An Algorithm for Open-world Reasoning using Model Generation

R.T. Hartley, M.J. Coombs, and E. Dietrich

Computing Research Laboratory
New Mexico State University
Box 3CRL, Las Cruces, NM 88003

ABSTRACT

The closed-world assumption places an unacceptable constraint on a problem-solver by imposing- an *a priori* notion of relevance on propositions in the knowledge-base. This accounts for much of the brittleness of expert systems, and their inability to model natural human reasoning in detail.

This paper presents an algorithm for an open-world problem-solver. Termed Model Generative Reasoning, we replace deductive inference with a procedure based on the generation of alternative, intensional domain descriptions (models) to cover problem input, which are then evaluated against domain facts as alternative explanations. We also give an illustration of the workings of the algorithm using concepts from process control.

1. Problem-solving Methods and Open-worlds

The predominant AI paradigm for inferencing in automated problem-solvers is that of formal syntactic reasoning. Thus, while much effort is expended in faithfully capturing the minutiae (even ideosyncracies) of an expert's world view, the automation of expert reasoning over that world is constrained by the formal requirements that i. all propositions map to true or false (or some pre-defined degree of truth or falsity), and ii. the world be closed under a defined set of inference rules. Not only are these constraints unacceptable as a framework for natural human reasoning (e.g. Johnson-Laird, 1983), but by imposing an *a priori* notion of relevance on propositions in the knowledge-base, they account for much of the brittleness of AI systems (Dietrich & Fields, 1987; Hewitt, 1985).

Although much of the power of expert human performance undoubtedly comes from simply having more facts, the most valued capability, and one AI is being increasingly required to emulate, is the ability to take decisions on novel situations where facts are incomplete or unreliable, and prior notions of relevance are likely to be incorrect. In such situations, humans have a significant ability to: a. reinterpret existing knowledge; b. progressively integrate new, often fragmentary, information with existing knowledge; c. relate associated information from different subject domains to the problem; d. utilize ambiguous (or incorrect) information; e. build models of events from a mixture of hypotheses and facts.

All of the above functions are difficult to achieve within the paradigm of formal syntactic reasoning because they are antithetical to an extensional view of entities and processes (implicit in logic and frame-based systems), and a closed-world mode of inference. They require the ability to reason about alternative interpretations of propositions, to synthesize such interpretations from term definitions, to utilize partial descriptions, and to be tolerant to possible conflicts and contradictions within interpretations. Such reasoning is strongly intensional (i.e. strongly dependent on the *non-truth functional* semantics of the descriptive terms), in addition to being essentially open-world.

In this paper we briefly present a theory of problem-solving which Supports effective automated reasoning in open-worlds. Termed Model Generative Reasoning (MGR), our approach is essentially *abductive* (Peirce, 1957), gaining its power from the generation of good hypotheses (rather than identification of the most relevant deductive inference), and from evaluating them empirically against the world (i.e. experimenting with the effects of intensionally derived models on the world). Initial ideas similar to this have been explored in diagnostic problem-solvers using some notion of set covering (e.g. Pople, 1982, Reggia et al., 1984).

In MGR we replace inference by logical proof with a procedure based on the the generation of alternative (semantically coherent) domain descriptions to cover some problem input, and their evaluation against the world as alternative problem explanations. This methodology stemmed from a decision to model human reasoning as an empirical phenomenon, rather than as an idealized, mathematical abstraction. Our MGR procedure emulates a general pattern of argument demonstrated by Toulmin (1958) to apply widely to practical reasoning. It is also echoed in much influential work on the structure of scientific thought (Kuhn, 1962; Cummins, 1983). Briefly stated (see figure 1), arguments are seen as supplying warrents, which, when evaluated, provide plausible grounds for drawing certain conclusions from a given set of facts (subject to given qualifications, or conditions).

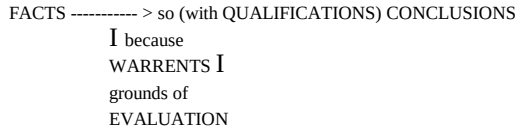

Figure 1: A pattern of practical reasoning (after Toulmin, 1958)

In MGR the above argument structure is implemented as a process by which candidate warrents *(models,* in our terminology) are constructed from intensional components (term *definitions),* and then evaluated against facts to test for extensional validity (i.e. the relevance of models is assessed in terms of their ability to explain facts). The procedure will be presented in detail in section 3. However, given the importance of the distinction between open-world and close-world reasoning to the MGR approach to inference, we first wish to make the definitions precise.

