Explaining Novel Events in Process Control through Model Generative Reasoning

M.J. COOMBS AND R.T. HARTLEY
New Mexico State University

ABSTRACT: This article discusses the application of an abductive reasoning method termed Model Generative Reasoning (MGR) to the construction of explanations for novel events in physical systems. The MGR algorithm progressively develops intensional domain descriptions (models) to cover problem assumptions, which are then evaluated against domain facts as alternative explanations of queried phenomena. Illustrations of the workings of the MGR approach are given using concepts from process control.

KEYWORDS: Abductive reasoning; process control; Model Generative Reasoning; Truth Maintenance Systems.

INTELLIGENT DECISION SUPPORT FOR PROCESS CONTROL

Intelligent decision aids in process control mostly employ deductive expert systems technology. *(6)* As such, solutions to process control queries follow from the deductive identification of predefined process events from system data, with domain knowledge related to the events being used as inference rules. Decision support functions thus essentially rely on the diagnostic capabilities of deductive inference and the reliability of the recorded relationships between data, knowledge structures, and target events. There are two difficulties with this approach. The first is a pragmatic problem involving computational resources; the second is a deeper theoretical problem.

of target events from a predefined set of system data. Nau and Reggia [8] 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 manifestations* THEN *disorder" will* produce the set of all disorders (process events) capable of causing any of the manifestations (data items). It will, however, fail to identify situations where: (i) all minimum sets of disorders capable of causing the set of manifestations should be considered as viable, (ii) all independent minimal sets of disorders capable of causing the set of manifestations should be considered viable. These "principles of parsimonious explanation" are important in real diagnosis, where they provide simplifying assumptions of proven practical value.(11)

The solution proposed by Nau and Reggia") is to exploit the *abductive* features of diagnosis. In abduction, reasoning proceeds from hypothesized disorders to the possible manifestations caused by the disorders, rather from manifestations to logically consistent sets of disorders. The focus is thus on the construction of sets of manifestations that may be caused by given disorders. Taking this approach, one starts with a view of explanation [of which (i)-(ii) above are examples], and proceeds to form possible solutions in its mold. This potentially reduces the search required to achieve a solution.

The importance of abductive reasoning in human problem solving was first discussed by Pierce(9) in the context of scientific reasoning, but has subsequently been found to arise in many areas of practical reasoning. Within artificial intelligence abductive inference has been studied widely in areas ranging from diagnosis [5,10,11] to text understanding. [2,12]

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 either as a whole or as some simple conjunction of parts; and as such form the acid test for 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 this 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 novelty implies that appropriate predefined 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, and the system will not necessarily know the relevant classes of data to be used as axioms.

An extensional 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 nontruth functional semantics of descriptive terms, rather than upon the actual existence of knowledge objects that they represent in

the world). 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(") indicates that conceptual fictions may play an important role in helping an operator reason about a complex system, and so actual existence of a knowledge object may be less important than the relationships it exemplifies. An extensional approach would disallow such helpful functions.

REASONING BY MODEL GENERATION

At CRL, we have developed a reasoning paradigm-Model Generative Reasoning (MGR)-in order to address both the pragmatic and theoretical difficulties discussed earlier. 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 (as the conclusion) and subsets of available facts, using the most parsimonious set of linking concepts.' 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 Figure 1.

The algorithm starts with a statement of the queried phenomenon (a query) and an initial set of **assumptions.** With the basic algorithm the query will be of the form "Is phenomenon 0 *possible, given the assumptions?,"* meaning that 0 only needs to be supported eventually by one of the set of generated explanations. The initial assumptions are taken from the set of available factual information about the problem world and are selected by the user for their ability to provide the desired conceptual foundation for the query

The assumptions are used to seed model generation. They are first interpreted using general declarative definitions of domain entities to form, possibly multiple, structural descriptions, which we term **contexts.** Contexts are then completed by the addition of procedural information related to declarative structures (procedural overlays) to form **programs.** Finally, programs are executed to produce models. Models represent hypothetical world descriptions, which have not yet been evaluated against any factual information about "states-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. Pairs of concepts are judged to be "incoherent" where they both cannot be true of a designated individual or 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."

