PROCESS CONTROL USING A REAL TIME EXPERT SYSTEM

Proc. IFAC 1990, Estonia, USSR pp. 234-239 (reformatted)

R. Moore, H. Rosenof, and G. Stanley *

Gensym Corporation, 125 CambridgePark Drive, Cambridge, Massachusetts 02140, U.S.A.

Abstract. The challenges and status of the application of expert system technology in process plants is considered. In particular the problems of knowledge representation include the need to represent dynamic qualitative knowledge, dynamic analytic knowledge and the deep structure of the process. The application of inference in real-time requires paradigms which use metaknowledge to focus the inferencing resources of the expert system. Finally the application of truth maintenance requires a temporal model of the time dependence of the truth of data and inferred results. A structure which includes these considerations is presented.

Keywords. Artificial intelligence, Expert systems, Real time computer systems, Process control

INTRODUCTION

The work done to connect expert systems to on-line processes goes back many years (Astrom 1984, Kawakami 1984, Lusk and Stratton 1983, Yamada and Motoda 1983). The early activity toward developing a real-time expert system technology (Cartwright and Ruskin 1986, Kramer and Palowitch 1985, Moore and co-workers 1984, Sauers and Walsh 1983) was intended for the area of on-line process diagnosis. Some early real-time designs were subsequently placed on-line in refineries and other sites (Moore and Kramer 1986). Subsequent work (Astrom 1989, Hofmann and co-workers 1989, Moore and co-workers 1987, Wolfe 1987) has extended the working definition of "real-time expert system" to include temporal reasoning based on integrated heuristic and analytic knowledge, and a full object-oriented representation integrated with deep knowledge of the structure of the plant.

This paper presents the current working definition of a real-time expert system, based on 200 site installations. The many technology-leading organizations involved have provided significant contribution to the concepts. Some public references to this are (Arzen 1989a), (Aynsley, Peel and Morris 1989), (Byrd, Fisher and Mallett 1989), (Merris 1989), (Nevins 1989), and (Noren 1988). Current installations include applications in process monitoring and control, manufacturing, spacecraft telemetry, robotics, environmental control, network management and others. The most frequent implementations are in the chemical, petrochemical, aerospace and nuclear power industries.

KNOWLEDGE REPRESENTATION

Engineers and other plant experts know many types of process behavior, ranging from the analytical models of reactions to the lessons of past experience. To represent this in an expert system requires a knowledge representation that integrates several kinds of knowledge, including not only rules, but also differential equation models such as material balances, energy balances and reaction kinetics. A paradigm which only allows heuristics would, in the case of a process plant, only represent part of the expert knowledge of plant behavior. In this view, what is required is an integration of heuristic and analytic forms of knowledge.

The object-oriented paradigm is used, with a hierarchical frame structure. Each object has behavior, which is directed or represented by combinations of analytic (model-based) and heuristic (rule-based) statements. In the object-oriented paradigm, behaviors may be defined for a class, and be inherited by each member of the class or subclass. However, with the object oriented paradigm, a member of a class can inherit some behaviors and not other, more specific, behaviors.

For the most efficient representation, behavior is defined generically, with as much abstraction as possible, so that a class hierarchy is defined with behaviors at the highest class possible.

Generic Representation of Behavior

In many plants there are sets of objects which have related knowledge. For example, there may be many chemical reactors, all of which behave according to their individual states and inputs, but all with temperature, pressure and reaction relationships functionally defined by the combination of these and generic relationships. All the reactors may need inference analysis of a similar sort. The generic knowledge at the various class levels is combined with the parameters, states, inputs and specific behaviors of the specific instance to determine the overall object behavior, such as;

> d/dt(the temperature of any reactor) =(the net-energy-input of the reactor - the net-energy-output of the reactor + the reactor-heat of the reactor) * (the ....

and;

> the net-energy-input of any process-equipment = the sum over the inputs of the process-equipment of (........)

