Real-Time Knowledge-Based Systems

Thomas J. Laffey, Preston A. Cox, James L. Schmidt, Simon M. Kao, & Jackson Y. Read

Real-time domains present a new and challenging environment for the application of knowledge-based problem-solving techniques. However, a substantial amount of research is still needed to solve many difficult problems before real-time expert systems can enhance current monitoring and control systems. In this article, we examine how the real-time problem domain is significantly different from those domains which have traditionally been solved by expert systems. We conduct a survey on the current state of the art in applying knowledge-based systems to real-time problems and describe the key issues that are pertinent in a real-time domain. The survey is divided into three areas: applications, tools, and theoretic issues. From the results of the survey, we identify a set of real-time research issues that have yet to be solved and point out limitations of current tools for real-time problems. Finally, we propose a set of requirements that a real-time knowledge-based system must satisfy.

As the application of knowledge-based systems evolves from an art to an engineering discipline, we can expect more challenging applications to be addressed. Some of the most challenging and interesting environments are found in real-time domains.

A knowledge-based system operating in a real-time situation (for example, crisis intervention or threat recognition) will typically need to respond to a changing task environment involving an asynchronous flow of events and dynamically changing requirements with limitations on time, hardware, and other resources. A flexible software architecture is required to provide the necessary reasoning on rapidly changing data within strict time requirements while it accommodates temporal reasoning, nonmonotonicity, interrupt handling, and methods for handling noisy input data.

Real-time computer systems have become an integral part of our everyday life. As discussed by Wright et al. (1986) they are being used in a growing number of applications ranging from small, simple controllers found in common household appliances to large, complex systems for industrial and military purposes. The complexity of these systems is increasing rapidly along three dimensions: (1) the number of functions controlled, (2) the rate at which the functions must be controlled, and (3) the number of factors that must be considered before a decision can be made. The increasing complexity has caused considerable interest in the use of knowledge-based techniques for real-time applications. Proper application of these problem-solving techniques can result in more sophisticated strategies for complex applications such as satellite control, autonomy, and data analysis; the Pilot's Associate; autonomous vehicles; battle management; aerospace systems (for example, monitoring electric, power, propulsion, and life-support systems); communications network monitoring and control; robotics and vision systems; process control; financial advice (for example, a market monitor, adviser, or trader); and medicine (patient monitoring).

Why Use a Real-Time Expert System?

As discussed by Turner (1986), the "principal reason for using a real-time expert system is to reduce the cognitive load on users or to enable them to increase their productivity without the cognitive load on them increasing."

Indicators that a real-time expert system might be appropriate, especially when conventional techniques have failed or are impractical, include problem-solving situations where humans suffer from cognitive overload, fail to effectively monitor all available information, are unable to resolve conflicting constraints, are expensive or scarce, make high-cost mistakes, miss high-revenue opportunities, cannot simultaneously manipulate all the relevant information to obtain optimal solutions, or cannot provide a solution quickly enough.

As shown earlier, many reasons explain why real-time domains could benefit from the capability to capture and apply expert knowledge that expert systems offer. For example, an operator on an oil platform can be confronted with 500 analog and 2500 digital signals, resulting in a considerable cognitive load in the event of a system problem. For the future, oil platform control rooms are being planned that will make available as many as 20,000 signals for just two or three operators (Sachs, Paterson and Turner 1986). In other domains, such as satellite operation and control, qualified personnel are becoming increasingly difficult to find, especially those who are able to evaluate complex situations and recommend actions (Krutchen 1986). Similarly, in the world of stock market and foreign currency exchange, good traders who can quickly assimilate and evaluate information and act on it are scarce and expensive (Behan and Lecot 1987). The stock market is also one of many domains where the sheer volume and complexity of information make it difficult for a human to make correct, timely decisions.

Many of the domains described earlier have problem areas that could benefit from the application of a real-time knowledge-based system. Our main goal in writing this article is to examine the current state of the art in real-time expert systems and indicate areas where future research is needed. First, we briefly discuss what "real time" means and how real-time processing differs from nonreal-time processing. We then present the results of our survey of real-time expert system applications, followed by concluding remarks. Our conclusions describe the limitations of current tools for real-time problems and list a set of requirements that a real-time expert system tool should offer.

Defining "Real Time"

The term real time is often easier to recognize than to define. For example, no one would call real time a computer program that spends the night updating bank accounts based on a tape of recorded transactions. However, most people would probably agree that a computer program which automatically pilots an aircraft is real time.

As discussed by O'Reilly and Cromarty (1985), many definitions of real time exist. Perhaps the most common usage of the term real time is "fast," in the sense that a system is considered real time if it processes data quickly. Another common definition is that real time means "perceptually fast" or, at least, "faster than a human can do it." A better definition of real time states that the "the system responds to incoming data at a rate as fast or faster than it is arriving." Although scarcely precise, this definition does link the concept of real time to problem-relevant performance measures.

The literature also offers the following two formal definitions of real-time: (1) a system exhibits real time behavior if it is "predictably fast enough for use by the process being serviced" (Marsh and Greenwood 1986) and (2) "there is a strict time limit by which the system must have produced a response, regardless of the algorithm employed" (O'Reilly and Cromarty 1985).

Response time is the time the computer takes to recognize and respond to an external event. This measure is the most important in real-time applications; if events are not handled in a timely fashion, the process can literally go out of control. Thus, the feature that defines a real-time system is the system's ability to guarantee a response after a fixed time has elapsed, where the fixed time is provided as part of the problem statement. If, given an arbitrary input (or event) and an arbitrary state of the system, the system always produces a response by the time it is needed, then the system is said to be real time.

Why Is Real Time Different?

Traditionally, knowledge-based problem-solving techniques have been applied in domains where the data are static, and no time-critical responses are required. Many of the early diagnostic, design, and configuration systems exhibit these characteristics. AI researchers find in real-time domains a new set of complex problems, such as those outlined in the following paragraphs.

Nonmonotonicity: Incoming sensor data, as well as facts that are deduced, do not remain static during the entire run of the program. The data are either not durable and decay in validity with time, or they cease to be valid because events have changed the state of the system.

Continuous operation: Many real-time systems continue to operate until stopped by an operator or some catastrophic external event. The discovery of a partial or complete failure of one or more parts of the system does not necessarily imply that the system will stop functioning. A real-time system monitor or controller must be capable of continuous operation, which also implies that close attention must be paid to the issue of garbage collection.

Asynchronous events: Much of the activity in a real-time system conforms to a schedule; that is, real-time systems process data in an orderly and predictable manner. However, in many real-time systems, events can occur that fail to conform to any schedule. Thus, a real-time system must be capable of being interrupted to accept input from an unscheduled or asynchronous event. Additionally, events can vary in importance. A real-time system must also be capable of processing input according to importance, even if the processing of less important input must be interrupted or rescheduled.

Interface to external environment: A real-time expert system typically needs to gather its data from a set of sensors. Traditionally, expert systems have asked the operator for input.

Uncertain or missing data: The validity of the data can decay with time. However, data can also lose validity or have questionable validity because of a degradation in sensor performance. Thus, a real-time system must be able to recognize and appropriately process data of uncertain or diminished validity.

High performance: High performance is a key issue in many real-time systems. For instance, response times in the Lockheed Pilot's Associate program vary from 500 milliseconds for assessing target values to 100 milliseconds for defining threat objects. Given the performance limitations of AI, the problem of controlling complex systems by responding to rapidly changing data is a critical bottleneck in applying knowledge-based systems to this domain.

Temporal reasoning: Time is an important variable in real-time domains. Typically, a real-time system needs the ability to reason about past, present, and future events, as well as the sequence in which the events happen.

Focus of attention: When a significant event occurs, it is important that the real-time system be able to focus its resources on important goals. This focus can involve bringing a new knowledge source (a specialist) to bear; modifying the set of sensors the system is currently looking at; and possibly, changing the rate at which the data are being analyzed.

Guaranteed response times: The system must be able to respond by the time the response is needed. Furthermore, one would like to produce the best possible response within the given deadline.

Integration with procedural components: A real-time knowledge-based system must typically be integrated with conventional real-time software. The conventional code will perform tasks such as data compression, signal processing, feature extraction, and application-specific input-output (I/O).

In the following sections, we examine the current state of the art in applying knowledge-based problem-solving techniques to real-time domains.

Survey Results

The survey results are grouped into applications, tools, and theory. Reusability is the primary criterion for distinguishing between applications and tools. An expert system developed to solve a specific problem is considered an application. An expert system developed to solve many problems is a tool. Applications developed using these expert systems are described in the tools section. Finally, we also examine theoretical issues for real-time domains that have been addressed by AI researchers.

Real-Time Expert System Applications

Applications have been divided into the categories of aerospace, communication, financial, medical, process control, and robotic systems. A summary of the real-time research issues that were addressed in each of the applications can be found in table 1.

Aerospace

The following paragraphs summarize applications of knowledge-based systems to real-time aerospace problems.

The Emergency Procedures Expert System (EPES). EPES was designed to be one of the cooperating expert systems that would make up a Pilot's Associate. EPES works in the domain of in-flight emergency procedures to detect an emergency, warn the pilot, and initiate corrective action. The system can be overridden by the pilot at any time. A prototype of EPES was implemented in ZetaLisp on a Lisp machine. The knowledge base for EPES includes parts, goals, and rules and implements emergency procedures from an F-16 flight manual. EPES interfaces with the pilot using an expert message display, as well as a simulated aircraft control system (Anderson et al. 1984).

The Expert System for Satellite Orbit Control (ESSOC). ESSOC was designed to assist in a satellite's station-keeping (delta V) maneuvers by continuously processing satellite telemetry data and determining the appropriate commands to execute a successful maneuver. ESSOC analyzes data during routine satellite status checks, diagnoses the cause of anomalies, and recommends commands to resolve the anomalies. The expert system dynamically communicates with a real-time, high-fidelity satellite simulation operating on a VAX 11/785. ESSOC buffers data received from the satellite simulator and checks for the presence of buffered data prior to executing each traditional production system recognize-act cycle. ESSOC was developed in ART on a Symbolics 3675 (Rook and Odublyl 1986).

The Expert System for Inertial Measurement Unit (EXIMU). Real-time monitoring of telemetry data from the space shuttle's inertial measurement unit (IMU) is the domain of the EXIMU expert system. EXIMU partitions the solution according to operating modes of IMU and problem-solving strategy. Currently, EXIMU modules address the preflight calibration and alignment operating modes of IMU. Mainline surveillance uses forward-chaining search rules for evaluating departures from nominal expectations. EXIMU is microcomputer based. An interrupt-driven telemetry interface operating in the background accepts incoming characters and stores them in predefined memory locations. An algorithmic program retrieves the numeric values and does the number crunching. The inference engine then tests the rules and performs the indicated actions (Campbell 1987).