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

Figure 1: The Basic MGR Algorithm

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

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 return SUCCEED    ;and 'acceptable'
    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 = null
  until empty (M)
  return FAIL
```

coherent subsets of current factual knowledge. Where a fact is found to be conceptually coherent with a model, 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, 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 implements a form of nonmonotonic reasoning in MGR closely related to de Kleer's ATMS.(4) The cycle is then repeated with the objective of generating a single model which covers, and thus explains, all of the assumptions. This may, of course, not be possible, in which case the cycle continues until either all assumptions are covered by the minimal number of explanations or some user-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).

SYSTEM ARCHITECTURE AND KNOWLEDGE REPRESENTATION IN MGR

The system architecture for MGR has three components. In order to generate intensional interpretations, we have a **Definitional Component.** This employs an epistemology based on schemata, after Sowa [13], 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 hypothesized world activities. The simulations reside in a second **Assertional Component of MGR.** The third **Factual Component** represents knowledge of the world in MGR.

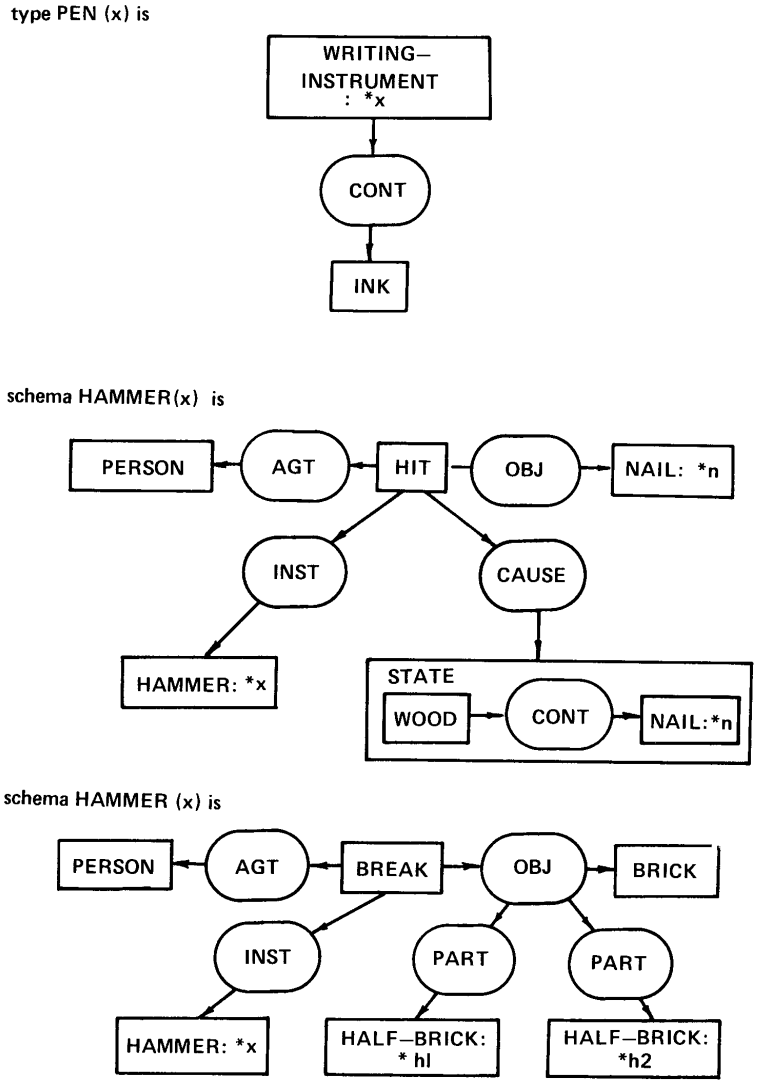
In his book *Conceptual Structures,* John Sowa describes a knowledge representation scheme as a means of expression of a theory of cognitive functioning. The atoms of the scheme are concepts and conceptual relations which are arranged into *conceptual graphs* according to theoretical ideas about language and the way the world works. The graphs are formed according to a set of formation rules which govern their ultimate structure.

Presenting these rules axiomatically, Sowa gives the *canon* (graphs formed according to the rules are *canonical).* Its parts are:

I . A type hierarchy which relates concepts according to the principles of generalization and specialization;
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; and
