PROCEEDINGS OF THE

26th IEEE CONFERENCE ON DECISION AND CONTROL

DECEMBER 9-11, 1987
WESTIN CENTURY-PLAZA HOTEL
LOS ANGELES, CALIFORNIA

IEEE
Control
Systems
Society

VOLUME 3 OF 3

87CH2505-6

The 26th IEEE Conference on Decision and Control

December 9-11, 1987

IEEE Control Systems Society
Sponsoring Organization

Cooperating Organizations
Society for Industrial and Applied Mathematics
Operations Research Society of America

OPERATING COMMITTEE

GENERAL CHAIRMAN
Professor William S Levine
Department of Electrical Engineering
University of Maryland
College Park, MD 20742
(301) 454-6841

PROGRAM CHAIRMAN
Professor John Baillieul
Aerospace/Mechanical Engineering
Boston University
110 Cummington Street
Boston, MA 02215
(617) 353-9848

FINANCE
Dr. David W. Porter
Business & Technological Systems, Inc

PUBLICATIONS
Dr Malcolm D. Shuster
Business & Technological Systems, Inc

PUBLICITY
Mr. Melvyn Rimer
Grumman Aircraft Systems

REGISTRATION CHAIRMAN
Professor Fawzi Emad
Electrical Engineering Department
University of Maryland
College Park, MD 20742
(301) 454-6873

EXHIBITS
Mr. James H. Beggs
Robertshaw Controls Company
3000-D South Highland
Las Vegas, NV 89109
(702) 733-6500

LOCAL ARRANGEMENTS
Prof Mohinder Grewal
Department of Eiectrical Engineering
California State University, Fullerton
Fullerton, CA 92634
(714) 773-3013

PROGRAM COMMITTEE
J Baillieul, Chairman
P. Bernhard
B. Bonnard
P Crouch
A Desrochers
J. Grizzle
J Hollerbach
P. Ioannou
P P Khargonekar
A J. Krener (SIAM)
P S. Krishnaprasad
S. I. Marcus
C F Martin
M P. Polis
H. V. Poor
W. F. Powers
W. E Schmitendorf
L Sennott (ORSA)
P. B. Usoro
R. B. Washburn
W S. Wong

Copyright and Reprint Permissions: Abstracting is permitted with credit to the source Libraries are permitted to photocopy beyond the limits of U.S copyright law for private use of patrons those articles in this volume that carry a code at the bottom of the first page, provided the per-copy fee indicated in the code is paid through the Copyright Clearance Center, 29 Congress St., Salem, MA 01970 Instructors are permitted to photocopy isolated articles for noncommercial classroom use without fee. For other copying, reprint or republication permission, write to Director, Publishing Services, IEEE, 345 East 47 Street, New York, NY 10017-2394 All rights reserved Copyright © 1987 by The Institute of Electrical and Electronics Engineers, Inc

Library of Congress Catalog Card Number: 79-640961
IEEE Catalog Number: 87CH2505-6

Additional copies of this proceedings may be ordered from the IEEE Service Center, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08854-1331, Telephone 201-981-1391.

Proceedings of the 26th Conference
on Decision and Control
Los Angeles, CA • December 1987

RULE-BASED REAL-TIME CONTROL SYSTEMS

James H Taylor
GE Corporate R & D
Schenectady, NY 12345

Richard W Gerhardt
General Motors Corp
Warren, Mich 48090

E Craig Luce
Kearfott Div Singer Co
Little Falls, NJ 07424

ABSTRACT

The premise that one can obtain more effective control systems with less engineering effort, less technical risk, and less cost via the use of artificial intelligence techniques is explored in this paper In particular, we discuss, develop, and illustrate the use of a **rule-based system for real-time control** as a software environment that can be used for control system implementation, relieving the design engineer of much of the burden of taking a basic control algorithm and making it work in a real-world application

We show that the rule-based systems approach can be used to provide substantial support for the above task A rule-based system provides the environment in which the designer can develop and test all of the required heuristic logic and control, using a programming language (production rules) that is ideally suited for the task. We develop the details of such an implementation methodology, and illustrate its application to failure detection and isolation