2. Definitions of Open-world and Close-world Reasoning

The most intuitive notion of closed-world reasoning can be stated in terms of inferencing. A problem-solving system is a closed-world reasoner if all of its inferences result in propositions with determinate truth-values known to the system. Intuitively then, an open-world reasoner is a system whose inferences sometimes result in propositions whose truth-value is unknown to the system and not determined by the current *intensional* state of the knowledge base.

Another way to understand the distinction is to couch it in terms of domain-dependent problem-solving, a *domain* being an organized, conceptually closed collection of knowledge on some subject (Hayes, 1985). All the sciences form individual domains, although they need not be scientifically based: driving a car and cooking with a wok are perfectly respectable domains.

A domain-dependent problem-solver is a system that makes inferences only within a certain domain, i.e., all of the input, output, and intermediate states fall within a certain, specified domain. Call such reasoning *domain- dependent reasoning. A* cross-domain problem solver is a system that opportunistically makes inferences across certain domains. Such a problem-solver can take input in domain *DI* , pass through intermediate states semantically interpreted as being in domain *D2,* and output a proposition in the original domain *DI* . The opportunism in such a System means that moving from *D 1* to *D 2* and back is left up to the system, not the implementer. (The control issues of such inter-domain searching are discussed in Dietrich & Fields, 1987; Fields & Dietrich, 1987). Call this kind of reasoning *opportunistic cross-domain reasoning* (analogical reasoning appears to be reasoning of this type, Gentner & Gentner, (1983)). Identifying a closedworld with a finite set of domains, we can now define a *closed-world reasoner* as a problem-solving

system using either domain-dependent reasoning or "cross-domain reasoning" where the "crossovers" between domains are specified ahead of time (i.e., they are not opportunistic). All expert systems are closed-world reasoners in this sense.

An *open-world reasoner,* on the other hand, is a problem solving system which can use opportunistic, cross-domain reasoning. An open-world reasoner can solve problems that would fail with a closed-world reasoner precisely because the former can use seemingly irrelevant information from other domains. There are currently no implemented open-world reasoning systems in the sense defined here, and it is to the realization of such a system that MGR is addressed (Coombs & Hartley, 1987a; Hartley, 1986).

3. The Model Generative Algorithm to Open-world Reasoning

The requirement of the MGR algorithm is to construct the most satisfactory explanations of some questioned phenomena by first building intensional models of the phenomena, and then evaluating them against known facts. Given our interest in open-world inference, we consider the intensional domains from which models are constructed to be a coherent, closed body of knowledge, and our aim is that the reasoner be able to make new inferences by arbitrarily extending the domains applied to the problem (see above). The algorithm may thus be able to generate alternative interpretations of a problem situation, which may, or may not, have any extensional value. The extensional aspects of the problem are determined empirically.

The system architecture has three components. In order to generate intensional interpretations, we have a *Definitional component.* This employs an epistemology based on schemata, after Sowa (1984), nested in a type lattice. This arrangement enables us to capture the pragmatics of domain knowledge, including both declarative and procedural aspects of a term's meaning. This permits us to compose intensional descriptions of processes and events, which can then be executed as simulations of hypothesised world activities The simulations reside in a second *Assertional component* of MGR. The third *Factual component* represents knowledge of the world in MGR.

All of these structures are represented using an extension of Sowa's conceptual graphs called Conceptual Programming, or CP (Hartley, 1986). Procedures are represented in a CP program by the integration of relevant actors into the definitions. Through actors, CP enables representation of the interaction between states and events comprising a theory of action inspired by Rieger's commonsense algorithms (Rieger, 1976).

The basic operations used for executing reasoning processes over these conceptual structures are the conceptual *join* and conceptual *projection.* Join takes two conceptual structures and unifies them by merging common concepts to form their greatest common specialization. Project abstracts common generalizations from pairs of conceptual structures in order to derive their points of discrepancy.

The algorithm starts with an initial set of assumptions and a query. The assumptions are first interpreted using declarative knowledge in the Definitional component to form (possibly multiple) **contexts.** Contexts are then completed by adding procedural definitions (procedural overlays) to form **programs.** When executed, programs result in **models.** When they have been tested against data in the Factual component models become candidate **explanations** of the problem. These candidate explanations are then evaluated against the query for acceptability. If there is no acceptable explanation, any new fact which matches a model element becomes a new, internally generated assumption, and the cycle continues. This is pictured in figure 2 below.

