REAL TIME CAUSAL MONITORS FOR COMPLEX PHYSICAL SITES*

Chuck Rieger and Craig Stanfill
Department of Computer Science
University of Maryland
College Park, MD 20742

ABSTRACT

Some general and specific ideas are advanced about the design and implementation of causal monitoring systems for complex sites such as a nuclear power plant or NASA missions control room. Such "causal monitors" are interesting from both the theoretical and engineering viewpoints, and would greatly improve existing man-machine interfaces to complex systems.

INTRODUCTION

Human understanding of a large, complex physical facility, such as a NASA mission control room or a nuclear power plant, is often a haphazard affair. Traditionally, once a site is built, knowledge about it resides in technical manuals (which tend to sit on the shelf) and in the minds of the experts. As technical staff turnover occurs over an extended period of time, this knowledge tends to get misplaced, rearranged, or altogether forgotten, at least as day-to-day working knowledge. The result is usually that no single individual fully appreciates or understands the system as a whole; the operators learn simply to cope with it to the extent required for daily operations and maintenance. Of course, this means that when an emergency occurs, or when need arises to perform some unusual maneuver, the required human expertise is often hard to locate. Manuals on the shelf are virtually worthless in most contexts, especially those in which time is critical. Even when the expert is on the site, it may take him too long to perceive the context of an emergency and perform the synthesis of a large set of parameters necessary to diagnose the problem.

Given this state of high technology, which produces systems too deeply or broadly complex for an individual or reasonably sized group of individuals to comprehend, there is a clear need for "intelligent" secondary systems for monitoring the primary systems. To develop such intelligent systems, we need (1) flexible representations for physical causality, (2) good human interfaces that accept descriptions of the physical world and build descriptions in these representations, (3) models of "comprehension" that are capable of relating the myriad states of the site to the causal description, then passing along only relevant and important information to the human operators, and (4) fast, efficient symbolic computation environments to support items 1-3 in real time.

This paper briefly describes the CAM (Causal Monitor) Project, whose aim is to develop a framework for the construction of intelligent, causally-based real-time monitors for arbitrary physical sites. The project is under current funding by NASA Goddard Space Flight Center, and will result in a prototype causal monitor generator system. Our aim here is to advance some specific ideas about the conceptual architecture of such a system. Background ideas on the concept of causal models for man-made devices can be found in [6].

* The research described here is funded by NASA. Their support is gratefully acknowledged.

GOALS

An ideal causal monitoring system does not replace humans, but rather extends and broadens their abilities to process information, and enhances their ability to collect and synthesize relevant information in time critical situations. The ideal system would appear to the human controllers as a small number of color CRT displays and a keyboard. The system would

1. continually sense and symbolically characterize all sensors in a way that reflected their relative importance and acceptable ranges in the current operating context
2. continually verify that causally related sensor groups obey symbolic rules expressing the nature of their causal relationship
3. be aware of the human operator's expressed intent (e.g., "running preventative maintenance check 32", "executing morning power-up") and adjust causal relations and expectations and lower-level parametric tolerances accordingly
4. have a knowledge of component and sensor failure modes and limit violations, their detectable precursors, their indicators, their probable causes, and their corrective procedures (both in the form of automatic correction algorithms and recommendations to the human)
5. have a knowledge of standard "maneuvers", expressed as action sequences with stepwise confirmable consequences (for both automatic execution and as an archival reference for the human)
6. continually synthesize all important aspects of the system and from this synthesis, identify which aspects of the system to display to the human controllers (in the absence of specific requests from the humans)
7. decide on the most appropriate screen allocation and display technique for each piece of information under the current context

Most procedural knowledge about the primary system would be on-line, and in a form meaningful to both the human and the computer. The system would play the role of intelligent watcher, continually monitoring hundreds of sensors and relating them to the causal model for the current operating context. The system would summarize the site's state via well-managed CRT displays, and would be capable not only of detecting irregularities and suggesting probable causes, consequences and corrective measures, but also of automatically carrying out both routine and emergency corrective measures when given the go-ahead by the human controller. The physical site would be "self aware" in some limited sense.