Expert Navigator. The Expert Navigator was designed to monitor, manage, and reconfigure navigation sensors (such as radio aids, inertial navigation systems, and digital terrain aids) on board an advanced tactical aircraft. The expert system monitors the ability of the navigation sensors to support the aircraft's primary mission and suggests alternatives when the mission becomes threatened. The system's knowledge is represented in rules operating in a blackboard architecture. The system was implemented in Lisp on a Symbolics 3600 processor (Pisano and Jones 1984).

The Flight Expert System (FLES). FLES prototype is designed to assist an airplane pilot in monitoring, analyzing, and diagnosing faults in airplane systems. The system is based on the Hearsay blackboard model. Knowledge sources are sensor-interrupt analyzers that receive interrupts indicating sensor faults, construct hypotheses about the causes of the interrupts, and place the hypotheses on the blackboard. When a verification procedure confirms a component is inoperative, a diagnostic procedure determines if the component is the source of the fault or a side effect of some other fault. The diagnostic process is nonmonotonic in that the absence of information about a suspect component is used in the reasoning process. The acceptable tolerance for sensor data varies according to the flight phase. Thus, a different monitor with different tolerances is used for monitoring the sensors in each flight

AEROSPACE

Application	High Performance	Guaranteed Response	Temporal Reasoning	Asynch. Inputs	Interrupt Handling	Continuous Operation	Noisy Data	Focust of Attention
AIRPLAN	•		•	•	•	•	•	•
EPES								•
ESSOC	•			•			•	
EXIMU	•			•	•		•	
Expert Navigator							•	
FLES				•	•		•	
HEAT	•			•		•	•	•
LES			•				•	
L*STAR	•		•	•		•		•
Malfunction Recovery								
NAVEX	•		•	•			•	
Pilot's Associate	•		•	•				•
PREMON			•				•	•
SCARES	•							•
SECURE								•

COMMUNICATIONS

Application	High Performance	Guaranteed Response	Temporal Reasoning	Asynch. Inputs	Interrupt Handling	Continuous Operation	Noisy Data	Focust of Attention
ECESIS	•		•	•	•	•	•	•
ERIK (f)	•	•		•			•	•
HANNIBAL								•
News Wire Monitoring							•	•

MEDICAL

Application	High Performance	Guaranteed Response	Temporal Reasoning	Asynch. Inputs	Interrupt Handling	Continuous Operation	Noisy Data	Focust of Attention
BABY							•	
CAPS								
EEG Analysis	•						•	
FORTES								
Polysomnographer							•	•
VM			•				•	•

PROCESS CONTROL

Application	High Performance	Guaranteed Response	Temporal Reasoning	Asynch. Inputs	Interrupt Handling	Continuous Operation	Noisy Data	Focust of Attention
ALFA	•							
CEALMON & REALM								
COOKER (f)	•			•			•	•
Diagnostic E.S. (f)			•			•	•	
ESCORT	•	•					•	•
Falcon			•					
LMA			•					
MCM			•	•	•			•
REACTOR								•
Rotary Cement Kiln Supervixor (f)	•		•	•			•	
STOCHASM			•					
YES/MVS (f)	•		•	•		•		

ROBOTICS

Application	High Performance	Guaranteed Response	Temporal Reasoning	Asynch. Inputs	Interrupt Handling	Continuous Operation	Noisy Data	Focust of Attention
Autonomous Vehicle Guidance (f)	•		•	•			•	•
HERMIES (f)	•	•	•	•	•		•	•
Ping Pong Player (f)	•	•	•		•	•	•	

Table 1. Summary of Applications.

phase. FLES is implemented as a prototype operating in a simulated environment. FLES is coded in FranzLisp and runs on a VAX computer (Ali and Scharnhorst 1985).

The Heuristic Error Analyzer for Telemetry (HEAT). HEAT analyzes space shuttle telemetry data that are collected and downlinked to a ground station every second. Once validated, the telemetry data are then transmitted to NASA mission controllers. HEAT distinguishes between routine telemetry errors caused by a loss of signal as the shuttle moves out of range and a more serious problem requiring human attention. HEAT was implemented in a FranzLisp version of OPS5 on a VAX 11/780 operating under Unix. To accommodate real-time demands of the incoming telemetry status data, a separate C program was written that translates the telemetry status data into a Lisp-compatible format which is then inserted into a queue called the AI log file. A separate Lisp program intervenes to cause the OPS5 code to execute a specified number of rules and then stop and check the log file for additional data. Messages are read either one at a time or in clusters of 20 or more depending on how far HEAT is behind in processing. Because the shuttle is out of range 12 minutes every hour, HEAT was also designed to purge unused data when there is nothing else to do, thus minimizing the impact on memory usage. Based on trial runs with simulated data, the developers concluded that OPS5 is not the optimal system architecture for real-time data processing. One reason given is that the RETE algorithm used in OPS5 was designed for environments where the knowledge base changes slowly (For an alternative view of Rete's applicability to real-time domains, see Haley [1987]). In real-time data analysis problems, the knowledge base changes too rapidly for the RETE algorithm to be optimal (Skapura and Zoch 1986).

The Liquid Oxygen Expert System (LES). LES monitors measurements from the launch-processing system during loading of liquid oxygen into the space shuttle. The measurements come from a real-time process controller and include values such as temperature, pressure, flow rate, and valve position. LES monitors using a model consisting of about 1500 frames. It is one of the largest model-based systems built to date. LES determines whether the sensor data indicate an anomaly and notifies monitoring personnel and initiates troubleshooting procedures. If LES cannot isolate the problem, it provides a list of suspected components and recommends further steps to isolate the fault. LES is implemented in ZetaLisp on a Symbolics computer (Scarl, Jamieson, and Delaune 1984).

The Lockheed Satellite Telemetry Analysis in Real Time (L*STAR). L*STAR is a knowledge-based system for performing real-time monitoring and analysis of telemetry data from the NASA Hubble space telescope (HST). In order to handle asynchronous input and perform in real time, the system consists of three or more separate processes that run concurrently and communicate by way of a message-passing scheme and can reside on the same or different computers. The data management task gathers, compresses, and scales the incoming telemetry data before sending them to the other tasks. The inferencing task consists of a proprietary high-performance inference engine written in C. It uses the telemetry data to perform a real-time analysis of the state and health of HST. The I/O task receives telemetry monitors from the data management task, updates its graphic displays in real time, and acts as the interface to the console operator. It was implemented on a set of VAXes and MicroVAXes and uses DECNET mailboxes as the message-passing mechanism. The knowledge-based system has been interfaced to a real-time simulator (Kao et al. 1987).

Malfunction Recovery. This system consists of a self-organizing controller that learns how to pilot a simplified two-dimensional aircraft model by continuously monitoring the model's position and velocity as the model is "flown" through a navigational mission. When trained, the system can pilot the model between two points without exceeding preestablished position and velocity limits by activating any of eight directional actuators, even when two of the actuators permanently fail. The system demonstrates "self-organizing control, learning, and malfunction recovery" (Cruz 1986; Vidal 1985).

The Navigation Expert (NAVEX) System. NAVEX was developed to support reentry navigation for the space shuttle program. NAVEX makes high-speed decisions about shuttle velocity and trajectory. The system was developed for use during reentry but could easily be modified to support launch as well. NAVEX was developed in ART and Lisp and operates on a Symbolics computer. Radar data and output from the high-speed trajectory determinator are monitored by NAVEX, which then warns the operator of current or impending problems and recommends possible actions. In tests with data generated by a previous shuttle flight, Marsh and Healey report NAVEX made the correct decisions 100 percent of the time and made its decisions faster than the human controllers (Marsh 1984; Healey 1986).

Pilot's Associate. In this system, Lockheed has interfaced symbolic processors to a flight simulator. The system consists of a VAX-11/780 driving a full, six-degree-of-freedom simulator linked by Ethernet to three Symbolics Lisp machines. On the Lisp machines are expert systems performing situation assessment, tactics and route planning, and intelligent pilot-vehicle interface. Broadwell and Smith (1986) report the system runs in "near real-time."

The Predictive Monitoring System (PREMON). PREMON uses the explicit model of a device to perform real-time monitoring. The system has been tested on the partial model of the mirror cooling circuit of the Jet Propulsion Laboratory space simulator. PREMON has three interacting capabilities: (1) causal simulation to generate predictions about the behavior of a physical system, (2) sensor planning to assess the importance of a device's behavior and allocate sensor resources appropriately, and (3) sensor interpretation to verify expected sensor values against actual sensor readings and raise alarms when necessary (Doyle, Sellers, and Atkinson 1987).

The Spacecraft Control Anomaly Resolution Expert System (SCARES). SCARES attempts to automate the diagnosis of anomalies in the attitude control system of a spacecraft with the ultimate goal of achieving spacecraft autonomy. SCARES has three major components, only one of which performs in real time. The SCARES monitor performs three real-time checks on downlinked telemetry data: a limit check on individual telemetry points, a rate check on two or more telemetry points in the same channel, and cross-channel checking tests for telemetry consistency. Hamilton (1986) concludes that SCARES, which is currently in a prototype stage, shows "how expert system technology can be applied in large real-time fault diagnostic systems to detect, diagnose, and recover from complicated anomalies." SCARES is implemented in Lisp on a Symbolics 3670.

SECURE. In the domain of commercial transport aircraft, SECURE continuously observes the flight environment and evaluates the situation for possible faults that could threaten flight safety. Faults requiring further action are detected by comparing the correct aircraft state with the observed aircraft state. A recovery procedure is derived and patched into the current flight, and the crew is informed of the threat and the recovery procedure if the crew's actions have indicated that the threat has not been noticed. SECURE's knowledge base is frame based and is organized according to the flight contexts of the takeoff phase, flight phase, and landing phase. Only the knowledge appropriate to the current flight context is accessed. SECURE was developed using a script-based approach implemented in MACLISP on a PDP-10 computer and works well with a "normal" flight containing common emergencies. The first phase of the planning-based SECURE has successfully flown a simplified DC-10 aircraft simulation running on a PDP-10 between any pair of six airports and has planned recovery actions for several failures. About 230 rules describe the planning control knowledge (Chen 1985).

Communications

This section summarizes the application of knowledge-based systems to real-time communication problems.

The Environmental Control Expert System in Space (ECESIS). ECESIS was designed for use on a manned space station to shift modes for various subsystems of an environmental control-life support subsystem (EC-LSS) as the station passes from shadow to sun. ECESIS monitors the EC-LSS and performs various actions in response to detected events. The system is implemented in Yet Another Production System (YAPS) and incorporates both rule-based and semantic net architectures (Dickey and Toussaint 1984).

Evaluating Reports Using Integrated Knowledge (ERIK). ERIK provides automated interpretation of daily ship reports from merchant ships throughout the world as part of the United States Coast Guard's Automated Mutual-assistance Vessel Rescue (AMVER) system. ERIK has several mechanisms that represent a partitioning of functions to achieve real-time performance. An interface filter performs some preliminary syntactic conversions such as the stripping of control characters from incoming messages, case conversion, and so on. A top-level controller aided by a keyword parser makes early determination of the content of the text. ERIK is required to interpret or reject a message in less than 60 seconds of processing time and, in fact, averages 19.35 seconds of central processing unit time per ship report. It is implemented in compiled Common Lisp on a VAX 11/785 (Hardt and Rosenberg 1986).