1. INTRODUCTION

Obtaining better control systems with less effort, cost, and technical risk using artificial intelligence (AI) techniques has become the focus of considerable research activity in recent years One approach is to use a rule-based system as an appropriate software environment for real-time control system implementation, taking advantage of the built-in heuristic logic and/or reasoning capability to facilitate control system realization and validation This promises to relieve the design engineer of much of the burden of taking an "academic design" (basic control algorithm) and making it work in a real-world application Another potentially useful application of this methodology is aiding a human operator in dealing with excessively difficult control situations, or even eliminating the need for operator "loop-closing" by implementing the human control strategy as an expert system

The specific problem addressed in the first case is the well-known fact that obtaining a control algorithm is often a small part of the control engineer's responsibility; making it work in terms of system interfaces, initialization procedures, exception-handling, operator interface, etc. is usually more difficult and time-consuming This problem is especially important when dealing with advanced systems designs where estimation routines, adaptive algorithms, failure detection and isolation schemes, parameter identification, etc are included in the control system software

The second application involves situations where AI-based approaches can be used to alleviate problems that presently arise from closing or supervising control loops via the actions of human operators. The problems solved by expert systems or other AI techniques in this case include excessive operator work-load, requirements for speed of response that stress or exceed human capability, human sensory or decision-making overload in the case of emergencies in complicated systems, etc We have primarily considered problems associated with implementation, as outlined above, although the same methodology should be applicable in both situations

There is often no systematic approach available for the implementation part of the control system design effort using standard programming methods; rather, the engineer often has to resort to a "fix and tune" strategy, adding heuristic logic to the controller software and testing the control system until it works in the real plant and operating environment The resulting logic and algorithmic modifications often becomes a large and unwieldy mass of patch-work or "spaghetti software" that is difficult to implement, document, and maintain and which greatly exceeded original cost estimates

The rule-based systems approach can be used to provide a great deal of support for the control system implementation task, resulting in a more flexible system realization with less designer iteration and frustration and greater likelihood of success than can usually be achieved by traditional means In essence, the rule-based system provides the environment in which the designer can develop and test all of the required heuristic logic and control, using a "programming language" (rule writing) that is ideally suited for the task and mechanisms (e g , "inference engines") that carry out the required control strategy

The resulting rule-based real-time control system may actually be implemented as a rule-based system, or it may be "compiled" or translated into a lower-level language if the flexibility of the rule-based environment is no longer needed in the target application Either way, the design task has been greatly simplified, and the full control system software definition can be maintained and documented in the rule-based form

The balance of this paper is organized as follows: Section 2 deals with a candidate general architecture for rule-based real-time control; Section 3 treats "internal" considerations such as expert system structures

CH2505-6/87/0000-1923$1.00 © 1987 IEEE

and mechanisms for efficiency, truth maintenance, and reasoning with time-valued information; Section 4 outlines the application used as the vehicle for this research; and Section 5 provides a discussion of our findings and conclusions.

2. GENERAL ARCHITECTURE FOR RULE-BASED REAL-TIME CONTROL

Our research in expert systems for control has focussed on the proper role of AI in real-time control systems implementation, architectures for rule-based real-time control systems, and customizing the rule-based system environment to achieve the best possible convenience and performance. This section and the next summarize our findings.

There are various models for the combination of expert systems and control technology. Our concept is a **rule-based real-time control system** having:

- **sensors** (monitors) to determine the state of the plant in terms of "signals", e.g., sensor outputs,
- **pattern recognizers** or feature extractors to process signals and create "symbols" or linguistic representations of the information (e.g., SENSOR_NO_1 VALUE . HIGH),
- **low- and intermediate-level conventional controllers,** with interfaces that permit the introduction of commands for gain-setting, reconfiguration, etc.,
- **rule bases** that contain the knowledge of the control system designer and of the overall control strategy for all regimes, and
- **an inference engine** that exercises the highest-level control ("meta-control") of the system.

A rule-based real-time control system configuration that incorporates these elements is depicted in Fig. 1.

