An Architecture for Expert System Based Feedback Control

Årzén, Karl-Erik

1988

Document Version:
Publisher's PDF, also known as Version of record

CODEN: LUTFD2/(TFRT-7399)/1-07/(1988)

An Architecture for Expert System Based Feedback Control

Karl-Erik Årzén

Department of Automatic Control
Lund Institute of Technology
September 1988

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name
Report

Date of issue
September 1988

Document Number
CODEN: LUTFD2/(TFRT-7399)/1-07/(1988)

Author(s)
Karl-Erik Årzén

Supervisor

Sponsoring organisation
STU

Title and subtitle
An architecture for expert system based feedback control.

Abstract

It is a recognized problem that many industrial control loops are badly tuned or run in manual mode. Two expert system approaches have been suggested for this problem. Fuzzy, rule-based control replace control algorithms by linguistic rules which model the operators manual control strategy. Knowledge-based control extends the range of conventional controllers by encoding general control knowledge and heuristics concerning tuning and adaptation in a supervisory expert system. An architecture for knowledge-based control is described where two concurrent processes are used for the knowledge-based system and the numerical algorithms. A modular, blackboard-based approach is used. This allows the decomposition of the problem into subtasks which are implemented as separate knowledge sources that can be rule-based with different inference strategies or procedural. The framework can be compared with a real-time operating system and has similar real-time primitives. The system has been implemented on a VAX 11/780 and used with good experiences.

Key words
Real-time expert systems, Feedback control

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
English

Number of pages
7

Recipient's notes

Security classification

Paper to be presented at the IFAC Workshop on
AI in real-time control, 21-23 September, Wales.

An architecture for expert system based feedback control

Karl-Erik Årzén
Department of Automatic Control
Lund Institute of Technology
Box 118
S-221 00 Lund, Sweden

Abstract. It is a recognized problem that many industrial control loops are badly tuned or run in manual mode. Two expert system approaches have been suggested for this problem. Fuzzy, rule-based control replace control algorithms by linguistic rules which model the operators manual control strategy. Knowledge-based control extends the range of conventional controllers by encoding general control knowledge and heuristics concerning tuning and adaptation in a supervisory expert system. An architecture for knowledge-based control is described where two concurrent processes are used for the knowledge-based system and the numerical algorithms. A modular, blackboard-based approach is used. This allows the decomposition of the problem into subtasks which are implemented as separate knowledge sources that can be rule-based with different inference strategies or procedural. The framework can be compared with a real-time operating system and has similar real-time primitives. The system has been implemented on a VAX 11/780 and used with good experiences.

Keywords: Real-time expert systems, Feedback control

1. Introduction

There is currently a significant interest in expert system techniques in the process control community. Applications of many different types have been proposed, implemented and a few also fielded. This paper considers the use of expert system, or knowledge-based system, techniques in the closed control loop.

It is a recognized problem that many industrial control loops are badly tuned or run in manual mode. This decreases the quality of the end product and thus increases cost. The manual control task also adds to the already high cognitive burden that process operators are exposed to in modern control systems.

The reasons for the poor control are many. One could be that the control loop is badly tuned from the beginning. Another could be that the operating conditions have changed since the initialization of the controller. This could, e.g. be due to operation at different operating points or time-varying dynamics.

The conventional solution to the problem of poorly tuned control loops is to use adaptive controllers. Adaptive controllers, e.g., (Åström and Wittenmark, 1989), are currently beginning to be used in industrial practice. There are, however, problems. Even though an explicit self-tuning regulator periodically updates the coefficients of a process model there still are many parameters that must be set explicitly. Examples are model orders and time scales. Such information can be difficult to provide and process operators typically lack the intuitive understanding that they have with conventional PID controllers.

Two expert system approaches have been suggested for the described problem. Both involve using the expert system as a part of the feedback loop. In the well-known fuzzy or rule-based approach, e.g., (Tong, 1984), the attempt is to model the manual control strategy of the process operator. It is expressed as qualitative, linguistic rules for how to choose the control signal in different situations. The rules replace conventional control algorithms. The intended applications are control of complex processes such as, e.g., cement kilns, for which either appropriate models do not exist or are inadequate.