5. A set of formation rules with which new canonical graphs can be derived from existing ones, the main rule being the *conceptual join.* This rule takes two conceptual structures and unifies them by merging common concepts to form their greatest common specialization.

Graphs acquire meaning through the notion of canonicality, and through definitional mechanisms for new concept types. Concept definitions can be formulated as *Aristotelian types* by specifying a set of necessary and sufficient conditions, or

Figure 2: Example Type and Schema Definitions



as *schemata* by expressing alternative contexts in which a term can gain meaning. When a concept is defined using schemata, the set of schemata are named using an empty type label, which is then given a position in the type hierarchy. Examples of type and schema definitions for two common objects are given in Figure 2.

Conceptual graphs, as briefly described above, express declarative knowledge. An *assertional* mechanism built using such graphs would allow a collection of graphs to represent a state of knowledge of an agent. If rules governing the assertion and de-assertion of graphs can be expressed within the Definitional Component, then procedural knowledge can also be incorporated. Sowa has actors behaving like functions embedded within a graph to provide concept referents (i.e., for instan- We have extended this representation to allow actors to be much more like "active concepts" which accept states as preconditions and events as triggers. Having been triggered, they 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. They operate, when their inputs are completely determined, by supplying referents for their output concepts (see Figure 3b).

All of MGR's knowledge structures are conceptual graphs and are represented using a specialized knowledge engineering environment named CP (Conceptual Programming) running on our Symbolics systems.")

GENERATING EXPLANATIONS THROUGH MGR: AN EXAMPLE