Representation of the Plant Structure

The process experts reason about the behavior using their knowledge of plant structure, knowing how the connected process units affect each other. In some applications the connections may change dynamically, as when alternate scenarios are being evaluated. To represent this, it is useful to use a graphical form of knowledge representation, so experts can define the plant structure and interactions by connecting objects on a computer workstation screen. The expert system can interpret this deep structure as part of the reasoning paradigm. The inference engine actively interprets the schematic representation of plant structure, reasoning about up-stream and down-stream causes and effects. To allow this, the object behaviors are defined in terms of data at object ports, rather than naming the specific objects providing the interactions, such as;

> .....the integrated-heat-flux of any reactor = the sum over the inputs of the reactor of.........

and;

> if the pressure of any reactor > the maximum-allowed-pressure of the reactor then invoke overpressure rules for every process-equipment connected to the reactor.

Temporal Knowledge Representation

Temporal knowledge is typically very important, as the expert may be more concerned with the direction of dynamic behavior than with the current values of data. The differential equation models are one analytic representation of temporal knowledge. Other forms, which may be incorporated into heuristic or analytic behavior include relationships between data or events over time, such as;

> If the temperature of any tank as of 2 minutes ago > .....then....

or functions of data over time, such as;

> if the rate of change per minute of the temperature of any reactor over the last 2 minutes >.....then invoke safety rules for the reactor

Integration of Knowledge Representations

The overall knowledge representation is done using a combination of heuristic and analytic object behavior, including temporal knowledge, some of which is inherited from higher classes and some of which is specific, combined with the deep structure of the connectedness of the objects as dynamically interpreted during operation of the expert system, Fig. 1. Hofmann (1989) shows how this knowledge representation can be used to rapidly develop large applications using block diagrams of objects and their interactions as the user interface.

We can say that now there is an increasing realization that a full control strategy requires not only parameter identification, state estimation and control , but that the validity of the data and process models must be checked even before the data and models are used in state estimation and subsequent control. As systems get more complex, there is a higher and higher probability of some failure occurring. The systems must be fault tolerant, allowing best available information to be used in flexible ways. For large system implementations, a productive human interface for the development and maintenance of the system, as is described here, becomes a necessary condition for practical implementation. Fig. 2. shows how the structured natural language parser can be used for building the knowledge base.

REASONING IN REAL TIME

First attempts at using expert systems for process control involved taking a "snap-shot" of plant data and using a static expert system paradigm to perform inference. Static expert systems use pattern matching of a set of "facts" and a set of rules, using some combination of forward and backward chaining to combine the inferences. With no time constraints, this might be practical. However with a time constraint, we need more efficient inference paradigms.

Code improvements and computer performance improvements can help. For example the knowledge base can be manually partitioned to cut down on the search space. This is conceptually like having separate libraries or separate expert systems. Another efficiency derives from truth-maintenance. If only a few data are changing at a time, the truth-maintenance techniques of unwinding obsolete conclusions and only updating conclusions dependent on the new values is useful. However in most real-time applications a considerable amount of data is changing, at a rapid rate. Even with the code improvements and the continuing computer performance improvements, the static expert system approaches lead to slow performance on even small prototypes of a few hundred rules and a few hundred rapidly changing data.

Real-Time Reasoning Paradigms

A fundamentally different inference approach is appropriate for real-time problems. One approach that a human expert uses in a real-time situation is to maintain a peripheral awareness across the domain, watching for performance exceptions, and then focusing on areas of interest, using knowledge appropriate to the task. The G2 inference engine operates similarly. This requires various types of metaknowledge, so the expert system can invoke the appropriate knowledge. The metaknowledge may involve the problem type, the objects involved including connected objects, and other knowledge-about-knowledge. A retrieval facility allows the inference engine to invoke the requested types of knowledge, such as;

> if.....then invoke safety rules for any reactor where the temperature of the reactor > ....