ARCHITECTURE OF THE CAM GENERATOR SYSTEM

In the CAM Project we are concerned not with modeling a specific site, but rather with developing a general-purpose model that can be imported to any site. Once there, it will interact with the site experts as they define the site's causal structure via an interactive, frame-driven system. From the results of this knowledge acquisition phase, the specific site CAM is generated.

The CAM generator system consists of several pieces:

1. A collection of frame-like knowledge units that collectively span all aspects of virtually any physical site. At present we incorporate 9 frame types: Sensor, Actuator, Component, FailureMode, ActionStep, Maneuver, OperatingContext, DisplayPacket, OperatorIntent. Others will emerge as the project progresses.

2. A collection of procedures for interacting with site engineers and scientists who will interactively describe the site by filling in frames. This interface is semi-active, in that it ensures that relevant information (structured by the frames) is elicited in a methodical way from the site engineers. The information compiled by the frame-driven interface results in a collection of data objects and production rule-like knowledge about their interrelationships.

3. A collection of display primitives and display handling techniques

4. A collection of primitive sensor-reading and actuator-driving schemata (i.e., code schemata)

5. A system for compiling the production rule description of the site onto an efficient lattice-like data structure that obviates most run-time pattern matching

6. A run-time maintenance system for coordinating the real time of the symbolic causal model monitor.

The frames and the nature of the knowledge acquisition interface are described in [3]. Since the real-time efficiency of the generated system is a critical issue, we devote the remainder of the discussion here to the architecture of the CAM real-time monitor system, which we have termed the Propagation Driven Machine (PDM).

PROPAGATION DRIVEN MACHINES

CAM will require an extremely high throughput from its runtime system. For that reason, we cannot tolerate much general pattern matching in the basic machine cycle. To eliminate the need for pattern matching, we base our machine on a form of dependency lattice, similar in structure to those described in [1], [2],[4], and [5]. The essence of such a scheme is to represent antecedent-consequent relations by a graph; this is useful in both forward and backward reasoning.

The central data structure of the PDM is a graph whose nodes represent the current value of some parameter or proposition (e.g., "the current value of sensor 23 is X", "there is an overpressure condition in chamber 19"). The lowest level PDM network nodes are the primitive sensor readers and real time clocks whose spontaneous ticking sequences all propagation in the net. Nodes in the PDM are connected by Above and Below links, which reflect computational dependencies among nodes. For example, if node N1 represents knowledge of the form (A and B and C) causes D, then in the PDM net D will be Above N1, and N1 will be Above each of A, B, and C.

Computation in the network resembles that performed in a dependency net: when the value of any PDM node changes, all nodes Above it are placed on the agenda for reevaluation. The effect is to have data-driven computaions percolating up from changes in the clocks and sensors at the bottom of the net. Hence the name "propagation driven" machine.

Specifically, PDM control goes as follows. Initially the PDM net is empty (or relatively so). Some event in the model (e.g., operator start-up, the computation at a node in the net) identifies a rule set, R, that deals with some aspect of the modeled environment, e.g., the rules that describe the causal relationships in the Main Chamber Pressure System. Rules in this set are then queued on the PDM agenda, Q, from which they will all eventually be removed and evaluated.

An important PDM concept is structural augmentation of the network as a natural byproduct of a node's evaluation: evaluation of a newly queued rule causes the rule to be structurally knit into the PDM network. During a node's evaluation, any expressions not already present in the net are created as new net nodes, linked to the referencing expression's node via Above links, and placed in Q for expansion themselves. This process of "downward chaining" structurally introduces the rule into the PDM net.

Once installed by downward chaining, the rule's value will begin to receive constant PDM evaluation via "upward chaining": changes in lower nodes' values in the net will propagate upward through this structure, causing higher nodes to be reevaluated. Nodes subject to reevaluation by this mechanism are queued on Q. By using the same queue for both structural integration of rules into the current context and for the propagation of actual values, the PDM scheme allows a graceful and dynamic context-shifting mechanism. Also, it obviates the need for most run-time pattern matching.