4. MGR through Model Integration: an example

In order to illustrate the operation of a MGR interpreter, we present an abstracted example concerned with explaining incoherent sensor readings of water flow in a pipe. This is taken from our work on diagnostic reasoning (Coombs & Hartley, 1987a) where we have found that in order to solve novel problems, not previously seen by a diagnostician, we need to use a synthetic strategy, rather than the more usual analytic methods. This provides a good illustration of the power

```
where: F = {facts), A = {assumptions), C = (contexts), P = (programs)
       M = {models},  E = {explanations}, Q = {queries), D = {definitions)
       PO = {procedural overlays}, CE = {candidate explanations}

define function: model-build(A) C ← interpret(A)
                                P← proc-overlay(C)
                                 M ← execute(P)
                                return M
define procedure MGR
   A  ←  select(F)
   M ←  model-build(A)
   loop
      CE ← evaluate(M,F)
      E ← query-match (CE,Q)
      if acceptable(E,A,Q)  then SUCCEED
      else
        if A'←  extract(E)  then
                                 A ← A + A'
                                 M ←  model-build(A)
        elseif M ←  merge(M) then continue loop
        elseif M ← generalize(M)  then continue loop
        else M ← ∅
   until empty(M)
```

Figure 2: An overview of the MGR algorithm

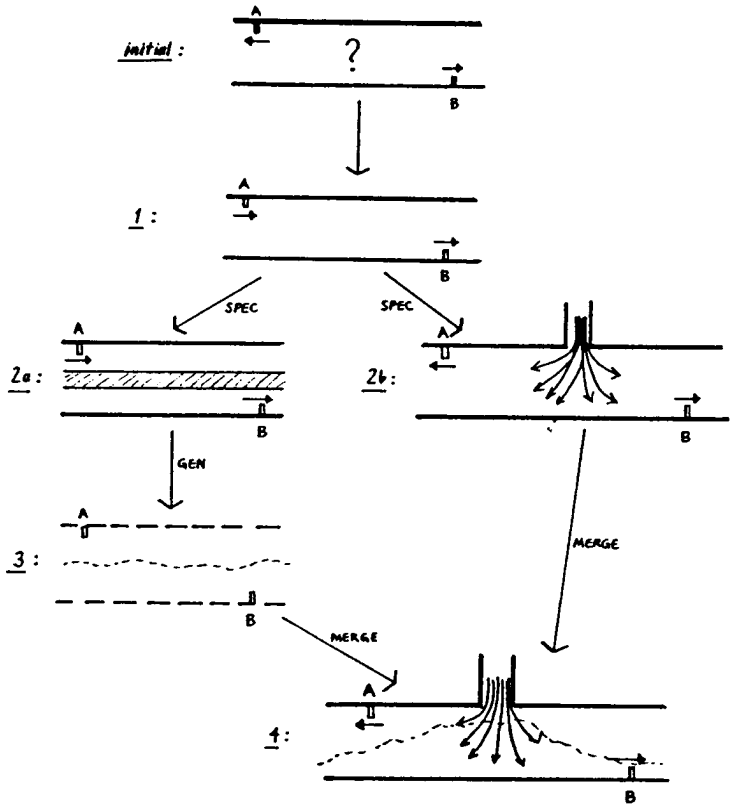

Figure 3. Explanatory sequence of the "pipe" problem.

of MGR. What follows is a summary of a much longer account of the same problem given in (Coombs and Hartley, 1987b). Figure 3 is a pictorial representation of the sequence of models produced by the MGR interpreter. The reader should refer to this diagram for better understanding of this brief account.

The MGR goal is to integrate domain information over the problem. The interpreter is seeded with an initial set of assumptions. In the present case, these are the logically minimal facts (without which there would be no query) that there is a flow-sensor, B, located at the bottom of some pipe, and that there is water in the pipe. The flow-sensor is registering the direction 'Out' meaning out of the pipe. The query is then recorded for use as a component in the evaluation of explanations. It is not, however, included in the set of assumptions because we have found that query statements often over-constrain model construction, possibly blocking unexpected solutions. In the example, We ask "Is it possible for sensor A to indicate opposing directions of flow?" i.e. also the direction 'Out' ('initial' in figure 3). The interpreter's task is to build intensional models of the domain from the assumptions, which, when evaluated against domain facts, provide candidate explanations of the queried phenomenon i.e. the sensor readings which do not appear to make sense.

We start with the definition of relevant domain entities and acts as hierarchically organized types and schemata within the Definitional component of the system. In the present example, it would be necessary to include descriptions of the concepts [PIPE-FLOW] and [SENSOR-CONFIG] (figure 4). The graphical representation of the initial assumptions and the query are given in figure 5.