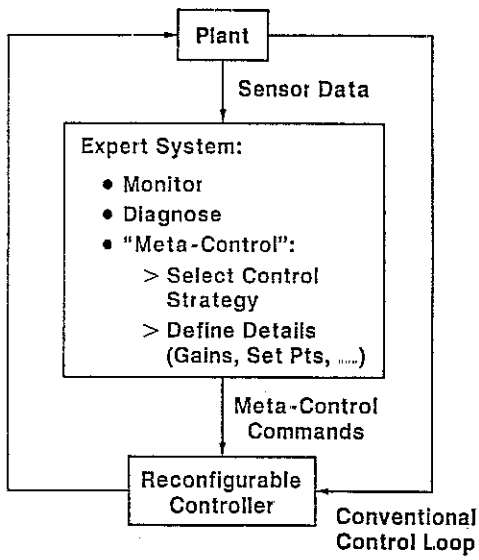

Figure 1. Rule-Based Real-Time Control Architecture

This architecture results in a hierarchical control scheme that, at the highest or expert system level, embeds the expertise of the experienced system designer (or, in some contexts, a capable human operator) while making the best possible use of conventional control technology. This general concept is quite similar to other approaches; cf. [1,2]

3. EXPERT SYSTEM INTERNAL STRUCTURE

The direct application of the overall architecture described above may present several problems: inefficiency (including inability to work in real time), lack of truth maintenance (e.g., inability to eliminate "stale" information from the knowledge base), and incorrect or ineffective reasoning with time-valued information. We have partitioned the rule-based real-time system into three parts, to manage and reduce the impact of these difficulties. The real-time expert system is divided into an input processor, a main inference unit or "supervisor", and an output processor. Each part is an independent rule-based system, using a knowledge base and inference strategy or strategies best suited to its role.

This partitioning is based on the premise that there are two essentially different types of data. The input and output processors deal solely with "volatile information" or data that is valid only from one sample point (t_k) to the next (t_{k+1}). The supervisor performs higher-level reasoning based on the input from the input processor and delivers the resulting conclusions to the output processor; the information processed in the supervisor module is of longer-term value ("non-volatile") and is carefully maintained to preserve its validity ("truth") and time-value.

In our implementation, the primary functions of the input processor are simple signal-to-signal transformation (e.g., testing a sensor output to see if it is "low", "nominal", or "high" and writing the corresponding fact into the knowledge base) and passing incremental volatile data (only that volatile data that has changed from the previous sample time) to the supervisor. The only inference mechanism in this part of our system is the data-driven or "forward chaining" mode. It is designed to be as simple and efficient as possible; it's primary function is to intelligently manage or "filter" the potentially large input data stream and supply the supervisor with just the information required for meta-control at an acceptable data rate.

The supervisor accepts incremental volatile data from the input processor and updates its knowledge base. The functions of this unit in our system include initialization, input mapping, diagnosis, meta-control, and "interfacing" (controlling its input and output interfaces). This rule-based system may use both goal-driven ("backward chaining") inference for overall objective management, plus forward chaining to update its knowledge base. Input mapping involves

the translation of incremental volatile data into the long-term facts in its knowledge base, and determining if the overall state of the system has actually changed. If no meaningful change is detected, then forward chaining stops; otherwise, the new data is used to update the diagnosis of the controlled process (plant) via further forward chaining. If the diagnosis is unchanged, then forward chaining stops; otherwise, the new diagnosis is used to update the control strategy (meta-control). Whenever this chain of reasoning is terminated, the interfacing rule is executed to output a change in meta-control (if any) and prepare to accept new input.

In this architecture, time-valued information management and truth maintenance are isolated within the supervisor. The input data stream is reduced and processed to provide symbolic input to the supervisor with clearly designated time-value. Within the supervisor, these functions are handled "manually", i.e., are specifically performed via procedures in the supervisory rule base. For example, time history data is managed in sequential form, with rudimentary knowledge of dynamic behavior in the form of rules for determining when a data sequence corresponds to transient or steady-state conditions. In the future, we believe that use of higher level expert system mechanisms such as frames (with slots for time-valued information and demons or inheritance properties that automatically manage such data) will provide a higher-level framework for this functionality.

The output processor provides the interface between meta-control decisions from the supervisor and the reconfigurable controller. The controller is a traditional digital control system, with an appropriate degree of programmability to accept the required meta-control commands. This may be as simple as accepting a revised set of control gains, or as complicated as performing a complete controller reconfiguration, depending on the application. The output processor's task is to convert meta-control from symbolic form to parameters and/or boolean variables to pass on to the process controller.

4. APPLICATION

The primary emphasis in this study was on the real-time rule-based system activity within the supervisor, especially, the efficient interface between the symbolic manipulation and data streams from the controlled system, truth maintenance, and reasoning about the time-dependence of symbolic information from the input processor. The following simulated application [7] was devised on the basis of these considerations.

4.1 Experimental Environment

We arrived at the results summarized in Sections 2 and 3 by assembling a "rapid prototype" of the rule-based real-time control environment outlined above.