The initiation of a particular reasoning process may be from several causes. The most obvious is an event, which in the process plant is typically an alarm, such as;

> whenever the level-alarm of any tank receives a value and when the alarm is not return-to-normal then ....

In process plants, one of the more promising roles of the expert system is to detect problems earlier, before they lead to an alarm condition. This requires that some reasoning be done continuously or periodically, regardless of the absence of "events" in the plant. This reasoning typically involves multiple measurements and dynamic models, analytic or heuristic, which in combination represent the way an expert would interpret the data.

The inference engine continually scans some of the knowledge which the expert has specified for awareness of conditions which are not yet at alarm limits. If a safety-threatening condition occurs, the inference engine uses metaknowledge to determine which knowledge to invoke, thus focusing on the area of interest.

Performance issues

One benefit of the metaknowledge approach is that very large knowledge bases can be run in real time. Since many types of problems and behaviors are represented in the knowledge base, it can get quite large, with thousands of objects and rules. However G2 does not consume computer time looking for patterns. Rather it focuses attention on the knowledge needed. Thus a knowledge base may contain thousands of rules and objects, most of which are not consuming much CPU time, since nothing interesting is happening with them at the moment.

In static expert systems, truth maintenance involves changing inferences when data changes. In real-time problems there may be an additional requirement to change inferences even if no new data is available, since measurements cannot be assumed to remain valid indefinitely.

One way to express this temporal validity information is to attach an expiration time to each value maintained by the inference engine, and propagate this when inference is carried forward. Generally, when a conclusion is based on several time-sensitive variables, the earliest of their respective expiration times will be carried forward. Expiration times can be propagated forward through multiple levels of inference, but there are also ways to limit this propagation. This requires that data types be more complex, including validity information as well as values.

The inference engine takes advantage of the truth maintenance structure to achieve further efficiency. If a high level conclusion is not currently of interest, then there is no need to forward-chain from the newest data values to update the conclusion. If the high level conclusion becomes of interest, then the validity information determines whether the conclusion needs updating, via backward chaining, or whether the latest value is still valid. This reasoning process can be overridden by the expert specifying that certain data should always drive forward chaining. The overall result is a considerable performance improvement over static paradigms, and it seems similar to the way a human expert deals with a dynamic domain.

The forward propagation of validity intervals is under control of the developer. An example where the expiration times would not be carried forward would be in identification applications, where individual data may contribute to the learning of the parameter values of a model, and the "learned values" might have longer validity than each individual datum that entered into the identification.

The Scheduler

A real-time scheduler maintains the overall operation. Each task is a small one, perhaps a few milliseconds, and then the scheduler is in control of the next task. Some tasks are done periodically, such as integration of the state variable models and scanning of rules which are looking for problems. Other tasks are scheduled for the next available time-slot, such as invoking rules through metaknowledge. Still other tasks are scheduled for specific times, such as procedural checking of a batch operation. The scheduler also manages the data acquisition, by scheduling the data servers, and the data output which may be setpoint changes or direct control actions.

The scheduler has facilities to consider the handling of asynchronous data when making an evaluation, which is a frequent problem in distributed applications. In most conventional continuous control theory (especially in state-variable systems), it is assumed that an entire vector of measurements is attained at a particular instant. All calculations are then done instantaneously as well. Neither data acquisition nor calculations are really done so instantly and synchronously in the real world. G2 is designed to handle every input, output, and calculation asynchronously and with priorities. This allows generality, allows true distributed processing, and insures that the computer never rests when there are tasks to be done.

Ensuring Coherence

The asynchronous data arrival time is available through its time stamp. As noted already, this time stamp can be propagated through subsequent calculations as a default option. More subtly, consider the following rule, where X is a variable that can change quite rapidly;

If X> 10 and Y > 10 then conclude that ...