To illustrate, suppose some node's evaluation has called for a rule set for monitoring the degree of openness of Relief-Valve-14 (RV-14) to be swapped in. The PDM system will, say, then queue up and evaluate (among others) a rule, R1, for comparing the position of RV-14 against the pressure of Pipe-14 (PP-14). This leads to a knitting process: R1 looks for RV-14 and PP-14 sensing or inferencing rules (nodes in the net) and finds neither. As a result, rules for determining RV-14's position and PP-14's pressure (say R2 and R3) are queued, evaluated, and also knit into the PDM net. Now in the net, when changes Below R2 and R3 propagate to either of these two nodes, R1 is requeued for evaluation via upward chaining, this time performing the actual comparison and computing a real value for the R1 node. This may in turn trigger a higher node, e.g., a display primitive for communicating the results to the human operators.

This dual-purpose use of the PDM queue, namely for staking out the net structure and for performing computations by propagation of changes in net node values, makes for a topologically dynamic system. As parts of the net are propagating values upward, other parts of the net may be sprouting, while others are atrophying (and being garbage collected). The system is essentially a compiled production rule system, but one in which "compilation" is as dynamic as the propagation of values. Rule sets can come and go as naturally as values can be changed.

Semantically, the lowest nodes of the PDM network, and those that are present initially, are spontaneously ticking real-time clocks whose Above pointers go to sensor-reading nodes. Additionally, certain basic monitoring expressions are represented by nodes initially in the net. As the site's state changes, these "caretaker" rules (net nodes) will make reference to other rules not yet structurally part of the net, queuing them up. As these references come to be evaluated, their evaluation draws them into the PDM net, where they begin to participate in the monitoring process.

When the context shifts in a way that causes the caretaker node to be uninterested in the paged-in rule set, the caretaker node drops its reference to the rule set, effectively cutting the Below link between itself and the top members of the rule set. Nodes with no Above pointers and which are not marked as "permanent" are subject to PDM garbage collection. Garbage collection is effectively the inverse of the original downward chaining that knit in the rule set, and structurally removes the entire rule set by following Below pointers.

SUMMARY

A CAM generator system is a special type of knowledge acquisition system whose domain is causality in large, complex physical sites. It requires (1) models of knowledge elicitation from the site experts, (2) frame-like models of the concepts common to all physical sites and their causal topology, and (3) an efficient, yet flexible engineering solution to the problems of controlling a large symbolic system of production rule-like knowledge in real time. The PDM model appears to be a realistic approach to real-time monitoring: it exhibits the breadth and thoroughness of a classical production rule system, but without the usual problems of pattern matching and database maintenance.

The area of data driven real-time causal models of complex physical sites appears to be a very fertile and largely unexplored area of AI research. Research in this area will help bring the following areas of AI closer together: frame-based systems, knowledge acquisition, causal modeling, and efficient implementation techniques for real-time symbol manipulation. The domain is manageable because we are modeling systems in which deep problem solving is not a key issue, and theoretically interesting because of its breadth-wise complexity.

REFERENCES

[1] London, P. E., Dependency Networks as a Representation for Modelling in General Problem Solvers, Department of Computer Science, University of Maryland, Technical Report TR-698, 1978.

[2] McDermott, D., Flexibility and Efficiency in a Computer Program for Designing Circuits, MIT, AIM-402, 1977.

[3] Rieger, C. and Stanfill, C., A Causal Monitor Generator System, Forthcoming TR, Department of Computer Science, University of Maryland, 1980.

[4] Shortliffe, E. H., MYCIN: A Rule Based Computer Program for Advising Physicians Regarding Antimicrobial Therapy Selection, Memo AIM-251, Artificial Intelligence Laboratory, Stanford University, 1974.

[5] Sussman, G. J., Holloway J., and Knight, I. F. Jr., Computer Aided Evolutionary Design for Digital Integrated Systems, MIT, AIM-526, 1979.

[6] Rieger, C., and Grinberg, M., A Cause-Effect System of Representation for Computer-Aided Design, Artificial Intelligence and Pattern Recognition in Computer Aided Design, J. C. Latombe (ed.), North Holland, 1978