The second approach, from now on referred to as knowledge-based control (Åström and Anton, 1984; Åström *et al*, 1986; Årzén, 1987), instead uses expert system techniques to extend the range of conventional control algorithms by encoding general control knowledge and heuristics regarding tuning and adaptation in a supervisory expert system. The knowledge-based control approach is closer in spirit to conventional adaptive control than fuzzy control is. The approach is also motivated by shortcomings of adaptive controllers.

The recursive identification algorithms and the control algorithms of adaptive controllers can be seen

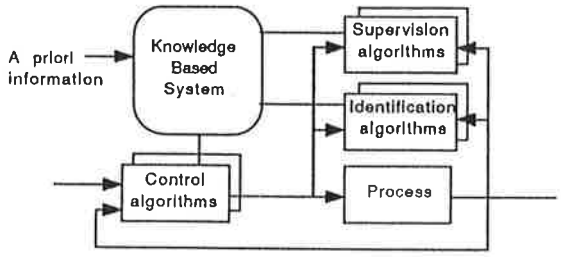

Figure 1. A knowledge-based controller

as the final algorithmic representation of an large amount of underlying theoretical as well as practical control knowledge. This representation is however not enough. It has to be combined with heuristic logic that assures the controller performance under non-standard conditions. These conditions include switching between different operating modes, insufficient process excitation, control signal saturation etc. The concept *safety jacket* or *safety net* (Isermann and Lachmann, 1985; Warvick, 1988) has been established for this logic. The safety net part of a controller is often much larger than the actual algorithms and is designed mainly from intuition, experience, and simulation. Safety nets tend to be very complex. Experience has shown that design and testing is quite time consuming.

The approach in knowledge-based control is to use an expert system to represent the heuristic safety net. The controller consists of the combination of the expert system and a set of control algorithms, identification algorithms and supervision algorithms, as shown in Fig. 1.

The topic of this paper is the organization and architecture of a knowledge-based controller. Knowledge-based control is a real-time expert system application and, as such, contains several difficult problems such as non-monotoning reasoning, representation of time and temporal reasoning, reasoning under time constraints, responsiveness to asynchronous events etc. For an overview of these issues see Laffey *et al* (1988) or Chantler (1988). Some of these issues are still unsolved and will be probably never be completely solved. In several cases, however, practical approaches exist that to some degree solve the problems.

There exist a widely spread mis-understanding that adding intelligent behaviour to a controller is simply a matter of generating a few rules and implementing them in an off-the-shelf expert system shell. This is far from the case. There is a strong interplay between the architecture of the expert system and the type of knowledge that can be naturally expressed in it. The majority of the expert system software is still intended for stand alone, off-line applications. Real-time capacities are only available in few cases such as, e.g., G2 (Gensym, 1987), PICON (Moore *et al*, 1985), and Muse (CCL, 1987).

Section 2 describes the overall organization of an expert system framework that has been developed and

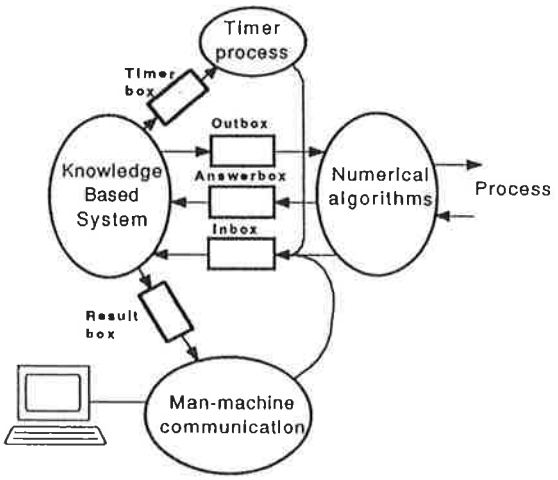

Figure 2. Overall implementation structure.

implemented on a VAX 11/780. The frame-work was developed for knowledge-based control applications but is not restricted to it. The system has been used for design of intelligent tuning controllers (Årzén, 1987). In Section 3, the architecture of the expert system part of the controller is described together. Implementational issues are described in Section 4. Finally, Section 5 contains a discussion about the system and a comparison with other systems.

2. Overall architecture

The knowledge-based controller consists of two major parts: the numerical algorithms and the knowledge-based system. To assure that the execution of the numerical algorithms are not delayed by the knowledge-based system the parts are implemented as two communicating concurrent VMS processes where the numerical algorithms have the highest priority.

