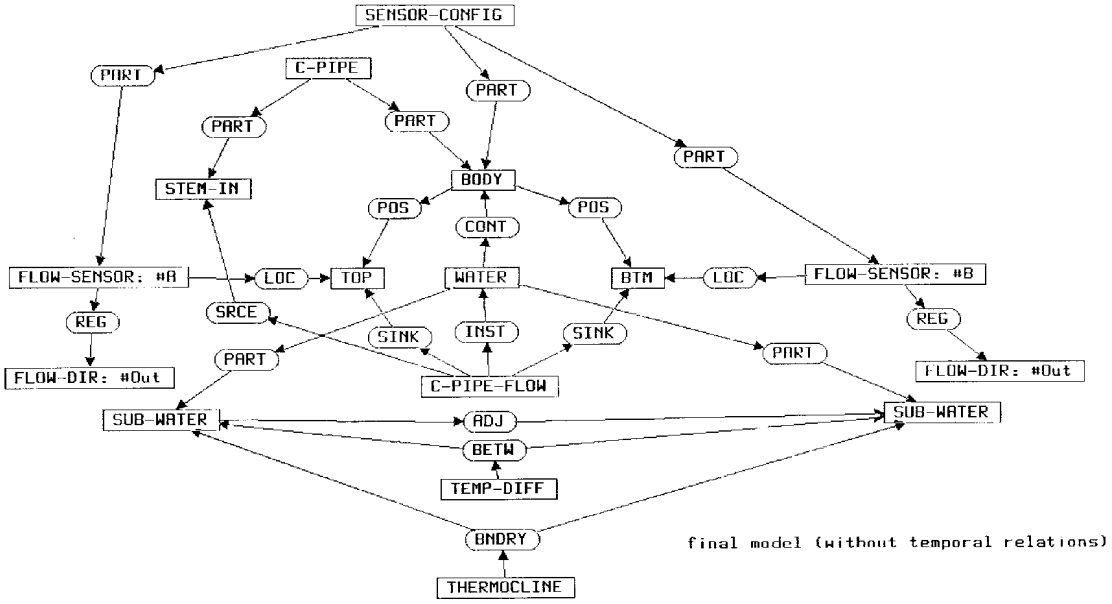


Fig. 23. Model 4 providing the final unified explanation for the "flow-sensor" problem.

The Model Generative Reasoning (MGR) algorithm replaces deductive inference with an abductive procedure based on the generation of alternative, intensional domain descriptions (models) to cover problem assumptions which are then evaluated against

domain facts as alternative explanations for queried events. MGR has been applied in its prototype form to a number of process control problems involving novel events and which we have previously found to be difficult to solve. In all these problems, the conceptual structure which represents the solution must be composed of fragments of defined knowledge objects, where the individual objects involved may be mutually incoherent (e.g. Models 2a and 2b in section 4.) The approach has proved, we believe, to be successful. MGR is able to make the most out of available background knowledge and facts, as demonstrated by its ability to construct unanticipated device structures which are coherent, support known procedures, and evaluate favorably against factual data.

In undertaking this work we have been led to consider many of the issues being explored by others in automated problem-solving. In particular, MGR overlaps with research on the various forms of truth maintenance system, most notably Assumption-based Truth Maintenance, and on qualitative reasoning.

Finally, there are three main directions to explore within MGR. First, there is little work in the literature on the decomposition and reconstruction of knowledge structures, which are clearly a significant source of the algorithm's power. Secondly, there is a set of related issues concerning the structure of explanations judged to be of value to a given problem area, their relationship to notions of conceptual coherence, and their use in controlling model generation. Thirdly, given the essentially parallel nature of the algorithm, there are issues of parallel implementa- tion. These provide topics for future work, which, from current evidence, promises to provide a better foundation for automated decision support.

6. References

ALLEN, J. F. (1983). Maintaining knowledge about temporal intervals. Communications of the ACM, 26, 832-843.

BRACHMAN, R. J. & SCHMOLZE, J. G. (1985). An overview of the KL-ONE knowledge representation system. Cognitive Science, 9, 171-216.

COOMBS, M. J. (1986). Artificial intelligence and cognitive technology: foundations and perspectives. In HOLLNAGEL, E., MANCINI, G. & WOODS, D. D. Eds, Intelligent Decision Support in Process Environments. Heidelberg: Springer-Verlag.

CHARNIAK, E. (in press). Motivation analysis, abductive unification and non-monotonic equality. Cognitive Science.

DAVIS, M. (1980). The mathematics of non-monotonic reasoning. Artificial Intelligence (Special Issue on Non-monotonic Reasoning), 13, 73-80

DE KLEER, J. (1986). An assumption-based TMS. Artificial Intelligence, 28, 127-162.