In order to illustrate the operation of a MGR interpreter, we present an abstracted example concerned with explaining an incoherent sensor reading of water flow in a pipe. This is taken from our work on diagnostic reasoning, where we have found that in order to solve novel problems, we need to use a synthetic strategy, rather than the more usual analytic methods. This provides a good illustration of the power of MGR. What follows is a summary of a much longer account of the same problem given in Coombs and Hartley .(3) Figure 4 is a pictorial representation of the sequence of models generated by the current implementation of the MGR interpreter. The reader should refer to this diagram for a better understanding of this brief account. The hierarchy of concept types and embedded schema (given in angle brackets) used in the example is given in Figure 5.

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 jointly describe a situation in which there exists a [PIPE] which: (1) contains
[WATER]; (2) has a market position [TOP]; (3) has a marked position [BTM] (4) has a [FLOW-SENSOR: #A] located at [TOP]; (5) has a [FLOW-SENSOR: #B] located at [BTM]; and (6) has a reading of [FLOW-DIR:#Out] for [FLOW-SEN- It may be recalled that assumptions have the status of facts in 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 may thus need to be removed at a later stage.

The query is recorded separately for use as a component in the evaluation of explanations. It is not included in the set of assumptions, because we have found

Figure 3a: An Example of an Actor

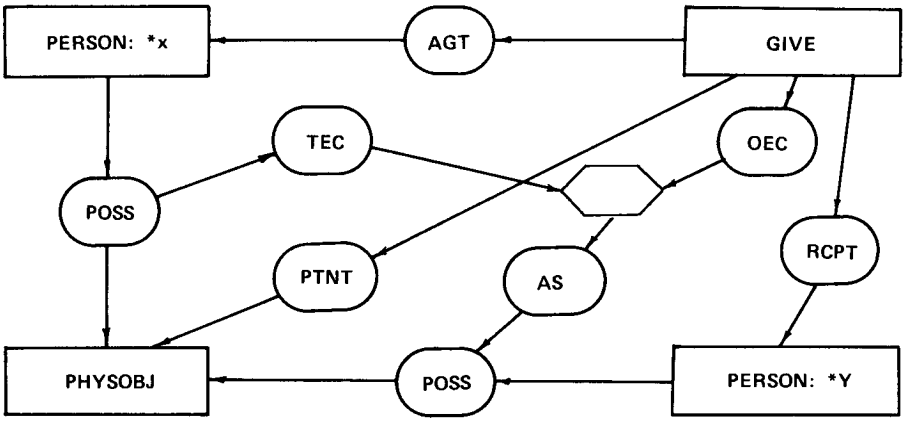


Figure 3b: Actor Inputs and Outputs

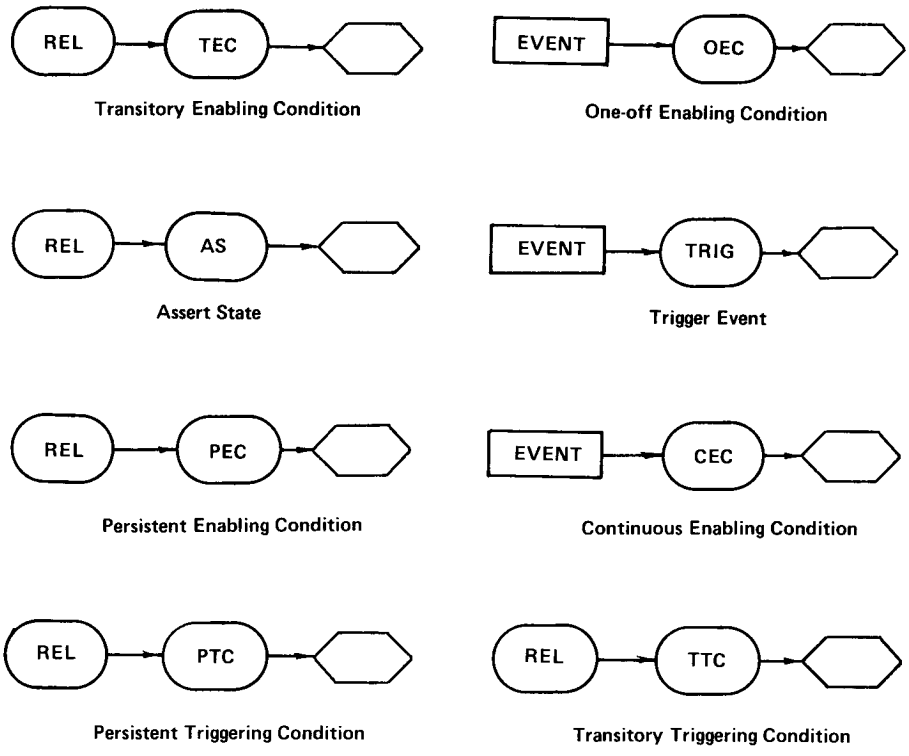
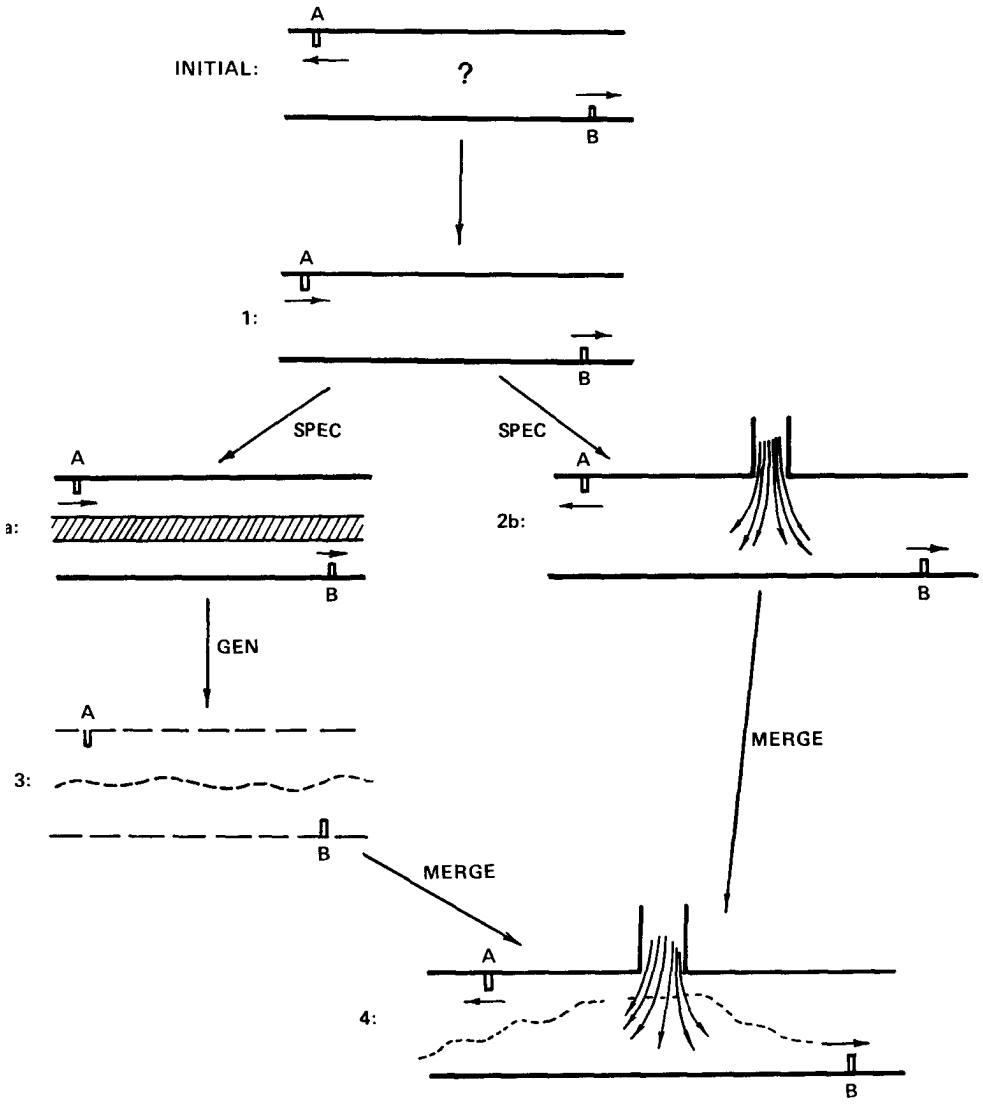


Figure 4: Explanatory Sequence for the "Flow-sensor" Problem



that query statements often over-constrain model construction, possibly blocking unexpected solutions. In the example, we ask "Is is possible for sensor A to indicate an opposing direction of flow?"; i.e., also the direction #Out ('initial' in Figure 2). 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 reading which does 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.

Figure 5: Type Hierarchy for the "Flow-sensor" Problem

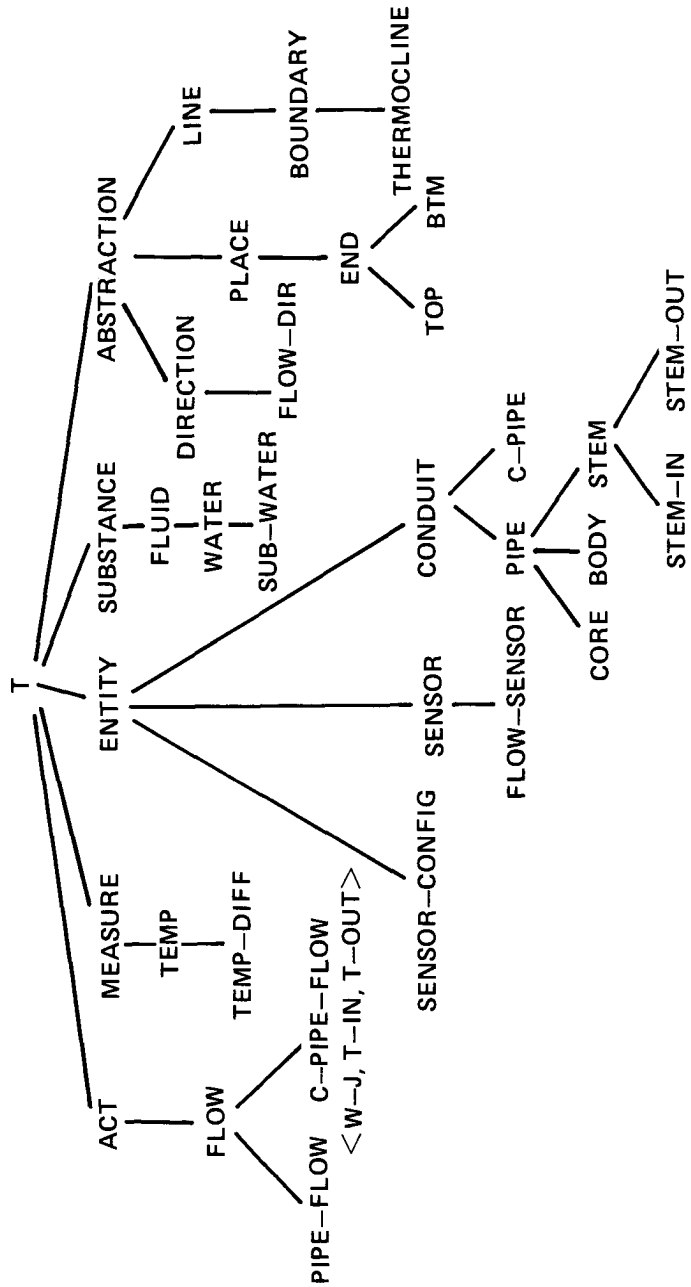


Figure 6: Declarative Definitions of [PIPE-FLOW]

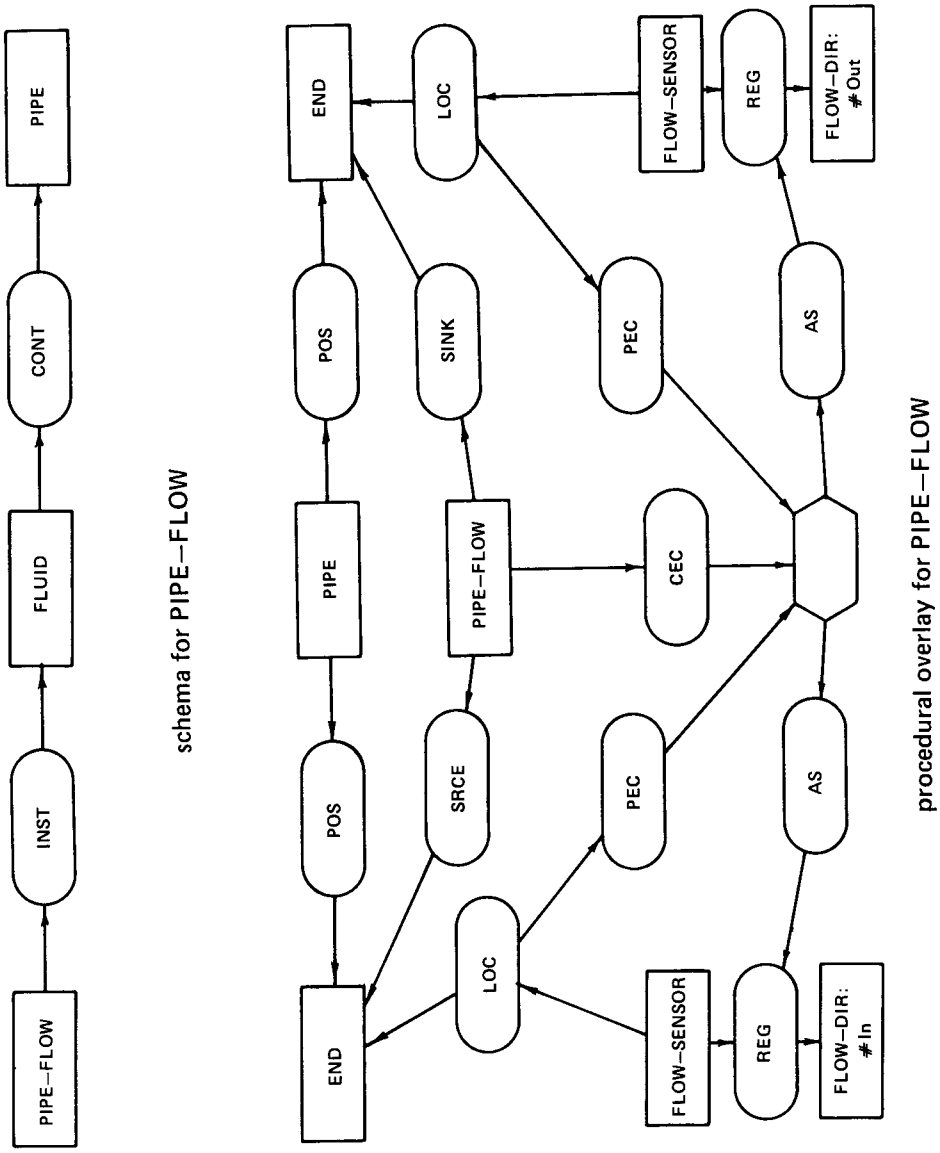
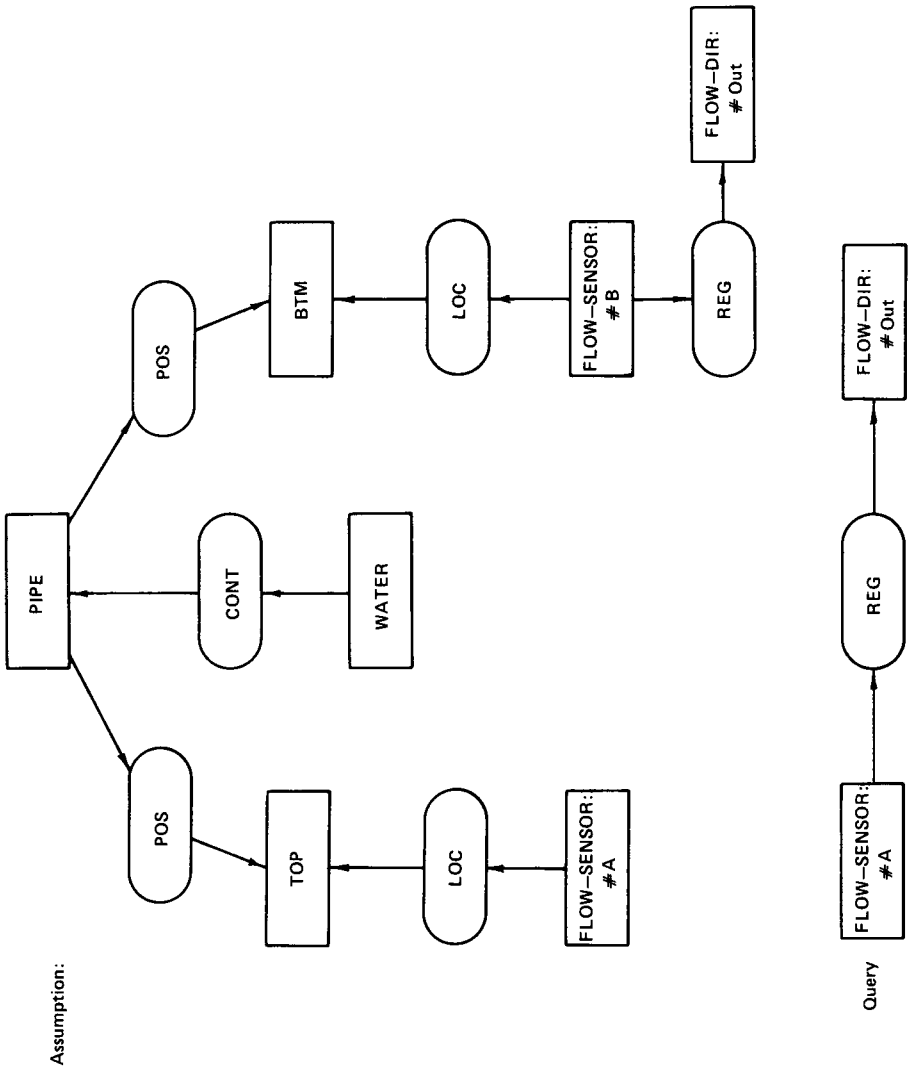


Figure 7: Initial Assumption and Query