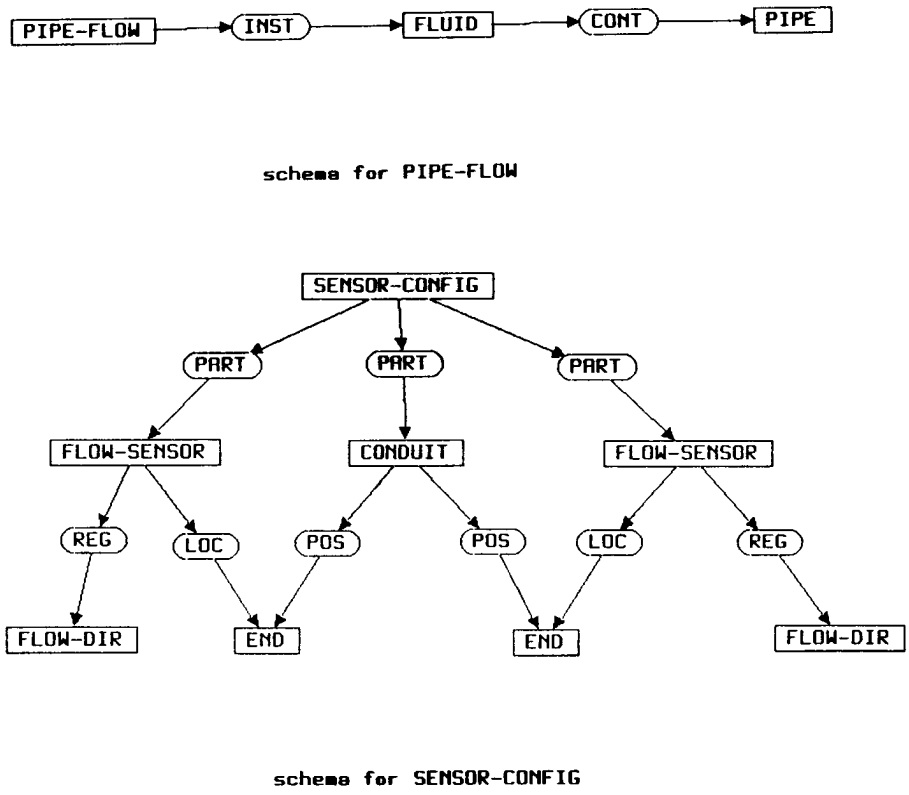

Figure 4. Declarative definitions of [PIPE-FLOW] and [SENSOR-CONFIG]

The first step is to seek interpretations of the initial set of assumptions by mapping concepts mentioned in them to Definitions. The process then continues iteratively, each new concept being expanded out by a definition where possible. The basic mapping operation is the conceptual join, two concepts of the same type (schemata also have a type designation) being represented as their greatest common specialization. Maximally joined concepts produce the **contexts** mentioned above. Different contexts can be produced because different combinations of definitions can cover

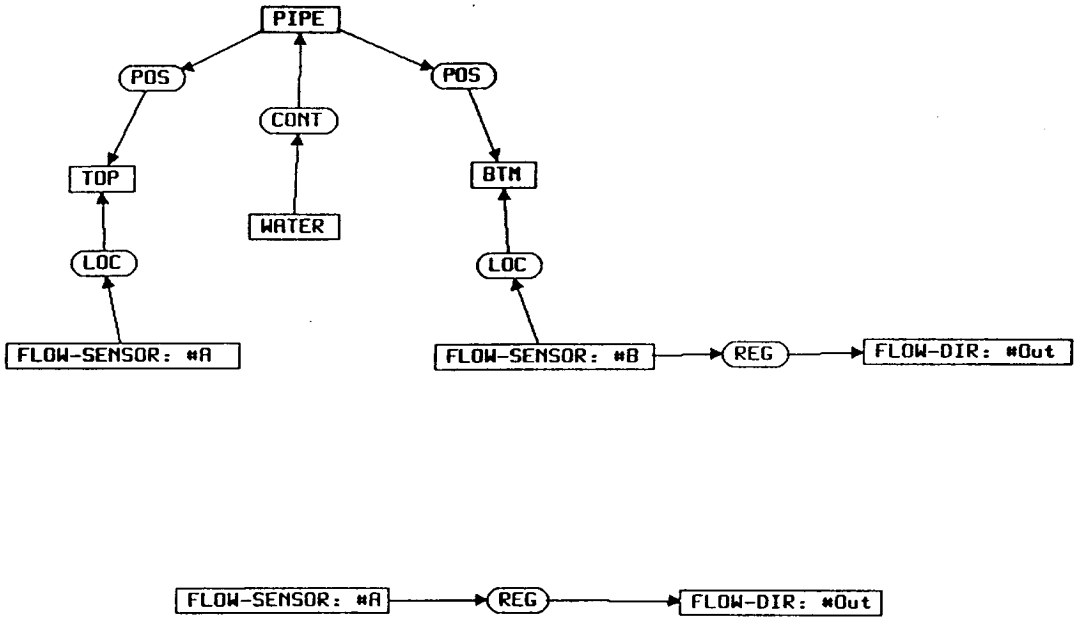

Figure 5. Initial assumptions and problem question.