G2 may be able to get a value of X quite quickly from a data acquisition system or internal calculation. Suppose, however, that it takes a long time to get Y, perhaps from a slow data acquisition system, or from a large external simulation. By the time Y arrives, the value of X may have expired due to a short validity interval assigned to a rapidly-changing variables. G2 will automatically get a new value for X in this case. G2 ensures that a coherent data set is acquired - that is, at one point in time, all variables have the indicated values, within the time resolution indicated by validity intervals. This data set coherence becomes an important issue as larger distributed systems are considered, and no single measurement vector representing values at a single point in time is available.

Similar considerations exist with rule actions. They can be carried out "in parallel", or "in order". The difference becomes especially apparent if one of the rule conclusions changes the value of one of the antecedent variables either directly or indirectly. For "parallel" rule action execution, the values of the antecedent's variables are held constant until all actions can be completed. "In order" execution allows sequential execution. Note that the inference engine is required to support "loops" between rule antecedents and consequents. While this is forbidden in many static expert systems, it is essential if the real-time expert system is going to deal with closed-loop systems, which do in fact have "loops". The provisions of validity intervals, data coherence and other aspects described above make this possible.

Acting Within a Time Limit

Tasks have priorities, and within each basic interval the tasks are executed by priority. Priorities can be changed, and whole sections of knowledge can be disabled or enabled under rule control as an overload strategy. The scheduler has active metering, accessible to rules, so that flexible overload strategies can be constructed.

Finding an answer within some specific time interval is facilitated by a default option. If a rule cannot be satisfied within a specified interval, another rule can be invoked to take a backup action. A typical cause might be the required data not being available. An alternative is to guarantee the "best" answer within a finite time-out period, such as;

> If the first of the following that has a current value (the temperature-sensor of the reactor, the model-inferred-temperature of the reactor, the manual-backup-temperature of the reactor, the default-temperature of the reactor) > 200
> then.......

In the above expression, if no valid value is currently available in the first element of the list, a data acquisition is started. In the meantime, the second element is checked. If it has no current valid value, the calculations for the second one are started, and so on. At the end of the time-out period, the "best" answer (closest to the head of the list) is taken, and any remaining outstanding calculations are cancelled.

KNOWLEDGE VALIDATION

Consistency and completeness of knowledge are significant issues in the process domain, particularly as safety may be involved. The knowledge structure of G2, which allows dynamic models to be attached to objects, provides a framework for simulation testing as a step toward judging the consistency and completeness of the knowledge.

Failure scenarios are a powerful way of validating a knowledge base. Where a dynamic model already exists, a failure scenario is created by introducing a problem such as a failed sensor or controller, or a physical problem in some component such as a leak or breakdown or overheating. The dynamic model will then propagate the effects of the failure, and one gets an opportunity to learn what the knowledge base will do in this situation. Many fault conditions, which would otherwise be untested until actual plant emergencies, can be simulated in this way.

Provisions for locking a knowledge base after validation provides security. At the same time, experimentation with new or modified knowledge bases can be done concurrently with on-line operation of the validated knowledge base. This is possible because the knowledge base is stored as an ASCII file, which can be transported to other G2's and operated on by other G2's. The abstraction of data serving allows multiple versions of the knowledge base to operate on the same CPU or on networked CPU's, accessing the same on-line data, so experimentation with alternative knowledge bases can be done concurrently with normal operation.

METHODOLOGY EXAMPLE

We consider a typical fault diagnosis using G2. The first step is detecting the problem, which may be indicated by an event (alarm). However the more desirable situation is to detect problems before they develop into "events". To allow this, generic rules can scan across the domain looking for problems. This typically involves a combined analysis using multiple measurements. Analytic and heuristic models define expected behavior. The expert system then compares expected performance with measured behavior. This is similar in concept to the Kalman filtering or state observer technology, except that heuristic as well as analytic behavior may be the basis for the expected behavior.

The next step is understanding the problem. For this purpose we use several tools for diagnosis of anomalous behavior. These include invoking diagnosis of connected objects, distinguishing between measurement errors and domain faults, invoking diagnostic knowledge using metaknowledge, and focus on problems.