In the present example, it would be necessary to include descriptions of the concepts [PIPE-FLOW] and [SENSOR-CONFIG] (see Figure 6 for their graphical representation and Figure 5 for their location in the type hierarchy). The graphical representation of the initial assumptions and the query are given in Figure 7.

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 the concepts contained in the assumptions. A minimal cover (in terms of the number of definitions required) follows the valuable principle of explanatory parsimony. 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 8. Note the joins between assumptions and the schemata for [PIPE-FLOW] and [SENSOR-CONFIG] at [PIPE], [FLOW-SENSOR] and [WATER].

Procedural knowledge, derived from the procedural overlays of concepts with the type ACTS, is then added to produce **programs.** The single program generated during the initial attempt to solve the flow-sensor problem is given in Figure 9. The angle-bracketed boxes (nodes) represent actors which implement the procedural aspects of [PIPE-FLOW] related to the [PIPE] diagrammed in Figure 6. Relations on the actors representing preconditions for the actor to fire include PEC (Persistent Enabling Condition-indicating a relation that persists after the firing of the actor) and CEC (Continuous Enabling Condition-an event that is required to be continuous for the actor to fire). The AS designates the state asserted (Assert State) as a result of an actor firing.

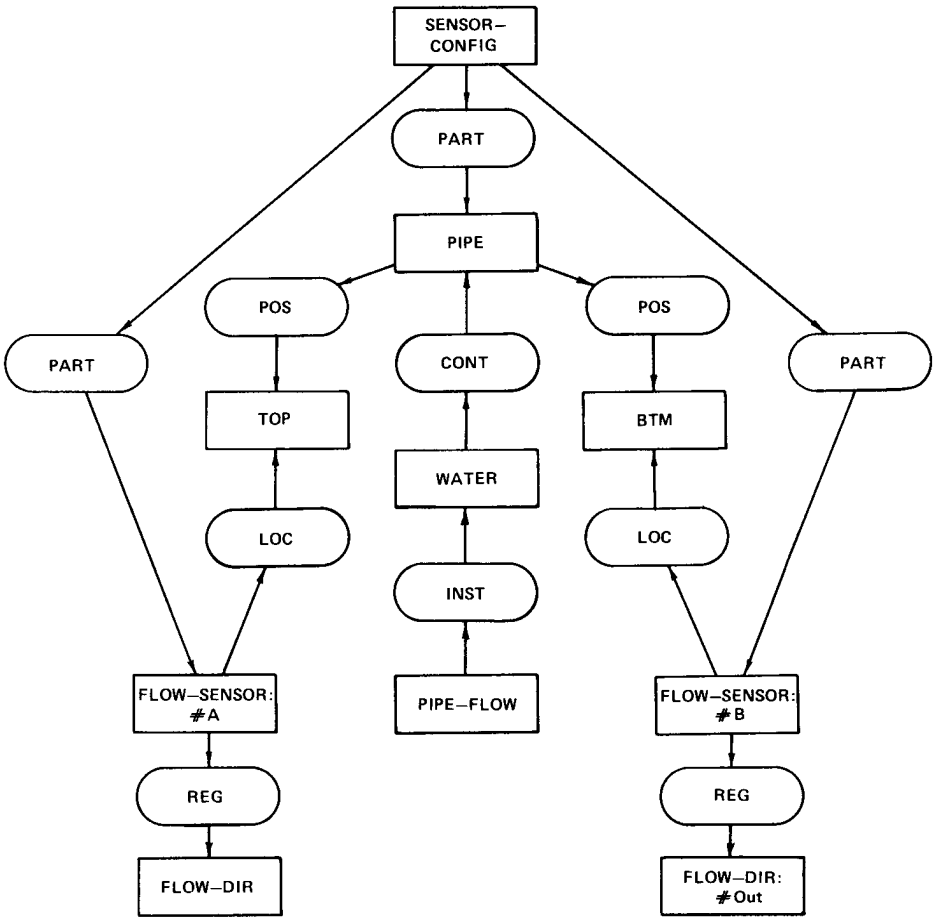
When executed, the programs create a **model** which asserts that "Sensor A is located at Top," "Sensor B is located at Bottom," "Sensor A registers Flowdirection In" and "Sensor B registers Flow-direction Out." This model is intensional and stands as an interpretation of the situation outlined in the assumptions. It is shown in Figure 10 (Model I in Figure 4).

In addition to the declarative information, the model contains temporal relations (such as DURING) which are inserted as a result of running the program. These relations, based on those used by Allen(1) are further described in Coombs and Hartley. (3)

We now enter the evaluation phase, in which models are first tested against the Facts (Figure 11). 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 incoherence occurs concerning the [FLOW-DIR] for [FLOW- The incoherence serves to reject the above model, while the specialization to the concept [BODY] provides an additional assumption for the next cycle of interpretation.