the concepts contained in the assumptions. A minimal cover (in terms of the number of definitions required) follows the valuable principle of least effort. This tends to produce the simplest models first, only going on to models of greater complexity when necessary. The single context produced during this interpretation is shown in figure 6. Note the joins between assumptions and the schemata for [PIPE-FLOW] and [SENSOR-CONFIG] at [PIPE], [FLOW-SENSOR], [FLOW-DIR] and [WATER].

Procedural knowledge, derived from the procedural overlays of concepts representing ACTS, is then added to each context to produce **programs.** The single program generated during the initial attempt to solve the sensor problem is given in figure 7. The angle-bracketed boxes (nodes) represent actors which implement the procedural component of [PIPE - FLOW] related to the [PIPE] diagrammed in figure 3. Relations on the actors representing preconditions for the actor to fire include PEC (Permanent Enabling Condition) and CEC (Continuous Enabling Condition). The AS designates the state asserted (Assert State) as a result of an actor firing.

When executed, the programs create a **model** which asserts that "Flow-sensor A is located at Top", "Flow-sensor B is located at Bottom", "Flow-sensor A registers Flow-direction In" and "Flow-sensor B registers Flow-direction Out". This model is intensional and stands as an interpretation of the situation outlined in the assumptions. It is shown in figure 8 (model I in figure 3).

In addition to the declarative information, the model contains temporal relations (such as DURING) which are the result of running the program. These relations (based on those used by Allen, 1983) are fully described in (Hartley, 1987).

We now enter the evaluation phase, in which models are first presented to the Facts (figure 9). In the example, this results in the discovery of a mapping to the concept [BODY] which is a subtype of [PIPE]. However, when a join is attempted with the query a contradiction occurs concerning the [FLOW-DIR] for [FLOW-SENSOR: #A] The contradiction serves to reject the above model, while the specialization to [BODY] provides an additional assumption for the next cycle of interpretation.

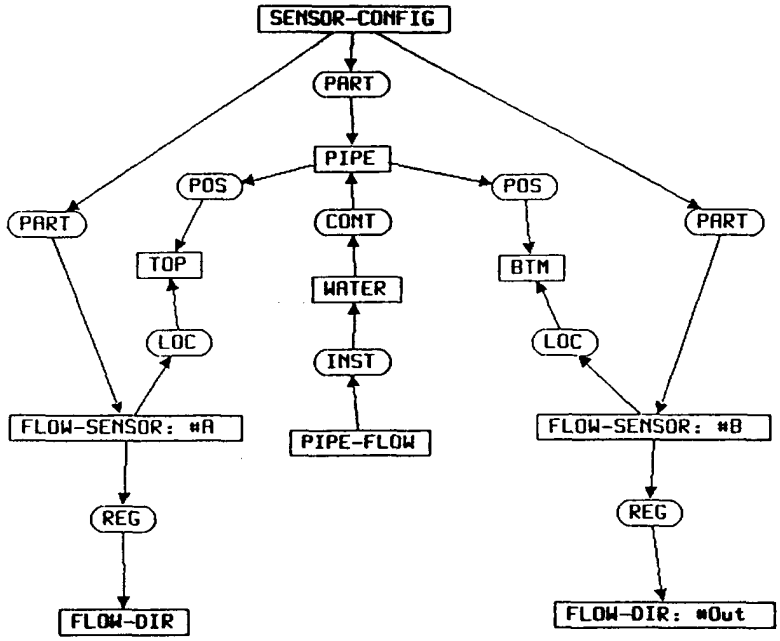

Figure 6. The context produced by joining [PIPE-FLOW] and [SENSOR-CONFIG].