Reacting quickly to problems is facilitated by testing hundreds of rules per second, with sometimes multiple focuses of attention. The expert system can give advice or take action.

APPLICATION STATUS

As of this writing, the on-line applications of G2 include telemetry monitoring of flight data, manufacturing processes, manufacturing assembly, a biochemical process, a variety of chemical and petrochemical processes, closed-loop robotics and closed-loop process control. Research prototypes are underway in network monitoring, financial transaction monitoring, office workflow planning, closed life support systems, semiconductor manufacturing and a number of nuclear reactor safety projects.

Some of the applications involve implementation of conventional multivariable and advanced control, using the combination of heuristic and analytic methodology, and taking advantage of the framework to facilitate implementation. Other applications are extending beyond what is reasonably possible with conventional systems, to implement measurement validation across a plant with 5000 objects and the equivalent of 7500 rules in one case. Other applications are directed toward large-scale fault-tolerant systems, which can use heuristics and the facilities for real-time truth maintenance to provide backup and "best available solution" behavior.

REFERENCES

Arzen, K-E. (1989a). Knowledge-Based Control Systems: Aspects on the Unification of Conventional Control Systems and Knowledge-Based Systems. Proc. 1989 American Control Conference, Pittsburgh, Vol. 3, pp 2233-2238.

Arzen, K-E. (1989b). An Architecture for Expert System Based Feedback Control. submitted to Automatica.

Astrom, K. J. and J. J. Anton (1984). Expert Control. IFAC World Congress, Budapest.

Astrom, K. J. Towards Intelligent Control - Keynote Speech to the 1988 American Control Conference. IEEE Control Systems Magazine, 9:3, 60-64.

Aynsley, M., D. Peel, and A. J. Morris (1989).   A Real Time Knowledge Based System for Fermentation Control.   Proc. 1989 American Control Conference, Pittsburgh, 3, 2239-2244.

Byrd, J. S., J. J. Fisher, and W. R. Mallett (1989).   Expert Robots for Process Environments. Robotics and Autonomous Systems. New York: Elsevier Science Publishers.

Cartwright, C. and P. Ruskin (1986). Musing on the needs of real-time AI toolkits. Expert System User, V2, N-9, London, 30.

Hofmann, A. G., G. M. Stanley, and L. B. Hawkinson (1989). Object-Oriented Models and Their Application in Real-Time Expert Systems.   Proc. 1989, Society for Computer Simulation International Conference, San Diego.

Kawakami, J. (1984). Overview of Artificial Intelligence Applied to Control Technology in Japan and in Hitachi. Hitachi Research Laboratory, private communication, Japan.

Kramer, M. A. and B. L. Palowitch, Jr. (1985). Expert Systems and Knowledge-Based Approaches to Process Malfunction Diagnosis. Proc. AIChE Annual Meeting, Chicago.

Lusk, E. and R. Stratton (1983). Automated Reasoning in Man-Machine Control Systems. Proc. Ninth Annual Advanced Control Conference, West Lafayette, 41-48.

Merris, D. (1989).   Gensym real time expert systems move into applications arena. Flexible Automation, Feb. 1989, 4.

Moore, R. L., L. B. Hawkinson, C. G. Knickerbocker, and L. M. Churchman (1984).   A Real-Time Expert System for Process Control.   Proc. First Conference on Artificial Intelligence Applications (IEEE), Denver, 569-576.

Moore, R. L., and M.A. Kramer (1986).   Expert Systems in On-Line Process Control.   Proc. Third International Conference on Chemical Process Control, Asilomar, 839-867.

Moore, R. L., L. B. Hawkinson, M. Levin, A. G. Hofmann, B. L. Matthews, M. H. David (1987). Expert System Methodology for Real-Time Process Control. Proc. 10th World Congress on Automatic Control, IFAC, Munich, 6, 274-281.

Noren, C. S. (1988). Rapid Prototyping Network Management Systems. IEEE Milcom '88, San Diego.