HANNIBAL. HANNIBAL interprets data from sensors that monitor radio communications to identify enemy organizational units and their communication order of battle. The data include information about the location of the detected communication and its signal characteristics. HANNIBAL uses a blackboard architecture and is implemented in AGE (Brown et al. 1982).

News Wire Monitoring. A knowledge-based system scans news wire text in real time and triggers an alarm when topics previously chosen by a user are found. A natural language interface allows the user to specify topics. The prototype system filters incoming text with a simple keyword search and then parses sentences further to derive primitive knowledge structures corresponding to user selections. The system was implemented on a Symbolics computer (Clippinger 1983).

Financial

Surprisingly enough, real-time finan-

... the feature that defines a real-time system is the system's ability to guarantee a response after a fixed time has elapsed, where the fixed time is provided as part of the problem statement.

cial applications are not frequently published. This fact might be because most financial institutions are extremely secretive about their automated analytic trading programs. Firms known to be involved in intelligent real-time trading systems include American Express, Bear Stearns, William Blair, Drexel Burnham Lambert, First Boston, Merrill Lynch, Morgan Stanley, Shearson Lehman, Solomon Brothers, Thomas McKinnon Securities, and the major banking and insurance companies. For more discussion on AI applications of security trading, see Szuprowicz (1987).

Another survey (Behan and Lecot 1987) found several expert system applications in the areas of financial planning, intelligent text understanding, portfolio management, trading, intelligent customer service assistance, insurance underwriting, credit evaluation, and auditing. However, these expert systems appear to be traditional, user-interactive decision aids.

Behan did describe a couple of AI-based systems that border on real-time domains. For example, a natural language parser developed by Arthur D. Little, called the NL Parser, analyzes telex messages from Reuters and United Press International that relate to mergers and acquisitions. Another system, called the Trader's Assistant, also developed by Arthur D. Little, provides assistance to security traders by assessing such factors as the instantaneous supply and demand of the stock market and the significance of current rumors. Both systems are written in Lisp and run on Symbolics Lisp machines.

Of the categories surveyed by Behan, trading seems to offer the highest potential payoff for a successful real-time knowledge-based system. As Behan points out, trading, especially foreign exchange trading, "is a fast moving activity that depends on a wide and complex scope of political, economic, climatic, and financial factors." Furthermore, "large profits are possible by only marginally improving the success ratio on trading transactions." A real-time expert system capable of directly receiving and evaluating stock market and other financial quotes seems attractive.

Medical

This section reviews the application of knowledge-based systems to real-time medical problems.

BABY. BABY is a forward-chaining, rule-based system that monitors infants in a newborn intensive care unit. The system monitors data, looks for significant patterns, and suggests further evaluations. BABY also tracks the clinical status of the newborns and can answer questions about each patient. BABY uses a Bayesian probabilistic method for handling uncertainty, similar to the method used in PROSPECTOR. The system is written in PASCAL (Rodewald 1984).

CAPS. CAPS is a system that combines a pattern-recognition module with an expert system module to perform respiratory and anesthesia monitoring. The pattern-recognition module characterizes and analyzes each segment of a capnogram. (A *capnogram* is a carbon dioxide wave form produced by monitoring patient-respired gases during surgery.) The analysis of the capnogram is then used by the expert system module to generate a probable diagnosis and recommend therapy or equipment adjustments. The system was implemented using the EXSYS tool on an IBM PC/AT. Rader, Crowe, and Marcott (1987) report that the speed of the system was not acceptable.

The EEG Analysis System. In this system, an expert system was developed to apply fast Fourier transforms to data received directly from electroencephalograms connected to renal patients. The system is rule based and uses certainty factors. It was written in C and embedded in a Motorola MC 6801 microprocessor (Baas and Bourne 1984).

A Forth Oriented Real-Time Polysomnographer Expert System (FORTES). FORTES was developed at the Stanford University of Medicine to perform real-time sleep staging. *Sleep staging* is a means of classifying a person's state while asleep. It is usually performed by a trained human scorer who evaluates a linear time-varying graph of the electroencephalograms, electrocardiogram, and eletromyographic activity while a person is sleeping. FORTES is implemented on an IBM PC and is interfaced to a polygraph with a standard digital to analog converter (Redington 1986).

The Ventilator Manager (VM). VM was one of the first and most significant applications of knowledge-based systems to a real-time domain. VM interprets online physiological data in an intensive care unit. An automatic monitoring system provides VM with the values of 30 physiological measurements at 2- or 10-minute intervals. The data are used to manage postsurgical patients receiving mechanical ventilatory assistance. VM maintains a set of patient-specific expectations and goals for future evaluation. It is expectation driven and uses the current and past patient history to establish guidelines for patient measurements. The guidelines are used to dynamically establish upper and lower limits for comparison with each new measurement from the monitoring system. VM responds with suggestions to clinicians and periodic summaries. VM uses forward chaining, checks that information previously acquired is still valid for making conclusions, and cycles through the rule set each time new information is available (Fagan, Kunz, and Feigenbaum 1979; Fagan 1980).

Process Control

This section reviews the application of knowledge-based systems to real-time process-control problems.

The Automated Load Forecasting Assistant (ALFA). ALFA provides an electric utility company in New York with hourly predictions of load as much as 48 hours in advance. ALFA is written in Lisp and runs on a large IBM mainframe computer. The rule base was derived from the expertise of load forecasters and considers variables such as load growth, hour of the day, day of the week, day of the year, holidays, special events, and prevailing weather conditions. Weather data are obtained directly from the National Weather Service. A real-time pattern-matching algorithm searches a 10-year database for eight hourly load entries that best match the predicted weather pattern. ALFA generates its forecasts in 20 seconds, compared to the two hours required by a human load forecaster. As a result, forecasts can be generated at any time, and what if scenarios can be explored (Jabbour et al. 1986).

The Computerized Emergency Action Level Monitor (CEALMON) and the Reactor Emergency Action Level Monitor (REALM). CEALMON is a small expert system to do real-time monitoring in a nuclear power plant. It is coded in Gold Hill Common Lisp for the IBM PC and embodies a rule-based problem-solving paradigm using forward-chaining logic. Sensor data have been simulated by reading from a file at regular intervals. The main purpose of the system is to make the operator aware of changes in the status of emergency action levels and give an explanation.

A successor to CEALMON, called REALM, is being developed in KEE and will interface with one of the plant computers in order to collect data (Touchton 1986).

COOKER. COOKER is a real-time process monitoring and operator advisory system for batch manufacturing processes. It runs on a Symbolics 3640 Lisp machine connected by an IBM AT microcomputer to a Honeywell TDC 2000 process control system and a programmable logic controller. COOKER has four main subsystems: data frames, data gatherer, operator interface, and inference engine. The latter three subsystems run as concurrent processes. The data gatherer sends data requests to, and buffers data from, the IBM AT. The operator interface manages all windows, displaying advice and questions to the operator, and receives replies. The inference engine handles unbuffering data from the data gatherer into data frames, receives replies from the operator, runs the monitoring and problem-recognition mechanism on the goals, and runs any problem solving required by the goals. The system uses a knowledge representation scheme called Goal/Subgoal (GSG). Allard and Kaemmerer (1987) report that COOKER has been installed in a manufacturing plant.

Diagnostic Expert System. Instrumentation and equipment for seven steam turbine generators at the Texas Utility Generating Company are being diagnosed each hour using the Diagnostic Expert System developed by Westinghouse Electric. About 110 different types of sensor data (from instrumentation and generator equipment) are transmitted by telephone line or satellite to the centralized diagnostic center in Orlando, Florida. The expert system then performs the diagnosis on a VAX-11/780 and transmits its recommendations to the plant (Osborne et al. 1985a).

The Expert System for Complex Operations in Real Time (ESCORT). ESCORT was designed to aid operators of information systems that produce large quantities of changing data. It is currently configured to aid process operators in oil production platform control rooms where as many as 500 analog and 2500 digital signals can confront the operator, a cognitive overload that is especially critical when problems occur. Partial shutdowns of platform operations can occur every half hour, and total shutdowns sometimes occur once a week. ESCORT provides advice on plant crises within 1 second and presents its advice in a simple and concise manner so the cognitive load on the operator will be reduced, not increased. ESCORT is implemented on a Xerox 1108 Lisp workstation running Interlisp-D and Loops and is connected by Ethernet to a PDP-11 running a simulation of part of a North Sea platform process plant and associated process control systems (Sachs, Paterson, and Turner 1986).

The Fault Analysis Consultant (Falcon). Falcon monitors and analyzes alarm signals in a chemical process plant. Falcon consists of five modules: supervisor, simulator, monitor, fault analyzer, and a human-machine interface module. The monitor module is time critical. It continuously examines process data and determines whether a disturbance exists. It converts the sensor data to a symbolic value (for example, low, normal, or high) and sends it to the fault analyzer. Trends and rates of change are computed from the data and also converted. The fault analyzer can use either a rule-based or model-based approach to find the cause of the disturbances (Chester, Lamb, and Dhurjati 1984).

The Logic Machine Architecture (LMA). LMA, developed by Argonne National Laboratory, is a package of Pascal subroutines that encapsulate functions required to perform reasoning. LMA is divided into four software layers: abstract data types, database support functions, inference mechanisms, and a theorem prover. Procedures residing in one layer have access only to those procedures in the next lower layer. LMA has been applied as a control system to keep the reactor-inlet temperature constant at the experimental breeder reactor nuclear power plant (Lusk and Stratton 1983).

The Materials Composition Management (MCM) System. The MCM system combines heuristic and analytic process control to address the problem of chemical manufacturing process control. The system is implemented on a Lisp machine and consists of the heuristic control virtual machine (HCVM) and an application shell. HCVM is a generic "event-driven object-oriented computational framework" for developing real-time heuristic control applications (D'Ambrosio et al. 1987). The architecture for HCVM resembles a blackboard architecture and consists of a control mechanism and a group of modules. Each module can contain either Lisp code, if-then rules, or another HCVM instantiation. The MCM communications handler manages all communication between MCM and the external world. Packet handlers time stamp and unbundle data packets received by the communications handler. Data handlers screen and store the data, which can trigger task handlers. A scheduler controls the execution of triggered task handlers. The application shell provides a tool set for monitoring, examining, and dynamically controlling module activity within HCVM (D'Ambrosio et al. 1987; Raulefs et al. 1987).