DE KLEER, J. & WILLIAMS, B. C. (1987). Diagnosing multiple faults. Artificial Intelligence, 32, 97-130.

DE KLEER, J. & BROWN, J. S. (1982). Foundations of envisioning. Proceedings of AAAI-82, 434-437.

DEAN, T. L. & McDERMOTT, D. V. (1987). Temporal data base management. Artificial Intelligence, 32, 1-55.

DOYLE, J. (1979). A truth maintenance system. Artificial Intelligence, 12, 231-272.

FORBUS, K. D. (1984). Qualitative process theory. Artificial Intelligence, 14, 86-168.

GALLANTI, M. & GUIDA, G. (1986). Intelligent decision aids for process environments: an expert systems approach. In HOLLNAGEL, E., MANCINI, G. & WOODS, D. Eds. Intelligent Decision Support in Process Environments. Heidelberg: Springer-Verlag.

HANKS, S. & McDERMOTT, D. (1986). Default reasoning, nonmonotonic logics, and the frame problem . Proceedings of AAA /- f0, Philadelphia, 328-333 .

HARTLEY, R. T. (1986). Foundations of conceptual programming. Proceedings of the 1986 Conference on Intelligent Systems and Machines, Rochester, Michigan.

HEWITT C. (19X5). The challenge of open systems. BYTE, April 1985.

KOPEC, D. & MICHIE, D. (1983). Mismatch between machine representations and human concepts: dangers and remedies. FAST Series No. Y, Commission of the European Communities.

McALLESTER, D. (19X2). An outlook on truth maintenance. Artificial Intelligence Laboratory, AIM-551, MIT, Cambridge, MA.

McCARTHY, J. (1980). Circumscription—a form of non-monotonic reasoning. Artificial Intelligence (Special Edition on Non-monotonic Reasoning), 13, 27-39.

McDERMOTT, D. & DOYLE, J. (1980). Non-monotonic logic I. Artificial Intelligence (Special Edition on Non-monotonic Reasoning), 13, 41-71.

MAIDA, A. S. & SHAPIRO, S. C. (1982). Intensional concepts in propositional semantic networks . Cognitive Science, 6, 291 -330.

MORAY, N. & MUIR, B. (1986). Intelligent aids, the theory of machines, and trust between humans and the systems they control. Paper given at NATO Workshop on Intelligent Decision Support for Process Control, Ispera, Varese, Italy, November 1986.

NAU, S. D. & REGGIA, J. A. (1986). Relationships between deductive and abductive inference in knowledge-based diagnostic problem solving. In Kerschberg, L., Ed., Expert Database Systems: Proceedings from the First International Workshop. New York: Benjamin/Cummings.

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

POPLE, H. (1982). Heuristic methods for imposing structure on ill-structured problems: the structuring of medical diagnosis. In SZOLOVITZ P. Ed., Artificial Intelligence in Medicine. Boulder, CO: Westview Press.

SCHANK, R. C. & ABELSON (1977). Scripts, Plans, Goals and Understanding. Hillsdale, NJ: Lawrence Erlbaum.

REITER, R. (1987). A theory of diagnosis from first principles. Artificial Intelligence, 32, 57-95.

REITER, R. (1980). A logic for default reasoning. Artificial Intelligence (Special Edition on Nonmonotonic Reasoning), 13, 81-132.

REGGIA, J. A., NAU, D. S. & WANG, P. Y. (1984). Diagnostic expert systems based on a set covering model. In Coombs, M. J. Ed., Developments in Expert Systems London: Academic Press.

SHAPIRO, S. C. (1979). The SNePS semantic network processing system. In FINDLER, N. V. Ed., Associative Networks: Representation and the Use of Knowledge by Computers. London: Academic Press.

RIEGER, C. (1976). An organization of knowledge for problem-solving and language comprehension. Artificial Intelligence, 7, X9-127.

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

STEFIK, M. J. (1980). Planning with Constraints. PhD dissertation, Heuristic Programming Project, Computer Science Department, Stanford University, Stanford. CA.

TOULMIN, S. (1958). The Uses of Argument. Cambridge: Cambridge University Press.

WOODS, D. D., ROTH, E. M. & HANES, L. F. (19X6). Models of cognitive behavior in nuclear power plant personnel: a feasibility study. Vol. 2: Main report. Westinghouse Electric Corporation, Pittsburgh, PA.

WOODS, D. D., WISE, J. A. & HANES, L. F. (19X2). Evaluation of safety parameter display concepts—Vol.]. NP-2239, Vol. 1, Research Project X91-5, Westinghouse Electric Corporation, Pittsburgh, PA.