Rosenof, Howard (1989). Real-Time Expert Systems in Process Control.   EPRI Conference on Expert Systems Applications for the Electric Power Industry, Orlando.

Sauers, R. and R. Walsh (1983). On the Requirements of Future Expert Systems. Proc. Eighth IJCAI , Karlsruhe, 110-115.

Stanley, G. (1989). The G2 Real-Time Expert System.   Third International Conference on Expert Systems and the Leading Edge in Production and Operations Management, Hilton Head.

Wolfe, A. (1987). An Easier Way to Build a Real-Time Expert System.   Electronics, March 5, 1987.

Yamada, N., and H. Motoda (1983). A Diagnosis Method of Dynamic System using the Knowledge on System Description. Proc. Eighth IJCAI, Karlsruhe, 225-229.

* Robert Moore may be contacted at

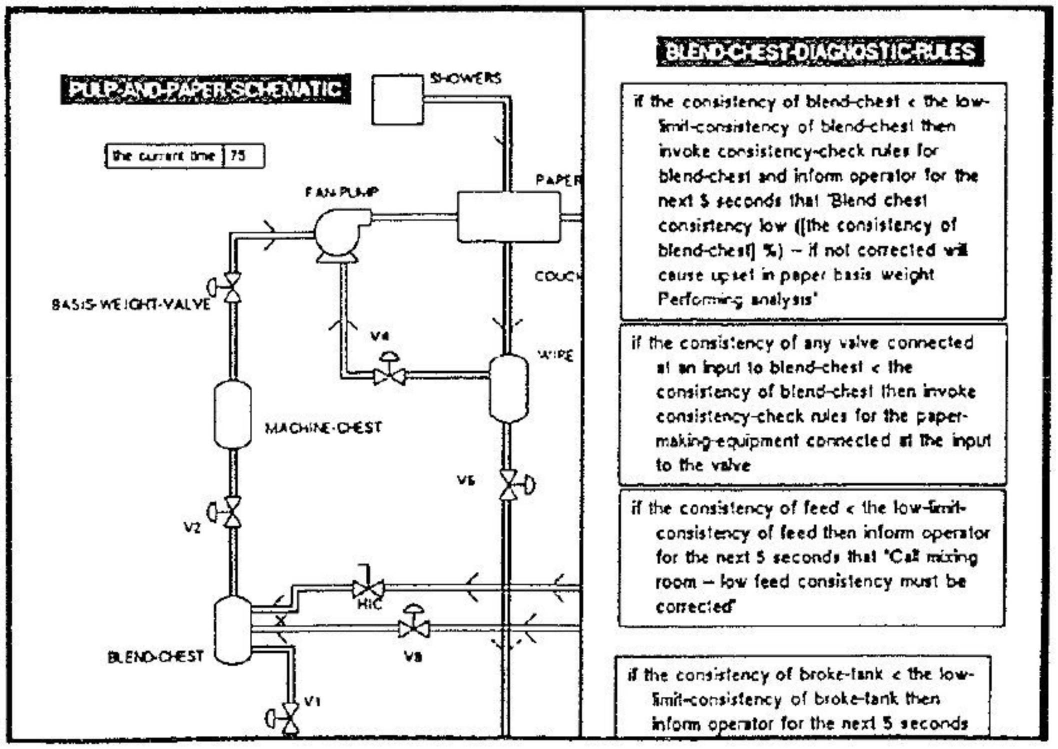
http://www.calventuretech.com/contactus.html