REACTOR. REACTOR assists nuclear reactor operators in the diagnosis and treatment of accidents. REACTOR is an expert system that monitors a nuclear reactor facility, detects deviations from normal oper-

Research which focuses on speeding up a version of the algorithm (or some derivative of it) that can guarantee response times should be a high priority.

ating conditions, determines the significance of the situation, and recommends an appropriate response. The knowledge base contains event-oriented knowledge and function-oriented knowledge. The event-oriented knowledge, expressed in if-then rules, describes expected behavior under known accident conditions and is based on experience with past accidents, experiments, and analyses of computer simulations. When an event occurs that does not match a known pattern, the function-oriented knowledge is used. The function-oriented knowledge represents the reactor system's configuration and the manner in which its components work together to perform a given function. The function-oriented knowledge is represented in a response tree and shows the success paths that can be used to provide a group of actions to prevent damage to the reactor core or a release of radioactivity. Forward chaining reasons from known facts to a conclusion. If no conclusion can be reached because of missing information, backward chaining determines what data are missing so instruments can be read, or an operator can be queried to obtain the missing data. REACTOR is implemented in Lisp and uses FORTRAN to provide color graphic displays for the operator. Response requirements for REACTOR are expressed in terms of "taxing the operator's patience" (Nelson 1982, 1984). Nelson (1984) also reports on an interesting survey on the application of expert systems to nuclear reactor operations.

Rotary Cement Kiln Expert System Supervisor. This expert system provides process control in cement manufacturing. It is encoded in York Portable Prolog and is installed on several computers, including an IBM PC. An interface was written to connect external inputs from RS-232 and A/D facilities to Prolog. The Rotary Cement Kiln Expert System Supervisor uses both a long-term and a short-term memory. The long-term memory contains rules, and the short-term memory contains data pertinent to the current state, which can be replaced in total or in part by new data as they are used. The operating mechanism consists of programs that use the current data by calling appropriate rules from the knowledge base plus a capability to generate new strategies under changing environmental conditions which confer the ability to supervise the process directly or through conventional control systems. A calculation tool module also exists that contains arithmetic operators and links to programs such as plant dynamic simulations of fault diagnosis routines (Norman and Naveed 1985).

STOCHASM. STOCHASM is a computer program for performing real-time fault detection and diagnosis for the lubrication oil subsystem in a gas turbine propulsion unit on a United States Navy surface ship. The system utilizes knowledge of the order in which alarms are triggered to diagnose the cause of malfunctions. Malkoff (1986) reports the system had learning capabilities that allowed it to automatically adapt to changes in the system and the environment. STOCHASM was interfaced to a simulator and works in real time. Several different approaches for performing temporal reasoning were developed and analyzed.

The Yorktown Expert System/Multiple Virtual Storage Manager (YES/MVS). YES/MVS is a real-time MVS computer operator aid implemented on three virtual memory machines operating the VM/370 operating system. It uses an augmented version of OPS5 written in MACLISP, with a compatibility package to permit use with Yorktown LISP. The modified version of OPS5 features compiled, rather than interpreted, executions; special LISP macros; and an efficient matching process to provide improved run-time performance. The OPS5 repetitive cycle phase was modified to include a "pickup" function during each cycle in which newly received messages are added to working memory. The OPS5 conflict-resolution strategy was modified to permit the use of explicit priorities so that when deciding which instantiations to fire, OPS5 will first consider those instantiations with the highest priority. YES/MVS makes time of day and time intervals available and provides a TIMED-MAKE LISP function that causes a specified element to be placed in working memory at a specified future time or after some specified time interval. This feature is good, for example, in periodically querying the state of various MVS resources and in sequencing actions that require delays between them. In one application of YES/MVS, a night operator's work was greatly reduced (Karnaugh et al. 1985; Ennis et al. 1986; Brooks 1987).

Robotic Systems

Autonomous vehicle research has been under way for over 20 years. An entire survey of the field is beyond the scope of this article, but we present the results from a few of the recent projects that are up and running and use knowledge-based problem-solving techniques to aid in real-time monitoring and control. For a complete discussion, see Weisbin (1987).

Autonomous Vehicle Guidance. A vehicle guidance system was created that incorporates a knowledge management module, sensor modules, and control modules. High-level reasoning and planning are provided by the knowledge management module, which observes sensor information generated by sensor control modules, infers the prevailing situation, and generates plans to change the situation toward a desired state. Plans are then implemented by control modules responsible for tasking sensors and applicable actuators to change the vehicle's state. The knowledge management module is able to reason into the future to project the progression of a task and construct expectations of future occurrences. These projections provide input into the planning process. Unexpected real-time occurrences are handled by the lower-level real-time control modules through previously stored contingency plans that attempt to stabilize the situation. This scheme allows the slower knowledge management module time to catch up with events. The knowledge management module and sensor and control modules reside in separate processors and communicate through a data bus. In a mobile ground robot application, the modules were implemented mostly in Pascal (Harmon 1983).

The Hostile Environment Robotic Machine Intelligence Experiment Series IIB (HERMIES-IIB). HERMIES-IIB is the latest in a series of sophisticated mobile robots developed at Oak Ridge National Laboratory. All high-level decisions are done with an expert system shell—C Logical Production System (CLIPS) (a derivative of OPS5)—which runs on one node of an NCUBE parallel processor and is linked to a set of navigation procedures (Burks et al. 1987).

Ping Pong Player. This real-time expert system supports a ping pong-playing robot. The expert system integrates sensor data, robot capabilities, and task constraints and generates an acceptable plan of action. The system also considers changes occurring in the environment during planning and robot motion. Sensor data are generated at a rate of 60 hertz and include three-dimensional position, velocity, and spin vectors plus a time coordinate. These data are used to predict where the ball will go. An expert system then plans the appropriate motion for hitting the ball back. A blackboard-based architecture is used to interrelate initial planning, temporal updating, and exception handling. The system plays against both human and machine opponents and is reported to be a "very good player" (Andersson 1987).

Real-Time Expert System Tools

In this section, a summary of knowledge-based development tools designed specifically for real-time applications is presented.

Activation Framework (AF)

AF is a software framework written in C that supports the implementation of real-time AI programs on multiple interconnected computers that can be geographically distributed. One of the principle features of AF is the use of message priority levels as the basis for distributed scheduling and focus-of-attention mechanisms. AF evolved from the HEARSAY II architecture. The major reasoning mechanism is C and Lisp procedures. Green (1987) reports that they are "starting to apply the system to the the problem of the navigator for an autonomous vehicle and to the smart conveyor belt robot."

Blackboard Objects (BLOBS)

BLOBS is a general-purpose language for building systems that simulate and interpret complex sensor data. BLOBS blends blackboard system concepts with an object-oriented system; it was initially developed for an air traffic control study for the simulation of aircraft and the interpretation of their detection by radar. Objects can be built to define aircraft and radar and to "represent a hypothesis about events and simulations which are perceived to have occurred within the simulation." BLOBS supports the concept of a blackboard partitioned into panels that represent groupings of types of objects with facilities for restricting access to certain information. Objects can communicate through passed messages or a demon that is activated when certain conditions are achieved on the blackboard. BLOBS is implemented in POP11 in the POPLOG environment operating under VMS on a VAX computer. BLOBS is relevant to other complex situations such as chemical plants and nuclear power plants (Zanconato 1988).

The Experimental Expert System Flight Status Monitor (EESFSM)

EESFSM is being developed at the Dryden Flight Research Facility at the NASA Ames Research Center. EESFSM is designed to be a test bed for concepts in rules, inference mechanisms, and knowledge structures to be used in a real-time expert system that will monitor the health and status of the flight control system of state-of-the-art, high-performance, research aircraft. The application system accepts telemetry downlink data from the aircraft and applies various inference mechanisms to deduce conditions of concern or alarm. The application system interfaces with both a flight system engineer on the ground and a research test pilot in the vehicle (Duke and Regenie 1985).

Expert Controller

UME Corporation of Larkspur, Cali-

An important conclusion from this analysis is that many of the common knowledge representations and inferencing techniques are not suitable for real-time applications.

fornia, sells a tool called Expert Controller. It is a shell and a processor box that has two RS-232 ports for direct interface to programmable logic controllers which monitor and control an application. The expert system development tool runs on a PC, and after the application is completed, it is ported to the controller box to be stored in memory. The box contains a custom, complimentary metal-oxide semiconductor (CMOS) chip and is based on a 64-byte word. The company claims the systems can run 5000 rules per second and use as many as 16,000 rules. UME's Expert Controller is being used in an automotive hood-stamping process control application at General Motors (Schwartz 1987).

Forth

It has long been recognized that Forth offers significant advantages over other languages in speed and compactness. Several efforts have used Forth in the development of real-time expert systems. One effort involved rewriting OPS5 in a multitasking version of Forth and demonstrated that "high-speed, intelligent software operating in a restricted environment" can be achieved (Dress 1986). In another application, the Forth-Based Production System (FORPS) combined the "real-time capabilities of Forth" with the "artificial intelligence qualities of a production system" to develop an obstacle avoidance program for an overhead manipulator transport system (Matheus and Martin 1986). A memory manager using a heap data structure was developed in Forth for a real-time expert system implemented on a Macintosh computer to permit "optimal use of memory even when the size and temporal characteristics of data blocks are a priori unknown" (Dress 1985).

Prolog has been implemented as an embedded real-time operating system using a version of Forth as the basis for a frame- and rule-based real-time expert system that serves as the astronaut interface for a series of vestibular investigations to be performed during a space lab mission (Paloski, Odette, and Krever 1986). The Forth-based Prolog combines Forth's ability for "handling classical real-time tasks such as data acquisition, experiment control, serial communication, and graphical data display" with Prolog's "procedure call mechanism, backtracking control, and built-in database."

Texas Instruments has built an expert system to perform industrial control of a water treatment plant. The tool is written in Forth and delivered on a specially developed 80186 board (Schwartz 1987). Park (1986) describes a concurrent system (called EXPERT-5) that consists of two shells running on separate M68000-based coprocessors mapped into the memory structure of a third system. The supervisor system is a blackboard controller running under MS-DOS on a PC. Forth Inc. offers a forward-chaining rule-based tool called Fuzzy Forth that is deployed using the Novix Forth 6 MHz chip. Built for McDonnell Douglas Aircraft, the system clocked 30,000 rules/second on the Novix chip (Schwartz 1987).

Fuzzy Inference Chip

A VLSI implementation of an inference mechanism has been developed by AT&T Bell Laboratories. The chip is suitable for real-time use in intelligent robot systems and decision-making areas of command and control. The inference mechanism implements fuzzy logic and is capable of executing in parallel all the separate rules that make up a fuzzy relation. The inference engine's architecture consists of a rule-set memory, an inference-processing unit, and a controller. Timing simulations have shown that the chip can operate at a rate of 20.8 megahertz. With a format of 124 bits/rule, a single inference process uses 256 clock cycles. Approximately 80,000 fuzzy logic inferences can then be performed each second. The chip uses custom CMOS technology and is in fabrication (Togai and Watanabe 1986).

G2

G2 allows the knowledge engineer to directly build and manage a real-time expert system, including simulation testing of the knowledge base prior to online use. G2 is a product of the Gensym Corporation in Cambridge, Massachusetts, and is now available. G2, which is implemented in Common Lisp, is available on a wide range of general-purpose computer systems as well as dedicated symbolic processing systems. G2 has sophisticated facilities for temporal reasoning, focus of attention, truth maintenance, real-time scheduling, and interprocess communication (Wolfe 1987).