The cycle continues with the new set of assumptions, until one or more structures are generated which both explain the query and provide an acceptable cover for

Figure 8: The Context Produced by Joining [PIPE-FLOW] and [SENSOR-CONFIG]



facts and assumptions. This is achieved after four iterations with the present problem. However, the critical models upon which the final solution is based are generated in the second cycle and are illustrated in Figure 12. These describe water flow in two forms of compound pipe: (1) a "water-jacket," where an inner pipe contains water which is cooled by water flowing in an outer pipe (la in Figure 4); (2) a "T-junction," where the stem inputs water to the body of the pipe (2b in Figure 4).

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 the concept [PIPE] to the notion of a pipe which functions as a [BODY], and in so doing generates two competing models. These models arise from picking up a schematic cluster concerning flow in a compound pipe, all containing [BODY]. This pipe could be the schemata water jacket-<[W-J]>, or input T-junction-- These two definitions have different procedural overlays, showing how

Figure 9: A Program Fragment Composed During the Generation of Model 1

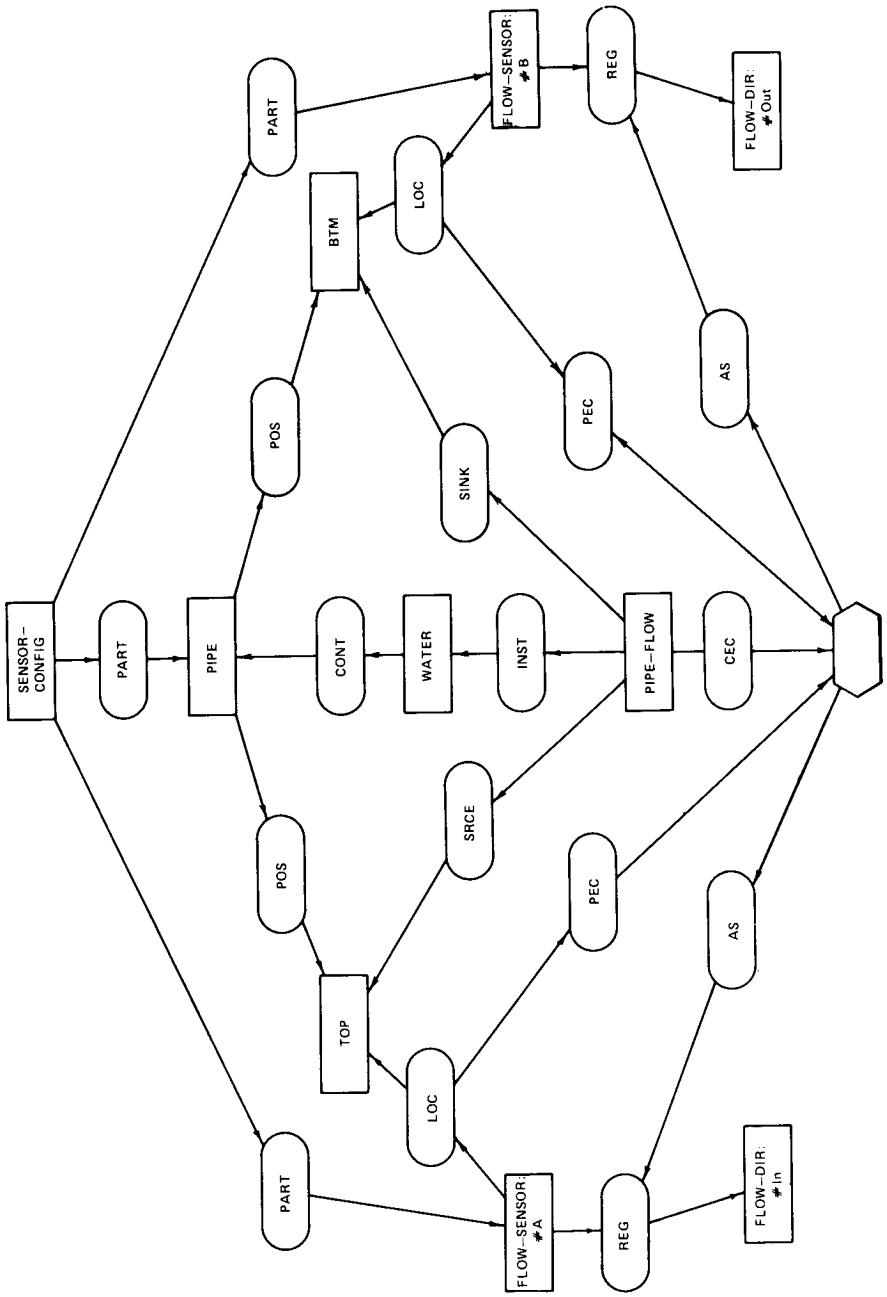


Figure 10: Model I With a Simple [PIPE] and [PIPE-FLOW]

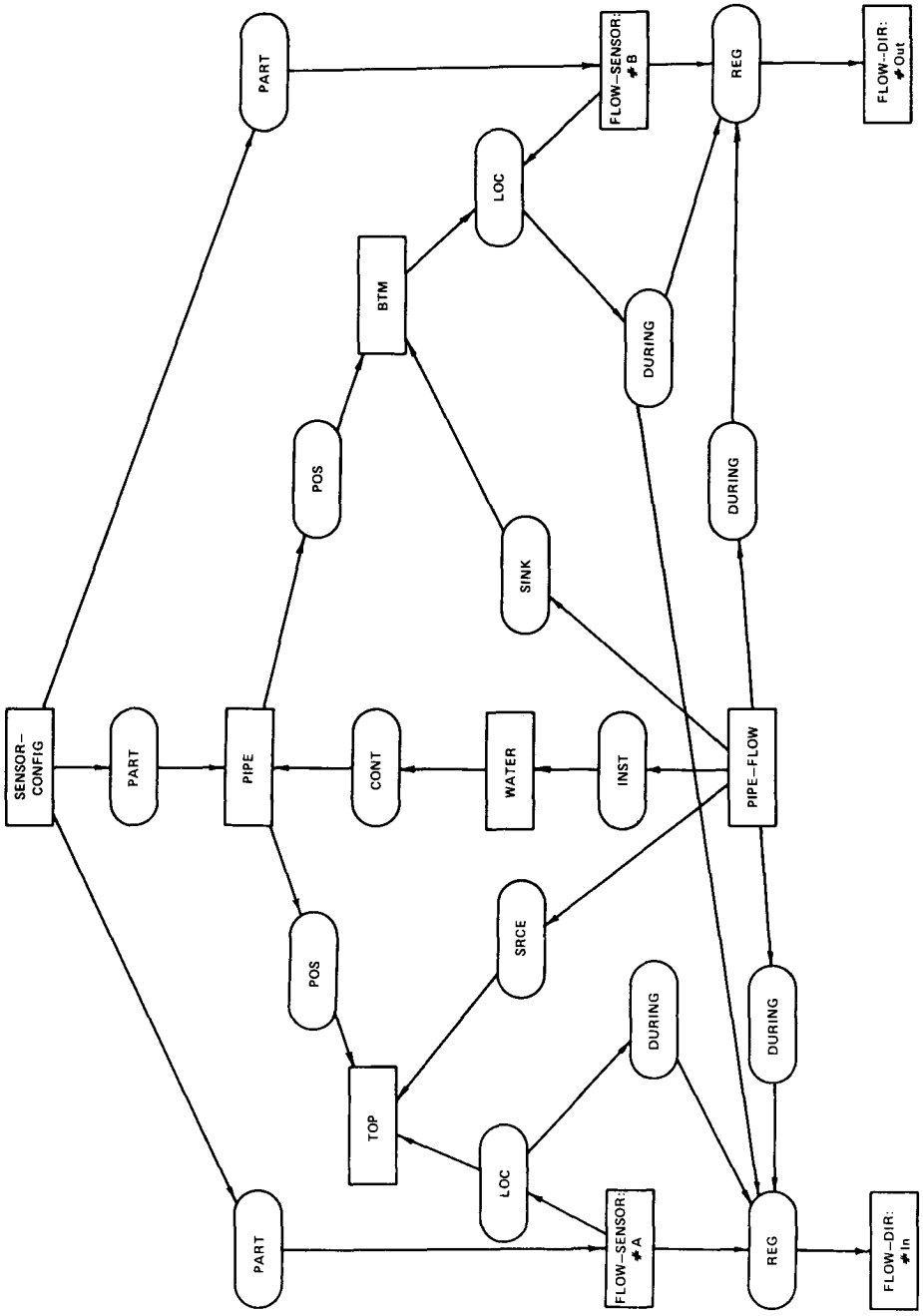


Figure 11: Domain Facts

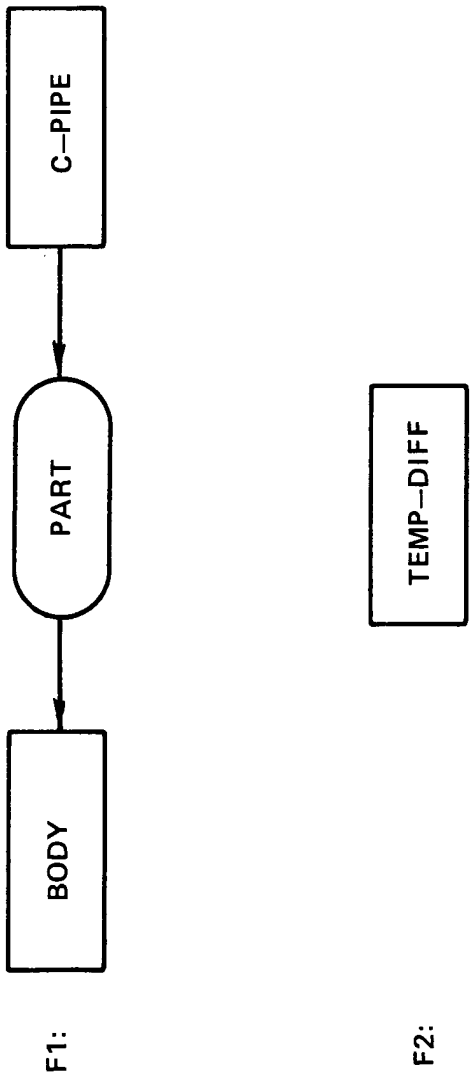


Figure 12: Model 2a <WJ> and Model 2b <T-IN> Generated During the Second Interpretation cycle

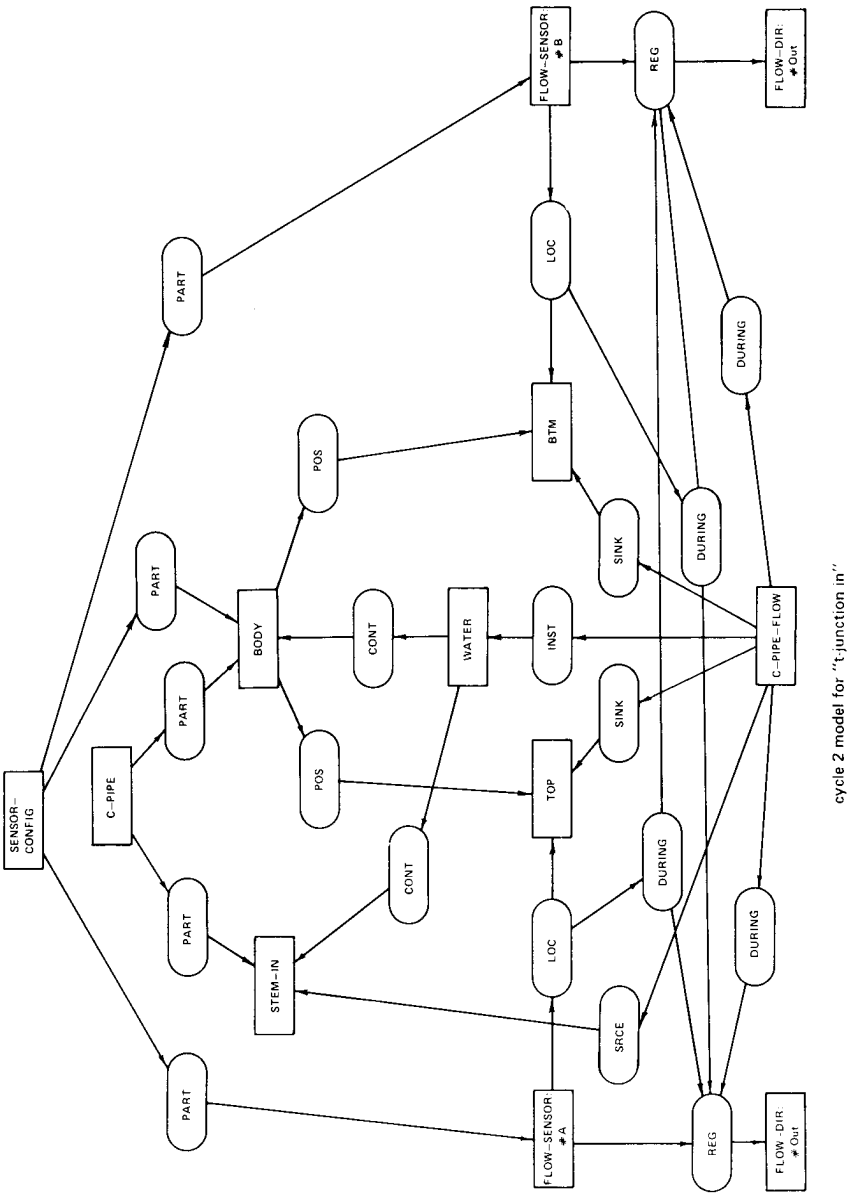
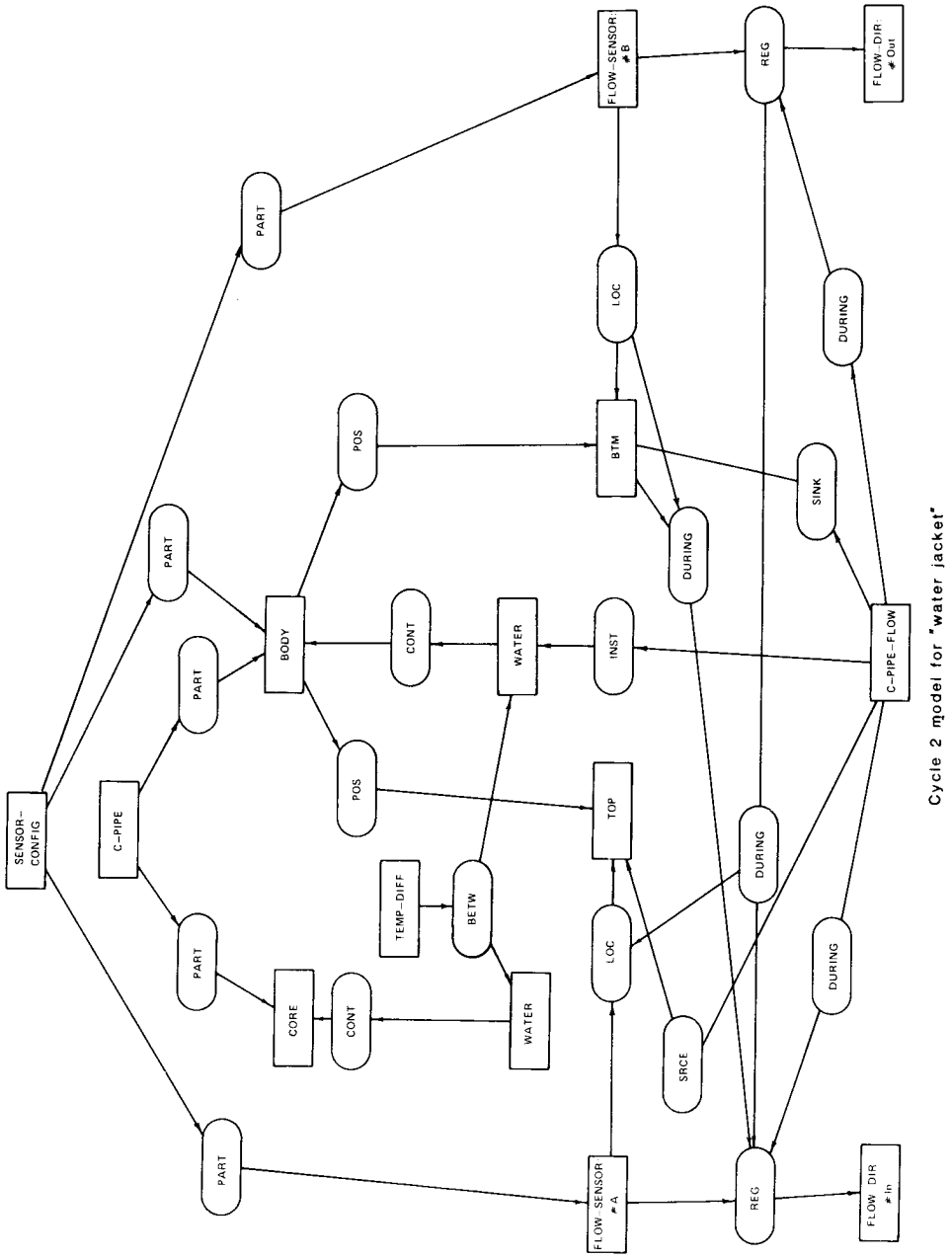
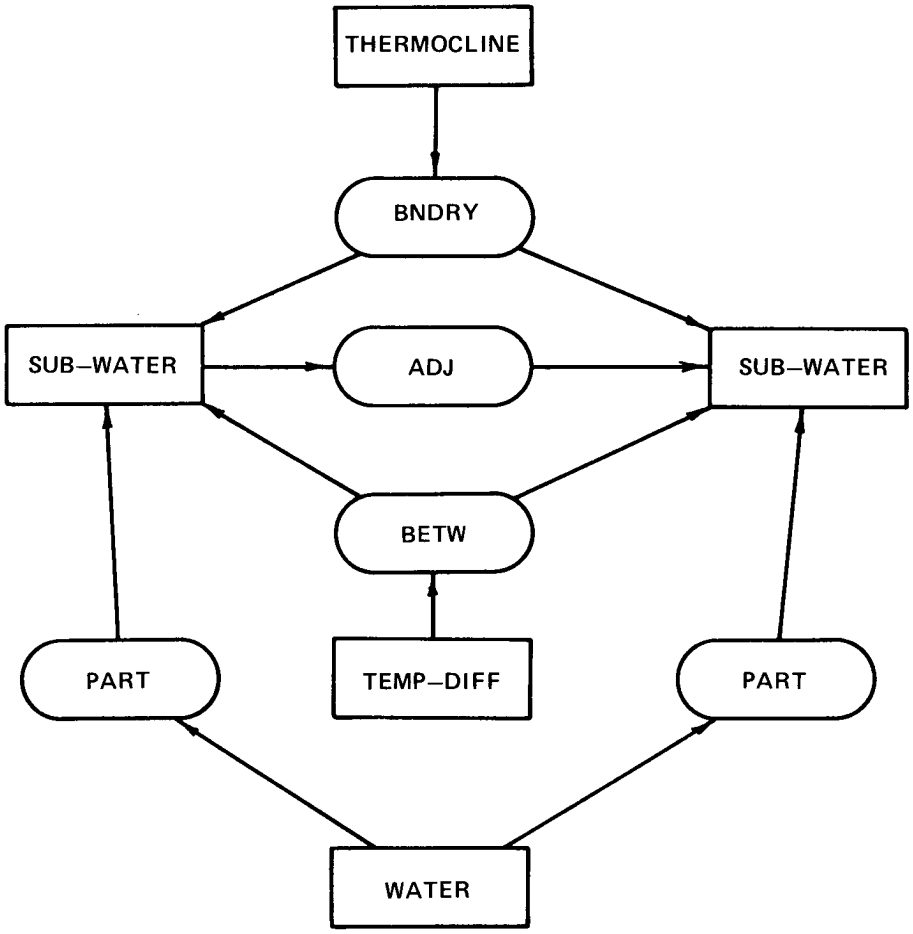


Figure 12: Continued



Cycle 2 model for "water jacket"

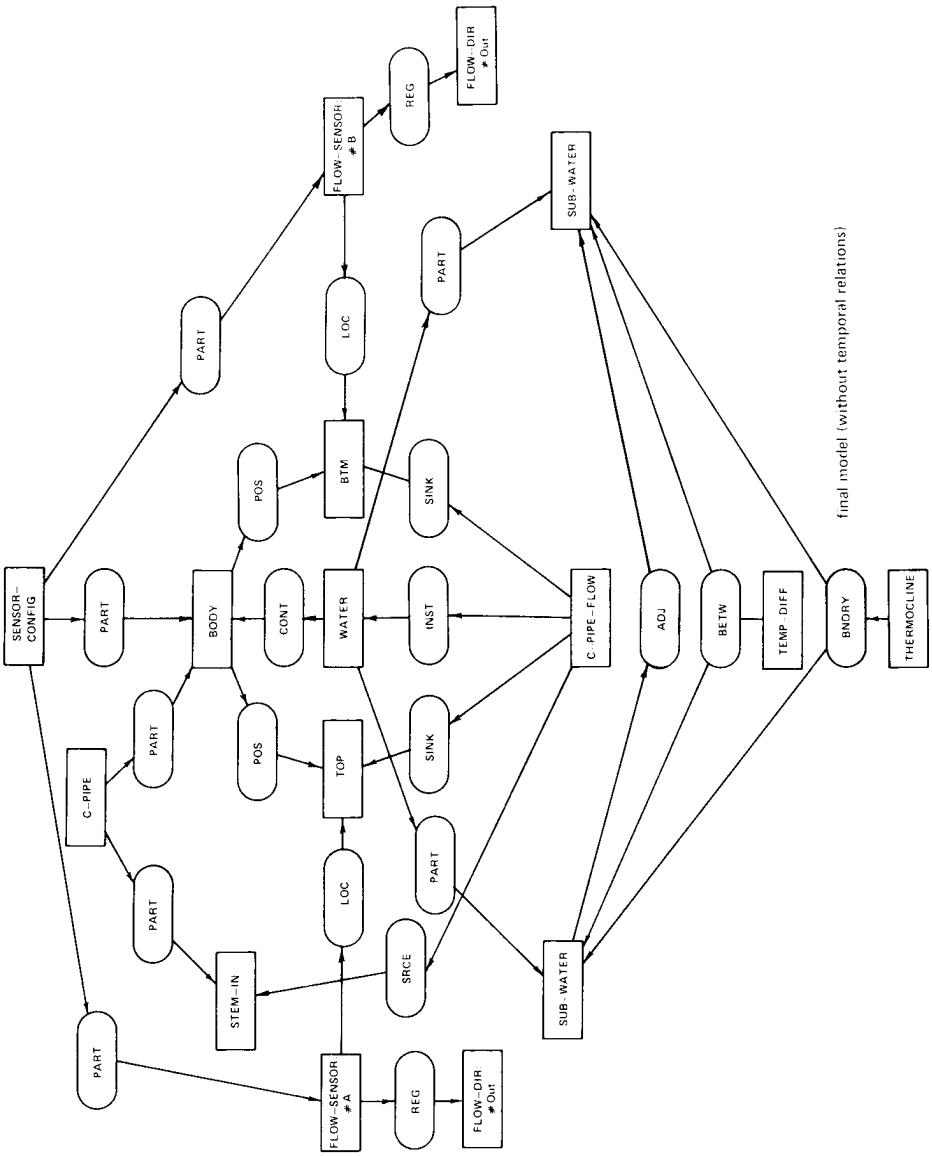
Figure 13: The Definition of [THERMOCLINE]



the sensors can register the same direction, or 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 the newly-discovered one that a temperature difference, [TEMP-DIFF], may be present. The concept [TEMP-DIFF] is brought in by <[W-J]>. 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 nonmonotonic 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 is no fact to support the part). The result