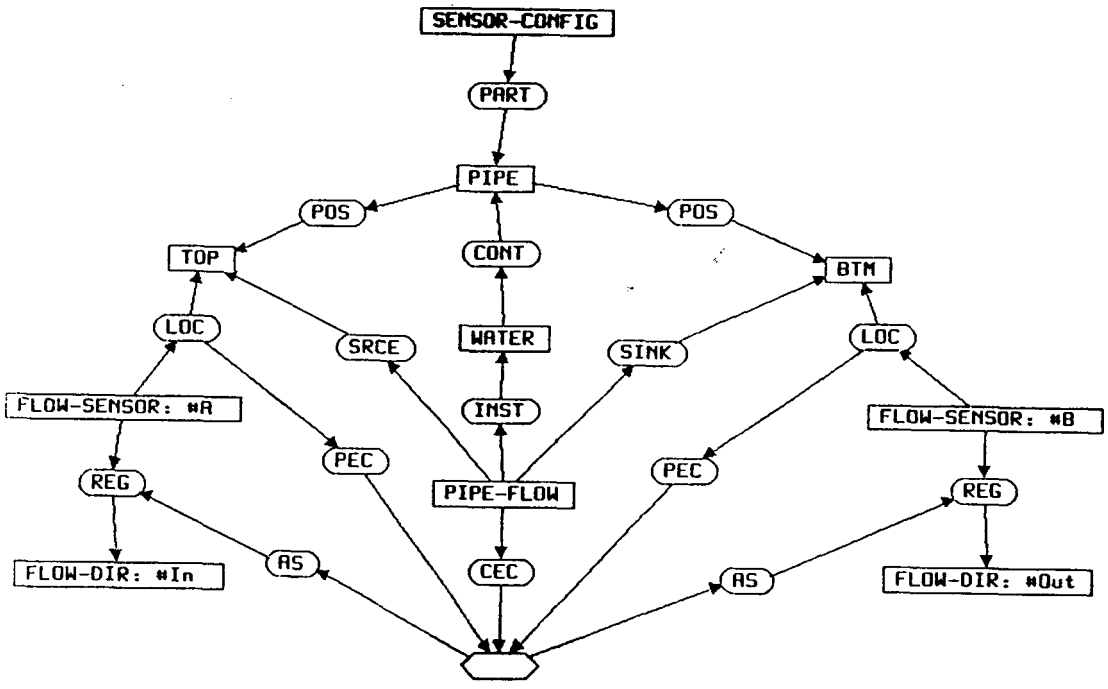

Figure 7. The program composed during the generation of model 1.

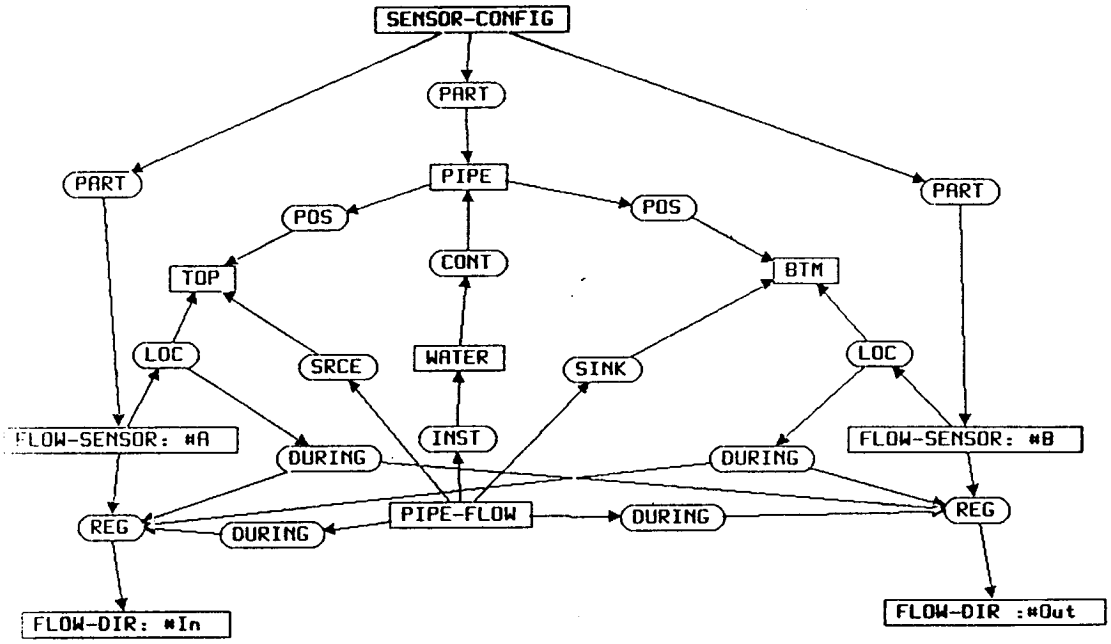

Figure 8. The model based on flow in a simple pipe.