We created this study environment by interfacing the GE expert system shell Delphi (see [3] for an overview of the capabilities and knowledge representation of Delphi) with the nonlinear simulation environment SIMNON [4]. This was done using a simple protocol for coordinating the SIMNON simulation with the rule-based system symbolic processing and meta-control via a minor modification of the mailbox / handshake file scheme developed in earlier efforts to apply expert systems to facilitate computer-aided control engineering (CACE) analysis and design called CACE-III [3,5,6].

Our "non-real-time simulation" of a rule-based control system thus proceeds as follows: Initialization (of both the expert system and the plant and controller simulation) is carried out by Delphi, then the "clock" started for a simulation from $t = t_0$ to $t = t_1$, the first meta-control "sample time" (which may be different from - usually longer than - the sample time of the conventional controller). At the first meta-sample time, sensor data is made available to the input processor, and an inference cycle (input processor / supervisor / output processor) carried out. Meta-control, if required, is performed as the final task of the expert system (e.g., revised controller gains are passed to the controller simulation model), and control passes back to SIMNON for simulation over the next interval ($t_1 < t \leq t_2$). This cycle of events continues until the simulation run is complete. Note that the inference activity in a real-time system would take place simultaneously with process dynamical behavior, finishing (it is hoped!) before the next meta-sample; non-real-time simulations (even of conventional sampled-data systems) do not reflect that property.

The required input processor signal-to-symbol transformations were actually hard-wired into the SIMNON simulation models, i.e., the simulation code included statements to write facts in a form intelligible to Delphi. The input processor then only had to determine the incremental volatile data and pass that information to the supervisor. The output processor was designed to accept symbolic information for meta-control, and to command the corresponding changes to the controller by writing SIMNON commands to change parameters in the controller model. The "programmability" of the controller was thus based on parameter changes and required the controller model to be written accordingly.

4.2 Application Overview

Our first application of rule-based real-time control involved a rule-based system implementation of a recently-developed failure detection and isolation (FDI) methodology [8]. We originally intended to add controller reconfiguration to accommodate failures, but have not done so. The FDI application met our study goals, demonstrating to our satisfaction the interplay between conventional control algorithms

and AI A brief overview of the method and the results of this study are summarized below.

4.2.1 FDI Problem Formulation: Given a linear time-invariant plant as depicted in Fig. 2, described by the $p \times m$ transfer function matrix $P(s)$. Let u_d be the desired or correct control input command and u be the actual plant input (output of the actuators). The difference between u and u_d is denoted by the additive signal $a(t)$, which could be set to $a(t) = a_0$ to model a bias error, $a(t) = a_0 - u_d(t)$ to model being "stuck" at a constant value a_0, $a(t) = (K - 1)\, u_d(t)$ to model a gain error, etc. Similarly, let y_d be the actual output of the plant (correct sensor output) and y to be the actual output of the sensor with error modeled by the additive signal $s(t)$ that may be set to model various errors in a similar fashion. The variables u_d and y are "external" or available for FDI; u and y_d are "internal" or inaccessible.

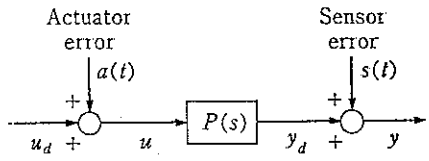

Figure 2. System and Failure Models

4.2.2 Derivation of the Parity Vector Relation: In normal operating conditions (no failures) we have

$$y(s) = P(s)\, u_d(s) \tag{4.1}$$

where y and u_d are $p \times 1$ output and $m \times 1$ input vectors respectively. $P(s)$ is the system transfer function matrix which can always be expressed in terms of a right stable coprime factorization (D, N) or left stable coprime factorization $(\tilde{D}, \tilde{N})$ [9] as:

$$P(s) = N(s)\, D^{-1}(s) = \tilde{D}^{-1}(s)\, \tilde{N}(s) \tag{4.2}$$

The *generalized parity vector* (GPV) [8] can be defined directly from (4.1) and (4.2) as