Figure 14: The Final Model, Which Merges The <[T-IN]> and [THERMOCLINE] Models



is Model 3 (Figure 4), 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- previously explained by <[W-J]>. Indeed, 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. (3) The definition of [THERMOCLINE] is given in Figure 13.

The [THERMOCLINE] model is now evaluated against both facts and the query. This is successful, as no contradiction results from matching to facts, and the model explains the incoherent sensor reading. Therefore, we now have two competing explanations, each of which covers different facts. This drives the interpreter to attempt a further merge, which proves successful and results in the generation of Model 4 (Figure 4). 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." The final model is given in Figure 14.

SUMMARY AND FUTURE DEVELOPMENTS

This article has presented an algorithm for automated reasoning about novel events. 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 which we have previously found to be difficult to solve. In all these problems, the conceptual structure which represents the solution must be composed from fragments of defined knowledge objects, where the individual objects involved may be mutually incoherent (e.g., Models 2a and 2b). 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. Second, 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. Third, given the essentially parallel nature of the algorithm, there are issues of parallel implementation. These provide topics for future work, which, from current evidence, promises to provide a better foundation for automated decision support.

ACKNOWLEDGEMENTS: This work has been supported primarily 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 # Wand Sandia National Laboratories (Grant # 95-3951).

NOTE

1. The parsimony principle currently used in MGR is to take "all minimal sets of concepts from definitions capable of jointly covering all of the concepts in the assumptions."

REFERENCES

(1) Allen, J. F. "Maintaining Knowledge About Temporal Intervals." *Communications of the ACM (1983) 26:832-843.*