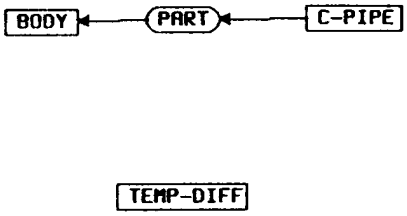

Figure 9. Domain facts.

The cycle continues with the new set of assumptions, until a structure (or structures) is generated which both explains the query and provides an acceptable cover for facts and assumptions. This is achieved after four iterations with the present problem, critical models generated during solution being illustrated in figure 3, labelled 2a and 2b.

The basic direction of model development is Specialization. For example, given the failure of the simple model generated by the first cycle, the interpreter specializes [PIPE] to the notion of a (BODY], and in so doing generates three groups of competing models. These models arise from picking up a schematic cluster concerning flow in a compound pipe, all containing [BODY]. This pipe could be a water jacket, a T-junction for inflow, or a T-junction for outflow. These three definitions have different procedural overlays, showing how the sensors can register the same direction, or in opposite directions. On evaluation, it is found that, while model 2b explains the sensor reading aspects of the query, it does not cover all the facts, namely a newly-discovered one that a temperature difference, [TEMP-DIFF] may be present. The concept [TEMP-DIFF] is brought in by the water jacket. However, model 2a, which covers this fact, does not generate

correct sensor readings.

The interpreter's response to the existence of two, partially satisfactory models, is to try to merge them. This fails on the first attempt, because it is not possible to join the definitions of simple and compound pipes. Having identified the set of concepts blocking the merge, the system cycles again with the same set of assumptions, but this time attempting to generalize away the blocking concepts. Generalize supports non-monotonic reasoning in MGR in that it removes those facts from a model's context which are exclusive to it. Generalize also removes unsupported parts of models (i.e. there was no fact to support the part). The result is model 3 (figure 3), in which there is a thermocline in a simple pipe. The definition of thermocline is brought in because of the need to explain the fact [TEMP-DIFF], previously explained by the water jacket. In fact, the thermocline was relevant at that stage, because it mentions [WATER], but its need was masked by the need to explain [BODY]. More details are in (Coombs and Hartley, 1987b). The definition of thermocline is given in figure 10.

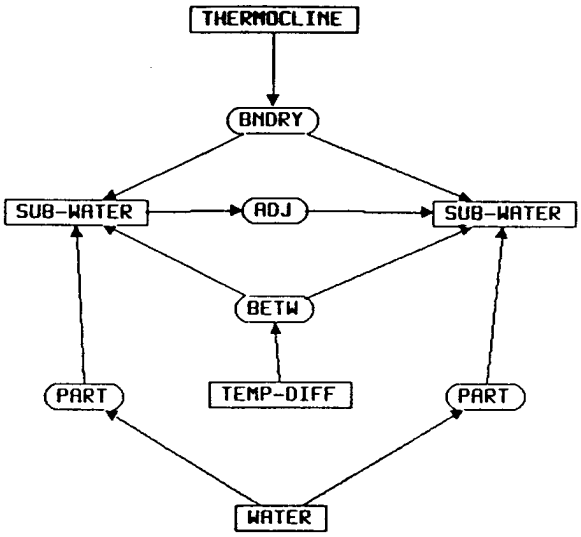

Figure 10. **The definition of [THERMOCLINE].**

The thermocline model is now evaluated against both facts and the query. This is successful as no contradiction results from matching to facts, (there are no more relevant facts), and the model explains the incoherent sensor readings. We therefore now have two competing explanations, each of which covers different facts. This drives the interpreter to attempts a further merge, which proves successful and results in the generation of model 4, figure 3. This represents the most complete explanation of the queried phenomenon, so providing strong evidence that, given the assumptions, "it is possible for the sensors to indicate opposing directions of flow". This final model is given in Figure 11.

5. Summary and Future Developments

The algorithm presented here implements the components of an open-world reasoning system. Since, the very notion of an open-world is currently a matter of debate, we have avoided programming a single reasoning strategy. Indeed, we have kept as close as possible to the general argument structure given in figure 1. We thus aim to investigate different open-world reasoning strategies using the components of MGR.