$$p(s) = \tilde{D}y - \tilde{N}u_d\,; \tag{4.3}$$

this vector is said to exist in the *generalized parity space* (GPS). Substituting (4.2) into (4.1) and multiplying by $\tilde{D}$ shows that the GPV is a p-dimensional vector of rational functions (where p is the number of sensors) which is zero under ideal conditions, i.e., when the plant is linear, noise-free, and there are no sensor or actuator failures. Under normal operating conditions, $p(s)$ is a time-varying function of small magnitude due to the presence of noise and modeling errors arising from linearization and order reduction. However, when failures occur, $p(s)$ takes on a relatively large magnitude representing inconsistencies among the actuator inputs and sensor outputs with respect to the unfailed model. Different failures produce parity vectors with different characteristics. Thus, the generalized parity vector p may be used as a *signature-carrying residual* for FDI. We note that it was shown in [8] that the GPV-based method is equivalent to FDI using failure detection filter methods (see [8] for citations), in the sense that a failure detection filter can be designed so that its output is identical with the GPV (4.3).

4.2.3 Failure Isolation based on GPV Direction: The basic idea is that each actuator or sensor failure results in "activity" of the GPV along certain axes or in certain subspaces of the GPS. This information can be used to isolate the failure.

Actuator FDI - We first consider i^{th} actuator failures modeled by an additive error vector a via

$$u = u_d + a(t);\ a_i(t) \not\equiv 0;\ a_j = 0,\ j \neq i \tag{4.4}$$

Substituting (4.4) into (4.3), and noting that $\tilde{D}y - \tilde{N}u_d = 0$, we get

$$p_{a_i}(s) = -\tilde{N}^i a_i(t) \tag{4.5}$$

where $\tilde{N}^i$ denotes the i^{th} column of $\tilde{N}$.

Sensor FDI - Suppose the i^{th} sensor has failed, as above. Following the same procedure, we obtain

$$p_{s_i}(s) = \tilde{D}^i s_i(t) \tag{4.6}$$

where $\tilde{D}^i$ denotes the i^{th} column of $\tilde{D}$.

The domain of the activity of the parity vector under the assumption that the i^{th} actuator or sensor has failed depends on the number of non-zero elements in $\tilde{N}^i$ or $\tilde{D}^i$, respectively. The GPV algorithm can always be designed so that the GPV lies along a vector or in a plane or higher-dimensional subspace of the GPS [8]; this information can be used for failure isolation by projecting the parity vector onto the appropriate subspaces as shown in Section 6 of [8].

4.3 Application to the GE-21 Turbine Engine

The generalized parity space (GPS) approach was used to formulate two failure detection and isolation algorithms for the GE-21 engine [8]. A single-failure detection filter was designed using the stable factorization approach to generate the generalized parity vector. Detection was carried out by monitoring the magnitude of the parity vector, and isolation was based on parity vector direction in the GPS. In both algorithms, actuator failure isolation was accomplished by the steady-state GPV direction method; sensor failure isolation was carried out using the steady-state GPV direction method for the bias error case and by the GPV subspace projection method when the errors are time-varying. These approaches are based on the developments of [8] outlined above.

The general description and stable factorization model of the GE-21 engine were given in Section 3 of [8]. The engine model portrayed in schematic form in Fig. 3 is three-input/three-output; it was linearized to obtain a model in the form (4.1), and the above FDI

scheme used to to detect and identify failures in 6 components: 3 actuators and 3 sensors The three sensors measure the variables N2 (low-pressure rotor speed), N25 (high-pressure rotor speed), and PS3 (compressor discharge pressure); the three actuator servo outputs are WF36 (fuel flow), STP48 (low-pressure turbine stator position), and A8 (outer nozzle area) The actuator servo loops and sensors are assumed to be instantaneous in this study.

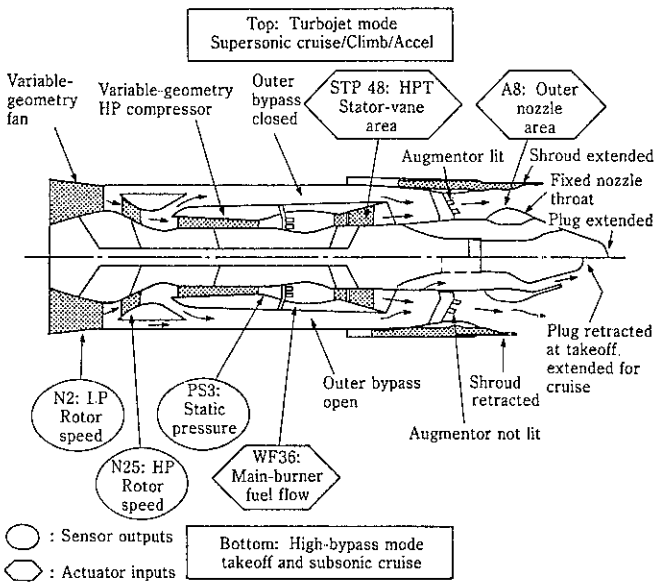

Figure 3. Inputs and Outputs in the GE-21 Engine

4.3.1 Generalized Parity Relation for the GE-21 Engine: In the GE-21 example [8] we used an extended parity relation similar to (4.6), i.e.,