(2) Charniak, E. "Motivation Analysis, Abductive Unification and Non-Monotonic Equality." Artificial Intelligence *34:275-295.*

(3) Coombs, M.J., & R. T. Hartley. "The MGR Algorithm and Its Application to the Generation of Explanations for Novel Events." *International Journal of Man-Machine Studies 27:1-30.*

(4) de Kleer J. "An Assumption-Based TMS." *Artificial Intelligence (1986) 28:127-162.*

(5) de Kleer, J., & B.C. Williams. "Diagnosing Multiple Faults." *Artificial Intelligence (1987) 32:97-130.*

(6) Gallanti, M., & G. Guida. "Intelligent Decision Aids for Process Environments: An Expert Systems Approach," edited by E. Hollnagel, G. Mancini, & D. D. Woods. *Intelligent Decision Support in Process Environments* (Heidelberg, W. Germany: Sprin- *1986).*

(7) Hartley, R. T. "Foundations of Conceptual Programming." *Proceedings of the 1986 Conference on Intelligent Systems and Machines* (Rochester, Michigan, *1986).*

(8) Nau, S.D., & J. A. Reggia. "Relationships Between Deductive and Abductive Inference in Knowledge-Based Diagnostic Problem Solving," edited by L. Kerschberg. *Expert Database Systems: Proceedings from the First International Workshop* (New York: Benjamin/Cummings, *1986).*

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

(10) Pople, H. "Heuristic Methods for Imposing Structure on III-Structured Problems: The Structuring of Medical Diagnosis," edited by P. Szolovitz. *Artificial Intelligence in Medicine* (Boulder, CO: Westview Press, *1982).*

(11) Reggia, J.A., Nau, D.S., & Pearl Y. Wang. "Diagnostic Expert Systems Based on a Set Covering Model," edited by M.J. Coombs. *Developments in Expert Systems* (Lon don: Academic Press, *1984).*

(12) Schank, R.C., & Abelson. *Scripts, Plans, Goals and Understanding* (Hillsdale, NJ: Lawrence Erlbaum, *1977).*

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

(14) Woods, D.D., Roth, E.M., & L.F. Hanes. *Models of Cognitive Behavior in Nuclear Power Plant Personnel: A Feasibility Study. Vol 2: Main Report* (Pittsburgh, PA: Westinghouse Electric Corporation, *1986).*

ManuscriptreceivedJune1987;RevisedAugust1987;AcceptedAugust1987.

Address correspondence to M.J. Coombs, Knowledge Systems Group, New Mexico State University, Box 30001, Las Cruces, NM 88003-001.