The Hybrid Expert System Controller (Hexscon)

Hexscon combines conventional logic programming with expert system technology for use as a real-time process controller. The use of conventional techniques is in recognition of the fact that there are many real-time operations which are handled adequately and rapidly by conventional programming logic. The expert system portion runs a separate process under the real-time operating system. A common data representation permits communication between the two parts. A production system using if-then rules provides the basic representation framework, and the rules are compiled into a compact form to minimize the impact on working memory. Problems to be handled by the expert system are assigned a priority to assure that the more important problems are worked on first. Time is partitioned into past, present, and future frames with past and future time frames residing on disk files to conserve microcomputer memory. Belief and confidence parameters permit distinguishing between uncertainty and lack of information. Importance parameters indicate the relative importance of facts in a rule and weight the belief and confidence parameters for determining when a rule fires. Multiple lines of reasoning can be pursued and then combined using a combination of evidence technique to produce a single overall value of belief and confidence for different and, possibly, conflicting results. Hexscon was designed to occupy no more than 512K of memory. It is implemented in Pascal for an 8086 processor and can operate both in an interactive simulation mode to support knowledge-base development and a real-time control mode. This effort is especially significant because it is one of few that takes into account size and memory limitations. Its target computer architecture was the 1750A (Wright et al. 1986).

ONSPEC Superintendent

Heuristics Inc. of Sacramento, California, offers a PC expert system development tool called ONSPEC Superintendent. The system is implemented in Pascal and can be used to drive programmable logic controllers. It requires a math coprocessor and runs under the FlexOS real-time operating system (Schwartz 1987).

Personal Consultant Plus (PC Plus)

Texas Instruments is addressing some of the important problems in online process control with the addition of PC Online to their set of PC Plus expert system development tools. PC Online allows PC Plus to interact online with process data. PC Online supports process and knowledge base synchronization modes (for example, event driven, sampling, continuous loop, or wall clock time). Reporting and trend-analysis capabilities are provided by PC Online as is support for multiple interfaces to data-acquisition and analysis programs. The new product enhances those PC Plus features which already provide some of the capabilities a real-time expert system needs. PC Plus supports reasoning about trends reflected in a history of previous sensor data. Thus, potentially erroneous sensor data can be detected by comparing data with the trend (Carlson 1987).

The Process Diagnostic System (PDS)

PDS provides online, real-time diagnosis of machine processes. It is implemented in SRL and written in FranzLISP on a VAX running VMS. PDS addresses two problems in the analysis of sensor-based data: spurious readings and sensor degradation. Retrospective analysis support is provided by PDS by storing successive readings of a sensor or successive values of any other node. Storing successive values permits the kinds of time-domain analysis (for example, rate of change, averages, filtering, and curve smoothing) used both at the front end of diagnostic systems and during the diagnosis itself. Metadiagnosis detects sensor degradation through the use of rules that monitor a sensor's behavior, then adapts rules to reflect the reduction in importance of a malfunctioning sensor. The inference mechanism performs forward propagation of belief from sensor nodes so that all directly or indirectly affected nodes and rules are reevaluated (Osborne et al. 1985b; Fox, Lowenfeld, and Kleinosky 1983).

The Process Intelligent Control (PICON)

PICON was the first commercial real-time expert system tool for developing process control applications. It is implemented in LISP on the TI Explorer and LMI Lambda/Plus machines. PICON was designed in response to the following requirements: (1) high speed, context-sensitive rule activation; (2) efficient recycling of memory elements that are no longer needed and maintenance of sensor histories; (3) interactive acceptance of command sequences from the operator; and (4) communication between multiple expert systems. The knowledge base used by PICON can be organized into a hierarchical framework to support the association of rules with contexts. A focus mechanism then searches only those rules applicable to the current context. Archived sensor histories can be used by rules to detect sensor degradation. Rule syntax allows referral to time intervals. Sensors represented in PICON's knowledge base can be assigned a currency interval to indicate the length of time that a sensor reading remains valid and a sampling interval to indicate the rate at which a sensor is to be sampled. The architecture of PICON supports parallel processing on a Lambda machine by allocating real-time sensor interfacing and low-level inferencing to a Motorola 68010 processor. This part of PICON is implemented in C for speedier execution and is responsible for maintaining current, valid sensor data based on the corresponding currency and sampling intervals. The remainder of PICON then resides in the Lisp processor portion of the Lambda machine. PICON has been used in several real-time applications, including petrochemical and industrial process control, aerospace and communications, electromechanical system fault isolation and diagnosis, robotics, and a prototype satellite electric power system controller (SICON) (Moore et al. 1984; Leinweber and Gidwani 1986; Leinweber 1987). Picon has been extended to work in the VAX/VMS environment and is now available from GigaMOS Systems Inc. in Lowell, Massachusetts.

The Procedural Reasoning System (PRS)

PRS was built to perform reasoning and planning in dynamic and uncertain worlds. The system maintains a process stack (containing all relevant procedures) that can be viewed as the system's current intentions for achieving its goals or reacting to some observed situation. It has been applied to the handling of many of the possible malfunctions of the reaction control system on the space shuttle, including procedures for handling faulty sensors and diagnosing jet failures under diverse conditions. It was completed using multiple communicating instantiations of PRS and a small simulator for providing real-time input to the system. This application showed the system's ability to coordinate various plans of action, modify intentions appropriately, and shift its focus of attention. PRS was also applied to the route planning, navigation, and malfunction-handling tasks of an autonomous robot. The system was implemented on a Symbolics 3600 Lisp Machine (Georgeff 1987).

The Real Time Expert System Club of Users (RESCU)

RESCU is a real-time expert system for production plants that was developed with the support of the Alvey Directorate and a 25-member club of users. It was applied to a quality control activity for an ethoxylates detergent plant. Shaw (1987) reports that RESCU is operational and provides consistent support 24 hours a day, connected to plant sensors.

Violet and Annie

Violet and Annie are PC/MS-DOS-based tools implemented in C and are available from Intelligent Applications, Inc., in Columbia, Maryland.

Violet provides the intelligent interpretation and control of complex instruments for vibration-based mechanical health monitoring. A PC-based expert system shell is used to analyze and interpret the output of a spectrum analyzer. These facilities are applicable to a wide range of vibration-based mechanical health-monitoring applications, including data history management, trend analysis, failure forecasting, and comparison across multiple-load conditions.

Annie couples PC-based expert system shells with transducer or process control-based measurement systems for process monitoring and mechanical health diagnosis. Annie can automatically measure values when required and keep track of histories, time constraints, and trends (New Product Announcements 1987).

XNET

This network-management expert system is a shell developed for use with a data network that extends over a large territory and operates 24 hours a day with complex support and diagnostic equipment and management tools which provide data requiring skilled interpretation. XNET monitors and analyzes network data online. Key test data are sampled in real time and include line error rates, network activity, loop-back test results, and so on. The expert system shell was developed specifically for network management and uses a rule-based representation for knowledge. XNET repetitively receives network test data, diagnoses the data, and records any diagnosis made. This cycle repeats until interrupted by the operator (Mueller and Cynar 1987).

The Yorktown Expert System Language One (YES/L1)

YES/L1 is a general-purpose language for developing real-time, rule-based programming applications. Implemented in OPS5 and MacLisp, the tool reflects earlier experience with YES/MVS (Karnaugh et al. 1985; Ennis et al. 1986), a real-time MVS computer operator aid, and incorporates a number of features aimed at providing real-time support. Real-time facilities include interprocess communication, timed reminders, and event waiting. Other efforts to provide speedier performance include the conversion of YES/L1 source to PL/I compiled code, the use of a RETE pattern-matching algorithm modified to resolve conflicts about which rule to fire, and a focus mechanism plus support for grouping rules into context blocks. YES/L1 is currently undergoing tests on the VM and MVS/XA operating systems (Cruise et al. 1987; Brooks 1987).

Real-Time Expert System Theoretical Issues

The following paragraphs describe ongoing research in key theoretical areas for real-time knowledge-based systems.

Performance

The slow execution speed of rule-based systems has prohibited their use in domains requiring high performance and real-time response. Research by Gupta (1985) indicates that current rule-based interpreters spend almost 90 percent of their time in the match step and only around 10 percent of the time in the conflict-resolution and the act steps.

The RETE algorithm (Forgy 1982) is an efficient procedure for determining the set of rules (that is, the match set) that is activated by a dynamically changing database. It capitalizes on the temporal redundancy of the database by caching the instances of patterns and the matches for "joins" across database modifications. The algorithm makes no commitment or decision regarding which activated rule will actually be executed.

The RETE algorithm works by compiling the antecedent conditions of all rules into a special kind of data flow network. On a database modification, the algorithm maintains the set of rules that are applicable given the current state of the database without regard to the applicability of any individual rule. The modifications filter through the network, updating the state stored within the network. The output of the network consists of a specification of changes to the conflict set.

Gupta (1985) and Gupta, Forgy, and Newell (1987) explore various methods for speeding up the execution of rule-based systems. They examine the role of parallelism for high-speed execution and examine various architectural features in the design of computers for rule-based systems. The analysis was carried out with several real production systems that were built with OPS5 and Soar. Contrary to initial expectations, they showed that the speedup which can be obtained from parallelism is quite limited, only about tenfold. The reasons for the small speedup are (1) the small number of rules relevant to each change to the database, (2) the large variation in the processing requirements of relevant rules, and (3) the small number of changes made to the database between synchronization steps.

Furthermore, they observe that to obtain this limited factor of tenfold speedup, it is necessary to exploit parallelism at a fine granularity. They propose that a suitable architecture to exploit such fine-grain parallelism is a shared-memory multiprocessor with 32 to 64 processors.

It should be noted that none of the systems which were studied would be considered real-time monitoring or control applications. In fact, in a real-time application involving many changes to the database each cycle, parallelism could provide a much greater boost to execution speed than the factor of 10 indicated by Gupta. Furthermore, as Gupta points out, the analysis is dependent on prevailing programming styles and loses some of its validity as programming styles evolve. Also further complicating the analysis, the simulation results do not include the scheduling, synchronization, and memory contention overheads that will be experienced in a shared-memory multiprocessor, and to this extent the results represent an upper bound on the amount of speed up that is attainable for the nonreal-time applications.

Guaranteed Response Times

Currently, ad hoc techniques are used for making a system produce a response within a specified time interval. Usually, the programmer is responsible for making the system run in real time. The programmer builds the system, runs it, discovers it won't work in real time, and then continues to modify the program until it does run in real time. As discussed by O'Reilly and Cromarty (1985), this hand tuning suffers from several major deficiencies.