$$p(s) = J(s)\,[\,\tilde{D}(s)y(s) - \tilde{N}(s)u_d(s)\,] \qquad (4.7)$$

where $J(s)$ is any stable, rational, proper matrix of transfer functions introduced simply to add another "degree of freedom" so that the parity vector form (and thus failure isolation) can be simplified A solution for J was determined in [8] so that the following failure signatures are obtained:

a. Suppose a failure occurs in the i^{th} actuator Then the GPV generated due to this failure lies along a distinctive vector in the GPS

b. Suppose a failure occurs in the i^{th} sensor Then the GPV generated due to this failure lies in a distinctive plane (Note: the GPS is 3-dimensional; also, it happens that the engine model has a particular form that causes the GPV for sensor three failures to lie along a vector)

Projecting the GPV onto vectors or planes to isolate its domain of activity unambiguously isolates single sensor or actuator failures

4.3.2 Simulation Results: In all of the FDI studies performed, we induced failures in the nonlinear simulation model that correspond to sensor or actuator failures as defined in Fig 2 (represented by $s_i(t)$ or $a_i(t)$ respectively) In some cases the actual input signals to the engine actuator loops were constant and the engine initial conditions were chosen to correspond to steady state; therefore, if there was no failure, the state variables and sensor outputs were also constant In other cases, sinusoidal inputs were applied to the system. The failures introduced via the actuator or sensor error signals were of two types: The "standard fault" was a bias error that ramped up to a steady-state value over a one-second interval The steady-state bias value was determined by taking a fixed percentage of the nominal (unfailed) value, e.g., 10 or 15 percent. The second fault considered was the injection of a sinusoidal error signal having a period (2 seconds) that is considerably longer than the time constants of the engine (about 0.4 seconds). This variety of faults allowed us to study FDI methods using several GPV-based algorithms. The results are summarized in [8]

These error models were used to generate measurement and input data for the FDI algorithms presented above The FDI algorithm was coded in SIMNON so that its outputs were the magnitude of the GPV (used for error detection) and six angles indicating the alignment of the GPV with the reference directions for actuator failures and the failure of sensor 3 or reference planes for the failure of sensor 1 or 2 This data was interfaced to the rule-based real-time system input processor, for interpretation by the supervisor

A representative case is portrayed in Fig 4. This simulation corresponds to a sinusoidal error in actuating the fuel flow (actuator 1; see time history plot 1 in the upper left panel) starting at $t = 0.5$ sec The rule-based system cycled every quarter of a second (meta-sample interval = 0.25 sec.) The GPV magnitude (lower left panel of Fig 4) rose rapidly, resulting in a failure detection at $t = 0.75$ sec The angle signals (determining the alignment of the GPV with the vector or plane corresponding to each possible actuator or sensor fault) are declared to be in a transient condition at $t = 0.75$ sec., and to have reached steady-state at $t = 1.0$ sec. The diagnosis (that actuator 1 has clearly failed) is made at the second steady-state meta-sample, $t = 1.25$ sec

5. SUMMARY AND CONCLUSION

In executing this research project, we developed a definition of a rule-based real-time control implementation environment (rule-based system capabilities and architecture), and thereby determined specific useful roles that expert systems technology can play in combination with conventional control technology including failure detection and isolation algorithms and