* Howard Rosenof may be contacted at

http://www.linkedin.com/pub/howard-rosenof/5/229/866

* Greg Stanley may be contacted at

http://gregstanleyandassociates.com/contactinfo/contactinfo.htm



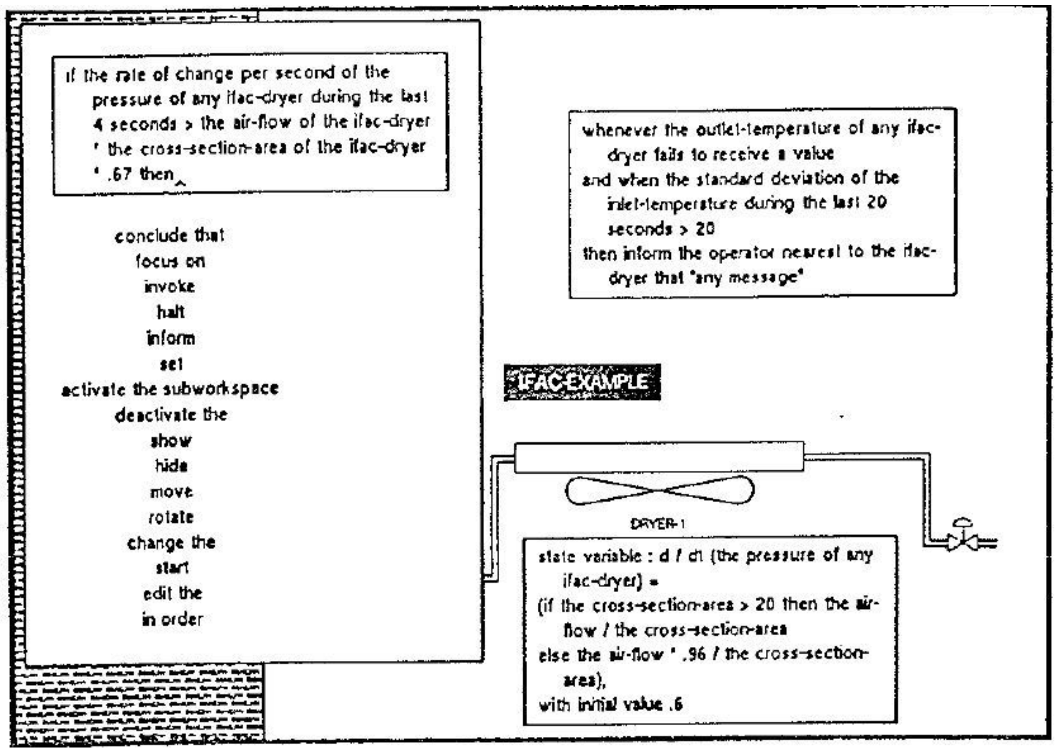
if the consistency of blend-chest < the low-limit-consistency of blend-chest then invoke consistency-check rules for blend-chest and inform operator for the next 5 seconds that "Blend chest consistency low ([the consistency of blend-chest] %) – if not corrected will cause upset in paper basis weight Performing analysis"

if the consistency of any valve connected at an input to blend-chest < the consistency of blend-chest then invoke consistency-check rules for the paper-making-equipment connected at the input to the valve

if the consistency of feed < the low-limit-consistency of feed then inform operator for the next 5 seconds that "Call mixing room – low feed consistency must be corrected"

if the consistency of broke-tank < the low-limit-consistency of broke-tank then inform operator for the next 5 seconds

Fig. 1. Examples of heuristic and analytic knowledge using deep structure.



if the rate of change per second of the pressure of any ifac-dryer during the last 4 seconds > the air-flow of the ifac-dryer * the cross-section-area of the ifac-dryer * .67 then

conclude that
focus on
invoke
halt
inform
set
activate the subworkspace
deactivate the
show
hide
move
rotate
change the
start
edit the
in order

whenever the outlet-temperature of any ifac-dryer fails to receive a value
and when the standard deviation of the inlet-temperature during the last 20 seconds > 20
then inform the operator nearest to the ifac-dryer that "any message"

state variable : d / dt (the pressure of any ifac-dryer) =
(if the cross-section-area > 20 then the air-flow / the cross-section-area
else the air-flow * .96 / the cross-section-area),
with initial value .6

Fig. 2. Structured natural language for building the knowledge base.