First, ad hoc methods cannot be carried beyond the current project. Second, performance becomes brittle in the face of changes in both problem specification and the type or quantity of data. Third, the hand-tuning process is extremely time consuming because the programmer is performing an unconstrained search through the space of possible program modifications to identify the subset that meets the required constraints. Finally, a formal basis is lacking and the process results in performance that is only "coincidently real-time" (that is, real-time performance is determined by running the program against a less-than-exhaustive set of test cases). Ideally, one would like to be able to prove that the knowledge-based system can respond within a fixed time interval without requiring a full understanding of the database at run time.

Production Systems. An analysis by O'Reilly and Cromarty (1985) shows that a production system which uses forward chaining is exponential time. The number of rules that will fire explodes exponentially with the depth of the inference tree.

They also show that in backward chaining (whether one uses depth-first or breadth-first search), every increment in the depth of the tree gives an exponential increase in the number of tree nodes and a combinatorial increase in the number of paths to search. The greater the branching factor of a node, the quicker the exponential explosion.

It should be noted that because an algorithm's performance degrades exponentially, it is still possible to place bounds on the response time. However, worst-case analysis probably results in response times that are unacceptable.

Real-Time for RETE. Haley (1987) discusses several issues involved in using rule-based systems for applications that must perform in real time. He focuses on needing proof rather than accepting the real-time performance of a less-than-exhaustive set of test cases. The main point of the paper is a description of aspects of the RETE algorithm that are not well suited to the formal analysis required to assure real-time performance. His analysis shows that one cannot a priori calculate the cost (that is, the number of *joins,* where a join is an instantiation of a variable across patterns) of adding an assertion to the database using the RETE algorithm. Instantiating a join results in unpredictable response times. Haley presents several methods for bounding the cost of matching a rule, including:

1. Join matching limitations: Establish some finite limit on the number of matches for a join.
2. Pattern instantiation restrictions: Establish a limit on the number of instances of a pattern.
3. Relation instance restriction: Establish some finite limit on the number of instances of a relation.
4. Cardinality restrictions: Establish some finite limit on the number of instances of a relation given a set of values for some subset of its arguments.

Concern exists that these restrictions might compromise the power and flexibility of the production system approach.

It is interesting to note that Haley's paper shows that the RETE algorithm in its purest form can result in unpredictable response times, but research by Gupta and others is concentrating on how to speed up the algorithm using special-purpose hardware. Although many applications require high performance but not guaranteed response times, many of the applications for which this special-purpose hardware is intended are truly real time. Thus, research which focuses on speeding up a version of the algorithm (or some derivative of it) that can guarantee response times should be a high priority.

Frame-Representation Languages. *Frame languages* are a common type of knowledge representation and are primarily used to define semantic nets. The *instance* relation is often used to allow properties and default values to be inherited from generic-type frames and to retrieve and process all instances of a given type at run time. By adding a second relation, *is-a*, and using it to organize type frames in an organizational hierarchy, frame languages can support object-oriented programming.

The inheritance network forms a tree with a single root given the following definition of simple inheritance: Each class can have at most one super class; only two relations are allowed, *is-a* and *instance.*

Retrieving a value from a slot consists of following a simple linear list. Worst-case times can easily be calculated for retrieving and storing values in the hierarchical tree (O(d)), where d is the depth of the inheritance tree. Thus, frame languages using only simple inheritance are suitable for keeping tight bounds on response times in real-time problems.

Although original frame languages provide only these two relations (is-a and instance), many of the new languages allow the user to define new relations between frames. Also, objects can have more than one superclass (this is called multiple or mixed inheritance). Although multiple inheritance allows the user to gain further expressiveness, it brings a whole new range of problems that must be solved. The inheritance network effectively becomes an arbitrary directed graph. Retrieving a value from a slot now involves some type of search, such as depth first or breadth first. Our earlier analysis of such strategies concluded that they might not be suitable for real-time applications because of their exponential nature.

An important conclusion from this analysis is that many of the common knowledge representations and inferencing techniques are not suitable for real-time applications. Furthermore, no one seems to be considering the ramifications of using exponential-time algorithms in a real-time application.

Making a Best Guess Given a Deadline

For many real-time monitoring and control applications, a variety of tasks exists in which the amount of time available to make a decision does not always allow careful consideration of all the options. Thus, the problem is to come up with the best possible decision in the time available. Surprisingly enough, little work has been reported in the AI literature in this important area of research.

Decision Analytic Techniques. Horvitz (1987, 1988) presents several areas of research on problem-solving trade-offs in reasoning systems. Areas of research on problem-solving trade-offs include (1) strategic control, (2) structural control, and (3) the explanation of computation. Horvitz describes the application of utility-decision theory to the task of controlling problem-solving trade-offs. Value trade-offs encountered in reasoning systems that Horvitz has examined include (1) immediacy versus the accuracy or precision of a solution, (2) the degree of certainty versus the level of abstraction, (3) solving a subproblem versus solving other subproblems, (4) metareasoning versus object-level reasoning, and (5) inference transparency versus inference optimality.

Progressive Deepening

Winston (1984) describes a method known as progressive deepening to keep computing within time bounds. This method analyzes (searches) each situation to depth 1, then depth 2, then depth 3, and so on, until the amount of time set aside for the analysis is reached. Thus, a response is always ready. The response is determined by the analysis at one level less deep than the analysis in progress when time runs out. Winston shows that this method does not waste much time in extra analysis at shallow levels.

Progressive Reasoning

Wright, et al. (1986) describe a method they call progressive reasoning that lets Hexscon obtain the best possible decision within the time available. The system has four levels of reasoning, with the first implemented in conventional logic, followed by three levels of reasoning in the knowledge-based part. Each successive level uses data that are more time-consuming to retrieve and process than the previous level. The progressive reasoning described by Wright is a subset of progressive deepening.

AIRPLAN

Masui, McDermott, and Sobel (1983) describe a system called AIRPLAN that is being developed to assist air operations officers with the launch and recovery of aircraft on a carrier. The task with which AIRPLAN assists has strong time constraints. Situations can arise in which the amount of time available is insufficient to adequately explore the implications of whatever event has just occurred. AIRPLAN provides four levels of assistance to the air operations officers: (1) Display raw data, (2) identify problems and roughly characterize the possible solutions, (3) refine the characterization of possible problems; and (4) identify future problems.

When AIRPLAN receives a report, it updates its display. It then determines the consequences of the report; if it discovers a problem, it characterizes the options as good or bad. If no backlog of unanalyzed reports exists, AIRPLAN continues with its analysis, and the result is a set of recommendations. When there are no reports to consider, it generates hypothetical future events and possible solutions. When a new report is received, AIRPLAN uses demon rules to "interrupt itself" and record the report and its probable urgency; it then returns to whatever it task was previously performing. The approach used in AIRPLAN is another special case of progressive deepening.

Time-Constrained Inference Network

Sorrells (1985) describes an inference strategy designed to work in time-constrained military domains. The inference strategy considers the most influential knowledge first and less significant data as time permits. This time-constrained inference strategy utilizes a semantic network to represent domain knowledge and a time-merit value to guide the search. The certainty of the expert system response increases in proportion to the amount of processing time available. It has been successfully demonstrated in a prototype system for weapon-to-threat assignment.

Variable Precision Logic

Michalski and Winston (1986) describe a variable precision logic (VPL) that is concerned with the problems of reasoning with incomplete information and resource constraints. It offers mechanisms for handling trade-offs between the precision of inferences and the computational efficiency of deriving them. Two aspects of precision are specificity of conclusions and the certainty of belief in them. The paper primarily addresses certainty and employs censored production rules as the underlying computational mechanism. Censored production rules are simply augmented ordinary production rules with an exception condition and are written in the form "if A then B unless C," where C is the exception condition.

Censored production rules are intended for situations in which the implication A=>B holds frequently, and the assertion C holds rarely. A system using censored production rules is free to ignore the exception condition when resources are tight. Given increased time, the exception conditions are examined, lending credibility to high-speed answers or changing them. The strong point of such a logical system is that it exhibits variable certainty of conclusions, reflecting variable investment of computational resources in conducting reasoning.

Jackson (1987) has implemented VPL with ART in a toy real-time monitoring problem of deciding if a car is being driven safely.

Haddawy (1986) has implemented a VPL system in Common Lisp on a Symbolics 3640. He presents two simple examples. The first deals with bird classification and highlights approximate inference methods. The second deals with driving a car and shows the ability of the system to vary the depth of censor chaining in response to time limits. The paper only investigates the trade-off between inference time and certainty.

Conclusions

From the results of the survey, we see that considerable effort is being put into developing intelligent real-time systems, a much more difficult area than has traditionally been approached using expert systems. In real-time problem solving, many human limitations—their tendencies to overlook relevant information, to respond inconsistently, to respond too slowly, and to panic when the rate of information flow is too great—are most apparent, and the need to overcome these shortcomings is at its greatest.

Limitations of Current Tools for Real Time

In this article, we have reviewed a variety of applications built with a number of tools. Over 100 different expert system-building tools (shells) are commercially available. Additionally, several proprietary shells have been developed in house by different corporations. Of the commercially available shells, only Picon (Moore et al. 1984; Leinweber and Gidwani 1986; Leinweber 1987) and G2 (Wolfe 1987) have been built explicitly for real-time monitoring and control applications.

Real-time domains present complex, dynamic problems because of their dependence on the time factor. A real-time expert system must satisfy demands that do not exist in conventional domains. However, current shells are not generally appropriate for real-time applications for the following reasons: (1) The shells are not fast enough (research from the Defense Advanced Research Project Agency's Pilot's Associate program indicates current tools are two to three orders of magnitude too slow.), (2) the shells have little or no capabilities for temporal reasoning, (3) the shells are difficult to integrate in an efficient manner with conventional software, (4) the shells have little or no facilities for focusing attention on significant events, (5) the shells offer no integration with a real-time clock, (6) the shells have no facilities for handling asynchronous input, (7) the shells have no way of handling software-hardware interrupts, (8) the shells cannot efficiently take input from external stimuli other than a human, (9) methods do not exist for verifying and validating the shells or the knowledge bases they execute, (10) the shells cannot guarantee response times, and (11) the shells run on hardware that was not built for harsh environments. Very few of the applications described in this article, discussed the issue of guaranteed response times.

Trying to apply current shells to real-time domains is like trying to use Prolog for a number-crunching application or Fortran for a symbolic processing application. One can try and stretch a tool for whatever the application calls for. However, the purpose of these tools is to increase one's productivity. Using them for real-time applications to which they are not well suited seems futile. Much research is needed in building special knowledge-based tools for real- time domains. In trying to fill this void, the GENSYM Corporation was created to pursue research and development in real-time applications of expert systems (Spang Robinson Report 1986).

Requirements for a Real-Time Expert System

The real-time expert system implementations surveyed in this article were reviewed to examine the problems that are important from an expert system viewpoint; to study how the problems have been addressed; and to see what can be learned about the features which would be desirable in a general-purpose, real-time expert system. Additionally, some major theoretical issues in real-time expert systems were included in the survey.

Based on consideration of the surveyed research and development, the following classes of features can be expected of a real-time expert system:

Efficient integration of numeric-symbolic computing: Many algorithms have been developed for solving various real-time problems such as data compression, signal processing, and feature extraction. These algorithms need to be efficiently integrated with the symbolic processing module.