One of the main strategies under investigation in CRL is the Wanton Inference of Dietrich and Fields (1987). This strategy begins with the notion that the search for conceptual cover from

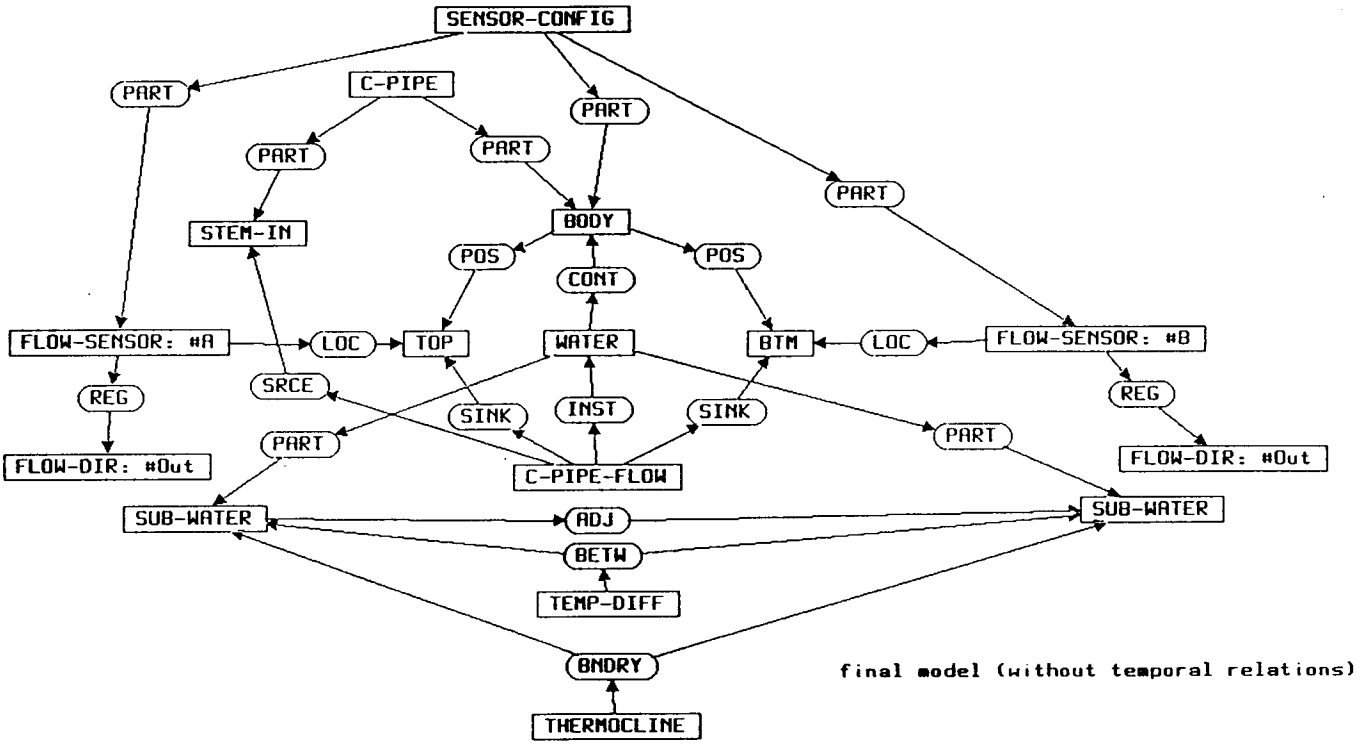

Figure 11. The final model, which merges the T-junction and thermocline models.

the definitional component is primarily stochastic.

MGR is currently being implemented on top of the CP environment in Common Lisp, which should provide a knowledge engineering system for a new breed of problem-solvers. Projects currently underway to explore the potential of MGR for real world problems include task planning for a mobile robot, exploration of scientific hypothesis generation, and co-operative problemsolving in decision support.

6. References

Coombs, M.J. & Hartley, R.T. (1987a). CP: A Programming Environment for Conceptual Interpreters. CRL Memoranda Series MCCS-87-82, New Mexico State University.

Coombs, M.J. and Hartley, R.T. (1987b). The MGR algorithm, and its application to generating explanations for novel events. Int. J. Man-Machine Studies (forthcoming)

Cummins, R. (1983). The Nature of Psychological Explanation. Cambridge,MA: MIT Press.

Dietrich, E. & Fields, C. (1987). Transition virtual machines 11: Algebraic representation of multi-domain problem solving. Technical Report # 3CR/AI-87-07.

Fields, C. & Dietrich, E. (1987). Transition virtual machines I: The TVM problem-solving architecture. Technical Report # 3CR/AI-87-06.

Gentner, D. & Gentner, D. (1983). Flowing waters or teaming crowds: mental models of electricity. In D. Gentner and A. Stevens (eds.), Mental Models. Hillsdale, NJ: Lawrence Earlbaum.

Hartley, R.T. (1986). Foundations of conceptual programming. Proc. First Rocky Mountain Conference on Artificial Intelligence, Boulder, CO., 3-15.