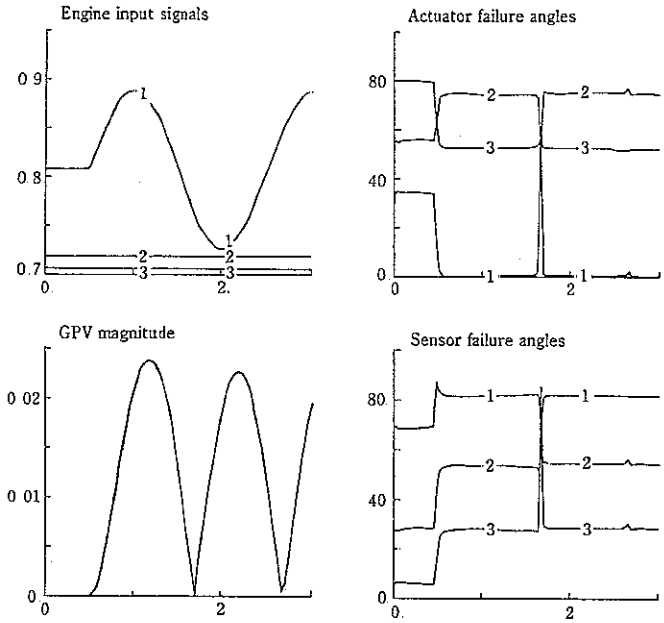

Figure 4. Simulated Rule-Based Failure Diagnosis

other traditional algorithms for estimation, identification, adaptation, etc. This set of concepts (Sections 2 and 3), we believe, together with the illustrative example presented in Section 4, provides further substantiation for the now-familiar claims that AI technology can be of substantial value in real-time control applications.

It is our position that the rule-based system methodology represents a high-level programming environment in which heuristic logic and "reasoning" can be implemented in a natural manner. This facilitates the development of rule-based systems that are modular, extensible, flexible, maintainable, etc. - as long as suitable discipline is followed. Heuristic logic has long been an integral part of working control systems; this methodology thus complements conventional control technology to make control system implementation less difficult and time-consuming.

Some of the standard misgivings regarding the use of rule-based systems in real-time control applications are not as fundamental as they are sometimes portrayed to be. For example, very complicated control systems with a preponderance of logic (including heuristics in most cases) have been fielded for several decades. Whether or not they may be rule-based, the control engineer must be concerned with the system's ability to run in real time, to finish data processing in time to accept the next input data sample, and to do the right thing in every circumstance; this is not an easy task in any case. At most, the difference between a conventional implementation and an expert system appears to be quantitative, not qualitative in these respects. Therefore, for example, it is no more reasonable to demand a "proof of stability" for a rule-based real-time control system than is is to demand the same for a comparably complex conventional one.

Several extensions to the methodology described above have been considered. The use of fuzzy logic to achieve better decision-making in the face of uncertainty has been explored in a preliminary fashion [8]. The use of higher-level knowledge representations compared with production rules - for example, frame-based systems - would seem to provide a better way to treat reasoning about time and causality; we have not experimented with this idea. Much remains to be done and learned in this research area.

Acknowledgement: The third author (ECL) wishes to acknowledge financial support received from the GE Corporate R & D Control Technology Branch during the period of this investigation.

REFERENCES

1. K. J. Åström and J. J. Anton, "Expert Control", *Proc. IFAC World Congress*, Budapest, 1984.
2. K. J. Åström, "Adaptation, Auto-Tuning, and Smart Controls", *Proc. Chemical Process Control III*, Asilomar, CA, 1986.
3. J. R. James, J. H. Taylor, and D. K. Frederick, "An Expert System Architecture for Coping with Complexity in Computer-Aided Control Engineering", *Proc. Third IFAC Symposium on Computer-Aided Design in Control and Engineering Systems*, Lyngby, Denmark, August 1985.
4. H. Elmqvist, "SIMNON - An Interactive Simulation Program for Nonlinear Systems", *Proc. Simulation '77*, Montreux, 1977.
5. J. H. Taylor and D. K. Frederick, "An Expert System Architecture for Computer-Aided Control Engineering", *Proc. of the IEEE*, December 1984.
6. J. R. James, D. K. Frederick, and J. H. Taylor, "On the Application of Expert Systems Programming Techniques to the Design of Lead / Lag Precompensators", *Proc. Control '85*, Cambridge, UK, July 1985; to appear in *IEE Proceedings D. Control Theory and Applications*, 1987.
7. R. W. Gerhardt, "An Expert System for Real-Time Control", MSc Thesis, RPI, Troy NY, May 1986.
8. N. Viswanadham, J. H. Taylor, E. C. Luce, "A Frequency-Domain Approach to Failure Detection and Isolation with Application to GE-21 Turbine Engine Control Systems", *Control Theory and Advanced Technology*, Mar. 1987.
9. Vidyasagar, M., *Control System Synthesis: A Factorization Approach*, MIT Press, Cambridge, MA, 1985.