Continuous operation: A real-time expert system must be capable of continuous operation, even if a fault with the associated real-time system has been encountered. Real-time systems do not necessarily cease functioning when a fault develops. Because the system must run continuously for long periods of time, close attention must also be paid to garbage collection.

Focus-of-attention mechanism: A capability should exist for assigning a context in which certain rules apply. There should also be a capability for focusing the system's resources when a significant event occurs.

Interrupt-handling facility: An ability to manage asynchronous messages should be possible so that ongoing processing can be interrupted and resumed after a higher-priority message is processed.

Optimal environment utilization: The expert system should make optimal use of its operational environment. Techniques used by applications found in the survey included using compiled rather than interpreted code.

Predictability: The behavior of the expert system should be predictable. That is, for a given time constraint, the ability of the expert system to provide a response should be determinable. This statement infers that garbage collection must be done "on the fly," not at the processor's discretion.

Temporal reasoning facility: The knowledge representation scheme should permit representation of temporal relationships. A capability should exist for maintaining, accessing, and statistically evaluating historical data.

Truth maintenance facility: Empirical data indicate that in some domains, some data decay in quality as a function of time. This decay was found to be true in some environments where sensors are the source of data. As the validity of the data goes to zero, mechanisms are needed to retract all assertions based on the now-invalid data.

Closing Remarks

Robert C. McArthur of the Arthur D. Little Company once told an executive training seminar that "real-time expert systems are real hard to develop" (Marsh 1986). Perhaps this statement explains why many more expert system applications exist in conventional domains than in real-time domains. Nonetheless, knowledge-based problem-solving techniques offer a great potential for adding needed intelligence to real-time domains. The results of this survey indicate that a substantial amount of research is still needed.

Very few of the applications surveyed have progressed beyond the prototype stage to be used everyday in a real-time domain. We concluded that one of the main reasons for this situation is that expert system developers have often tried to apply traditional tools to applications for which they are not well suited. Tools specifically built for real-time monitoring and control applications need to be built. An immediate goal should be the development of high-performance inference engines that can guarantee response times.

Real-time domains offer a new and challenging environment for the application of knowledge-based systems. However, many hard problems need to be solved before we will see the widespread use of real-time knowledge-based systems.

References

Ali, M., and Scharnhorst, D. A. 1985. Sensor-Based Fault Diagnosis in a Flight Expert System. In Proceedings of the Second Conference on Artificial Intelligence Applications, 49-52. Washington D.C.: IEEE Computer Society.

Allard, J. R., and Kaemmerer, W. F. 1987. The Goal/Subgoal Knowledge Representation for Real-Time Process Monitoring. In Proceedings of the Sixth National Conference on Artificial Intelligence, 394-398. Los Altos, Calif.: Morgan Kaufmann.

Anderson, B. M.; Cramer, N. L.; Lineberry, M.; Lystad, G. S.; and Stern, R. C. 1984. Intelligent Automation of Emergency Procedures in Advanced Fighter Aircraft. In Proceedings of the First Conference on Artificial Intelligence Applications, 496-501. Washington D.C.: IEEE Computer Society.

Andersson, R. L. 1987. Real Time Expert System to Control a Robot Ping Pong Player. Ph.D. diss., Dept. of Computer and Information Science, Univ. of Pennsylvania.

Baas, L., and Bourne, J. R. 1984. A Rule-Based Microcomputer System for Electroencephalogram Evaluation. *IEEE Transactions on Biomedical Engineering* BME-31(10).

Behan, J., and Lecot, K. 1987. Overview of Financial Applications of Expert Systems. In Proceedings of WESTEX-87: IEEE Western Conference on Knowledge-Based Engineering and Expert Systems, 223-229. Washington D.C.: IEEE Computer Society.

Broadwell, M. M., and Smith, D. M. 1986. Interfacing Symbolic Processes to a Flight Simulator. In Proceedings of the 1986 Summer Computer Simulation Conference, 751-755. San Diego: SCS.

Brooks, M. 1987. IBM ES to Aid in Mainframe Management. *Applied Artificial Intelligence Reporter* 4(6): 17.

Brown, H. D.; Buckman, J.; Engelmore, R.; Harrison, D.; and Pfefferkorn, C. 1982. Communication Intelligence Task—HANNIBAL Demonstration, Technical Report, ESL, Inc.

Burks, B. L.; de Saussure, G.; Weisbin, C. R.; and Hamel, W. R. 1987. Autonomous Navigation, Exploration, and Recognition Using the Hermies-IIb Robot. *IEEE Expert* 2(4): 18-27.

Campbell, M. 1987. Building EXIMU: Problems and Solutions from the Expert's Point of View. In Proceedings of the Third Annual AI and Advanced Computer Technology Conference, 609-619. Wheaton, Ill.: Tower Conference Management Co.

Carlson, K. 1987. Process-Driven Expert Systems. In Proceedings of the Third Annual Artificial Intelligence and Advanced Computer Technology Conference, 281-288. Wheaton, Ill.: Tower Conference Management Co.

Chen, D. C. 1985. Progress in Knowledge-Based Flight Monitoring. In Proceedings of the Second Conference on Artificial Intelligence Applications, 441-446. Washington D.C.: IEEE Computer Society.

Chester, D.; Lamb, D.; and Dhurjati, P. 1984. Rule-Based Computer Alarm Analysis in Chemical Process Plants. In Proceedings of the Seventh Annual Conference on Computer Technology, 22-29. Washington D.C.: IEEE Computer Society.

Clippinger, J. H. 1983. An Artificial Intelligence System for the Realtime Monitoring and Analysis of Textual Information. In Proceedings of the Trends and Applications Conference, 65-67. Washington D.C.: IEEE Computer Society.

Cruise, A.; Ennis, R.; Finkel, A.; Hellerstein, J.; Klein, D.; Loeb, D.; Masullo, M.; Milliken, K.; Van Woerkom, H.; and Waite, N. 1987. YES/L1: Integrating Rule-Based, Procedural, and Real-Time Programming for Industrial Applications. In Proceedings of the Third Conference on Artificial Intelligence Applications, 134-139. Washington D.C.: IEEE Computer Society.

Cruz, R. E. 1986. Application of Adaptive Learning to Malfunction Recovery, Technical Report, NAS 1.26:166620, NASA-CR-166620, National Aeronautics and Space Administration.

D'Ambrosio, B.; Fehling, M; Forrest, S.; Raulefs, P.; and Wilbur, B. 1987. Real-Time Process Management for Materials Composition in Chemical Manufacturing. *IEEE Expert* 2(2): 80-89.

Dickey, F. J., and Toussaint, A. L. 1984. ECESIS: An Application of Expert Systems to Manned Space Stations. In Proceedings of the First Conference on Artificial Intelligence Applications, 483-489. Washington D.C.: IEEE Computer Society.

Doyle, R. J.; Sellers, S. M.; and Atkinson, D. J. 1987. Predictive Monitoring Based on Causal Simulation. In Proceedings of the Second Annual Research Forum, 44-59. Moffett Field, Calif.: NASA Ames Research Center.

Dress, W. B. 1986. AI (Artificial Intelligence) Goes Forth. In Proceedings of the Goddard Conference on AI Applications, Conf-8605123-1. Washington D.C.: Dept. of Energy.

Dress, W. B. 1985. FORTH Implementation of the Heap Data Structure. *The Journal of Forth Application and Research* 3(2): 135-138.

Duke, E. L., and Regenie, V. A. 1985. Description of an Experimental Expert System Flight Status Monitor, Technical Report, Memorandum 86791, NASA Ames Research Center, Dryden Flight Research Facility.

Ennis, R. L.; Klein, D.; Milliken, K.; Schor, M.; Greismer, J.; Hong, S.; Karnaugh, M.; Kastner, J.; and Van Woerkom, H. 1986. A Continuous Real-Time Expert System for Computer Operations. *IBM Journal of Research and Development* 30(1): 14-28.

Fagan, L. 1980. VM: Representing Time-Dependent Relations in a Medical Setting. Ph.D. diss., Dept. of Computer Science, Stanford Univ.

Fagan, L. M.; Kunz, J. C.; and Feigenbaum, E. A. 1979. Representation of Dynamic Clinical Knowledge: Measurement Interpretation in the Intensive Care Unit. In Proceedings of the Sixth International Joint Conference on Artificial Intelligence, 260-262. Los Altos, Calif.: Morgan Kaufmann.

Forgy, C. L. 1982. RETE: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. *Artificial Intelligence* 19: 17-37.

Fox, M.; Lowenfeld, S.; and Kleinosky, P. 1983. Techniques for Sensor-Based Diagnostics. In Proceedings of the Eighth International Joint Conference on Artificial Intelligence, 158-163. Los Altos, Calif.: Morgan Kaufmann.

Georgeff, M. P. 1987. An Embedded Real-Time Reasoning System. In Proceedings of the Second Annual NASA Research Forum, 286-329. Moffett Field, Calif.: NASA Ames.

Green, P. E. 1987. AF: A Framework for Real-Time Distributed Cooperative Problem Solving. In *Distributed Artificial Intelligence*, 153-175. Los Altos, Calif.: Morgan Kaufmann.

Gupta, A. 1985. Parallelism in Production Systems: The Sources and the Expected Speed Up. In Expert Systems and Their Applications, Fifth International Workshop, 26-57. Avignon, France: Agence de l'Informatique.

Gupta, A.; Forgy, C.; and Newell, A. 1987. High Speed Implementation of Rule-Based Systems, Preliminary Draft, Dept. of Computer Science, Stanford Univ.

Haddawy, P. 1986. Implementation of and Experiments with a Variable Precision Logic Inference System. In Proceedings of the Fifth National Conference on Artificial Intelligence, 238-242. Los Altos, Calif.: Morgan Kaufmann.

Haley, P. V. 1987. Real-Time for RETE. In Proceedings of ROBEXS'87: The Third Annual Workshop on Robotics and Expert Systems. Research Triangle Park, N.C.: Instrument Society of America.

Hamilton, M. 1986. SCARES—A Spacecraft Control Anomaly Resolution Expert System. In Proceedings of the 1986 Expert Systems in Government Conference, 436-443. Washington D.C.: IEEE Computer Society.

Hardt, S. L., and Rosenberg, J. 1986. Developing an Expert Ship Message Interpreter: Theoretical and Practical Conclusions. *Optical Engineering* 25(3): 456-464.

Harmon, S. Y. 1983. Coordination between Control and Knowledge Based Systems for Autonomous Vehicle Guidance. In Proceedings of the Trends and Applications Conference, 8-11. Washington D.C.: IEEE Computer Society.

Healey, K. J. 1986. Artificial Intelligence Research and Applications at the NASA Johnson Space Center. AI Magazine 7(3): 146-152.

Horvitz, E. J. 1988. Reasoning about Beliefs and Actions under Computational Resource Constraints. In *Uncertainty in Artificial Intelligence, Vol. 3*, eds. T. Levitt, J. Lemmer and L. Kanal. Amsterdam: North Holland. Forthcoming. Also, Technical Report, KSL 87-29, Dept. of Computer Science, Stanford Univ.

Horvitz, E. J. 1987. Problem-Solving Design: Reasoning about Computational Value, Trade-Offs, and Resources. In Proceedings of the Second Annual NASA Research Forum, 26-43. Moffett Field, Calif.: NASA Ames.

Jabbour, K.; Vega-Riveros, J.; Landsberger, D.; and Meyer, W. 1986. ALFA: Automated Load Forecasting Assistant. In Proceedings of WESTEX-86: IEEE Western Conference on Knowledge-Based Engineering and Expert Systems, 209-214. Washington D.C.: IEEE Computer Society.

Jackson, P. C. 1987. Elements of Variable Precision Logic Based on ART. In Proceedings of 1987 Society of Automotive Engineers International Congress and Exposition, Detroit Mich.

Kao, S. M.; Laffey, T. J.; Schmidt, J. L.; Read, J. Y.; and Dunham, L. 1987. Real Time Analysis of Telemetry Data. In Proceedings of the Third Annual Expert Systems in Government Conference, 137-144. Washington D.C.: IEEE Computer Society.

Karnaugh, M.; Ennis, R.; Griesmer, J.; Hong, S.; Klein, D.; Milliken, K.; Schor, M.; and Van Woerkom, H. 1985. A Computer Operator's Expert System. In Proceedings of the Seventh International Conference on Computer Communications: The New World of the Information Society, 810-815. Amsterdam: North Holland.

Krutchen, R. J. 1986. Artificial Intelligence and Satellite Autonomy. In Proceedings of the 1986 Expert Systems in Government Conference, 7-15. Washington D.C.: IEEE Computer Society.

Leinweber, D. 1987. Expert Systems in Space. *IEEE Expert* 2(1): 26-36.

Leinweber, D., and Gidwani, K. 1986. Real-Time Expert System Development Techniques and Applications. In Proceedings of WESTEX-86: IEEE Western Conference on Knowledge-Based Engineering and Expert Systems, 69-77. Washington D.C.: IEEE Computer Society.

Lusk, E. L., and Stratton, R. 1983. Automated Reasoning in Man-Machine Control Systems. In Proceedings of the Ninth Annual Advanced Control Conference, 41-47, West Lafayette, Ind.

Malkoff, D. B. 1986. Real-Time Fault Detection and Diagnosis: The Use of Learning Expert Systems to Handle the Timing of Events, Technical Report, NPRDC TR 87-8, Navy Personnel Research & Development Center.

Marsh, A. 1986. *Guide to Defense and Aerospace Expert Systems.* Arlington, Va.: Pasha Publications.

Marsh, A. 1984. NASA to Demonstrate Artificial Intelligence in Flight Operations. *Aviation Week and Space Technology*, Sept. 17, 1984.

Marsh, J., and Greenwood, J. 1986. Real-Time AI: Software Architecture Issues. In Proceedings of the IEEE 1986 National Aerospace and Electronics Conference, 67-77. Washington D.C.: IEEE Computer Society.

Masui, S.; McDermott, J.; and Sobel, A. 1983. Decision-Making in Time Critical Situations. In Proceedings of the Eighth International Joint Conference on Artificial Intelligence, 233-235. Los Altos, Calif.: Morgan Kaufmann.

Matheus, C. J., and Martin, H. L. 1986. Forps: A FORTH-Based Production System and Its Application to a Real-Time Robot Control Problem. In Proceedings of American Society of Mechanical Engineers Pressure Vessel and Piping Conference and Exhibit. Washington D.C.: Department of Energy.

Michalski, R. S., and Winston, P. H. 1986. Variable Precision Logic. *Artificial Intelligence* 29(2): 121-146.

Moore, R. L.; Hawkinson, L. B.; Knickerbocker, C. G.; and Churchman, L. M. 1984. A Real-Time Expert System for Process Control. In Proceedings of the First Conference on Artificial Intelligence Applications, 569-576. Washington D.C.: IEEE Computer Society.

Mueller, D., and Cynar, L. 1987. A Network Management Based Expert System. In Proceedings of the Third Annual Artificial Intelligence and Advanced Computer Technology Conference, 604-608. Wheaton, Ill.: Tower Conference Management Co.

Nelson, W. R. 1984. Response Trees and Expert Systems for Nuclear Reactor Operations, Technical Report, NUREG/CP-3561, EGG-2293, U.S. Nuclear Regulatory Commission.

Nelson, W. R. 1982. REACTOR: An Expert System for Diagnosis and Treatment of Nuclear Reactor Accidents. In Proceedings of the Second National Conference on Artificial Intelligence, 296-301. Los Altos, Calif.: Morgan Kaufmann.

New Product Announcements. 1987. *AI Magazine* 8(2): 130-131.

News Column. 1986. *The Spang Robinson Report* 2(10): 12.

Norman, P., and Naveed, S. 1985. An Expert System Supervisor for a Rotary Cement Kiln. In IEE Colloquium on Real-Time Expert Systems in Process Control, No. 107, 71-79. London: IEE.

O'Reilly, C. A., and Cromarty, A. S. 1985. "Fast" is not "Real-Time" in Designing Effective Real-Time AI Systems. *Applications of Artificial Intelligence II* 548, 249-257. Bellingham, Wash.: International Society of Optical Engineering.

Osborne, R. L.; Gonzalez, A. J.; Emery, F. T.; Hurwitz, M. J.; McCloskey, T. H.; Edmonds, J. S. 1985a. Increased Power Plant Availability and Reliability through On-Line Diagnostics Based on Artificial Intelligence. In Proceedings of the American Power Conference, 539-543. Chicago, Ill.: Illinois Institute of Technology.

Osborne, R. L.; Gonzalez, A. J.; Bellows, J. C.; and Chess, J. D. 1985b. On-Line Diagnosis of Instrumentation through Artificial Intelligence. In Proceedings of the Instrument Society of America Power Symposium, 89-94. Research Triangle Park, N.C.: Instrument Society of America.

Paloski, W. H.; Odette, L. L.; and Krever, A. J. 1986. Use of a Forth-Based Prolog for Real-Time Expert Systems. *Journal of Forth Applications and Research* 4(2).

Park, J. 1986. Toward the Development of a Real-Time Expert System. *Journal of Forth Applications and Research* 4(2): 133-154.

Pisano, A. D., and Jones, H. L. 1984. An Expert Systems Approach to Adaptive Tactical Navigation. In Proceedings of the First Conference on Artificial Intelligence Applications. Washington D.C.: IEEE Computer Society.

Rader, C. D.; Crowe, V. M.; and Marcott, B. G. 1987. Caps: A Pattern Recognition Expert System Prototype for Respiratory and Anesthesia Monitoring. In Proceedings of WESTEX-87: IEEE Western Conference on Knowledge-Based Engineering and Expert Systems, 162-168. Washington D.C.: IEEE Computer Society.

Raulefs, P.; D'Ambrosio, B.; Fehling, M.; Forrest, S.; and Wilber, M. 1987. Real-Time Process Management for Materials Composition. In Proceedings of the Third Conference on Artificial Intelligence Applications, 120-125. Washigton D.C.: IEEE Computer Society.

Redington, D. 1986. A Forth Oriented Real-Time Expert System for Sleep Staging: A Fortes Polysomnographer. *Journal of Forth Applications and Research* 4(1): 47-56.

Rodewald, L. E. 1984. BABY: An Expert System for Patient Monitoring in a Newborn Intensive Care Unit. Master's Thesis, Computer Science Dept., Univ. of Illinois at Champaign-Urbana.

Rook, F., and Odublyl, J. 1986. An Expert System for Satellite Orbit Control (ESSOC). For copies of this publication, contact Rook and Odublyl at Contel SPACECOM, 1300 Quince Orchard Boulevard., Gaithersburg, MD 20878.

Sachs, P. A.; Paterson, A. M.; and Turner, M. H. M. 1986. Escort—An Expert System for Complex Operations in Real Time. *Expert Systems* 3(1): 22-29.

Scarl, E.; Jamieson, J.; and Delaune, C. 1984. Knowledge-Based Fault Monitoring and Diagnosis in Space Shuttle Propellant Loading. In Proceedings of the National Aerospace and Electronics Conference. Washigton D.C.: IEEE Computer Society.

Schwartz, T. J. 1987. Real Time Processing on the PC. *Spang Robinson Report* 3(7): 9-17.

Shaw, R. 1987. RESCU—On-Line Real-Time Artificial Intelligence. *Computer-Aided Engineering Journal* 7(3): 29-30.

Skapura, D. M., and Zoch, D. R. 1986. A Real-Time Production System for Telemetry Analysis. In Proceedings of the 1986 Expert Systems in Government Conference, 203-209. Washington D.C.: IEEE Computer Society.

Sorrells, M. E. 1985. A Time-Constrained Inference Strategy for Real-Time Expert Systems. In Proceedings of the IEEE 1985 National Aerospace and Electronics Conference, 1336-1341. Washigton D.C.: IEEE Computer Society.

Szuprowicz, A. 1987. Expert Systems Wrong-Headed in Intensely Competitive Securities Trading Area. *Intelligent Systems Analyst* 7: 14-16.

Togai, M., and Watanabe, H. 1985. A VLSI Implementation of Fuzzy Inference Engine: Toward an Expert System on a Chip. In Proceedings of the Second Conference on Artificial Intelligence Applications, 192-197. Washington D.C.: IEEE Computer Society. Also in 1986. *Information Sciences* 38: 147-163.

Touchton, R. 1986. Emergency Classification: A Real-Time Expert System Application. In Proceedings of Southcon 1986, 2321-2323. Los Angeles, Calif.: Electronics Conventions Management.

Turner, M. 1986. Real Time Experts. *Systems International* 14(1): 55-57.

Vidal, J. J. 1985. AI (Artificial Intelligence) Based Real-Time Support for High Performance Aircraft Operations, Technical Report, NAS 1.26:176906, NASA-CR-176906, National Aeronautics & Space Administration.

Weisbin, C. R. 1987. Real-Time Control: A Significant Test of AI Technologies. *IEEE Expert* 2(4): 16-17.

Winston, P. H. 1984. *Artificial Intelligence,* 2d ed., 129-131. Reading, Mass.: Addison-Wesley.

Wolfe, A. 1987. An Easier Way to Build Real-Time Expert Systems. *ELECTRONICS* S8(S1). New York: McGraw Hill.

Wright, M.; Green, M.; Fiegl, G.; and Cross, P. 1986. An Expert System for Real-Time Control. *IEEE Software* (March): 16-24.

Zanconato, R. 1988. BLOBS: An Object Oriented Blackboard Framework for Reasoning in Time. In *Blackboard Systems,* eds. R. Engelmore and A. Morgan. Wokingham, England: Addison-Wesley. Forthcoming.