The man-machine interface is implemented as a separate process. From this process, the user can interact directly with the knowledge-based system and indirectly with the algorithms. The timer process is used to implement certain real-time interrupts described in the next section. The overall structure is shown in Fig 2.

The knowledge-based system and the man-machine interface are both written in Lisp. The Lisp used is the Unix dialect Franz Lisp, (Foderaro *et al*, 1983). The software package EUNICE, (Kashtan, 1982), is used to create a Unix environment under VMS. The reason for using Lisp is mainly its powerful symbolic processing capabilities. A drawback with Lisp in a real-time system is the garbage collection. The problem can be avoided in two ways. By using a Lisp system with incremental garbage collection the garbage collection activity is spread uniformly in time and performed in the background. The second approach is to write Lisp code that does not generate any garbage. This is what is done in G2.

The numerical algorithms

The numerical algorithm process is written in Pascal. It consists of a library of different algorithms such as PID algorithms, pole-placement algorithms, discrete filters, relays, recursive least-squares algorithms, level crossing detectors, etc. The algorithms are uniformly coded and have well-defined interfaces. It is relatively straight forward to add new types of algorithms to the system. The process is connected to A/D and D/A converters.

The algorithms can principally be divided into three groups: control algorithms, identification algorithms and monitoring algorithms. The control algorithms all compute a control signal based on command and measurement signals. Only one control algorithm can be running at a time. The identification and monitoring algorithms all in some sense extract information from the numerical signal flow. This information is sent to the knowledge-based system. The algorithms in these two groups can be viewed as filters or feature extractors that send information to the knowledge-based system only when something significant has happened. During steady-state operation, the knowledge-based system is not involved and the system resembles a conventional controller. The separation between the numerical algorithms and the knowledge-based system is favourable from the point of information flow. If a knowledge-based system was interfaced directly to a physical process or to an existing control system, numerical information would have to be sent forth and back again at a high rate. The knowledge-based system also had to itself extract all useful symbolic information from the signals. This is a task that often is expressed in the form of numerical algorithms. Using expert system techniques for such tasks is often inefficient.

Inter-process communication

The processes communicate by sending messages through mailboxes shown as rectangles in Fig. 2. The messages that are sent to the algorithms via **Outbox** are configuration commands, parameter changes, and information requests from the knowledge-based system. The messages that go to the knowledge-based system contains results obtained by an algorithm, alarms that have been detected, answers to information requests, user commands, and timer interrupts. Messages to the knowledge-based system are normally sent to **Inbox** which is a standard first-in, first-out VMS mailbox. Messages with priorities indicating their importance are allowed. This is made possible by having an internal mailbox inside the knowledge-based system process into which the messages in **Inbox** are inserted according to their priority. Important messages such as alarms and timer interrupts have a high priority and are thus taken care of as fast as possible. **Answerbox** is used for responses to information requests that has been made by the knowledge-based system. **Resultbox** is used by the knowledge-based system to return results to the man-machine interface.

The use of Lisp has interesting consequences for the communication between the knowledge-based system and the man-machine interface. Mailboxes can be seen as text files where a text line corresponds to a message. In Lisp there is no syntactical difference between data objects and program code, i.e., Lisp functions. Lists are used to represent both. This makes it possible to implement remote evaluation of Lisp functions. An arbitrary Lisp function can be sent to the knowledge-based system from the man-machine interface where it is evaluated and the result is returned in **Resultbox**. erronous

3. Knowledge-based system architecture

A standard off-the-shelf expert system framework, OPS 4, was used to implement the knowledge-based part of the system in a first prototype (Årzén, 1986a, 1986b). OPS 4 (Forgy, 1979) is a simple rule-based, forward-chaining expert system framework. The reason for this choice was the data-driven nature of knowledge-based control. Data in the form of significant events detected by the algorithms are sent to the knowledge-based system which should react and generate some response. The framework OPS 4 uses the incremental pattern-matching algorithm RETE (Forgy, 1982) and is therefore also reasonably fast. Another reason for the choice was simply that the system was available to us and that we wanted to test the basic ideas rapidly.

Experiences of a prototype

The first prototype was used to implement a relay-based PID auto-tuner (Åström and Hägglund, 1984). Experiments with the prototype gave many results concerning both the feasibility of the approach and the demands on a expert system framework for knowledge-based control. We were reassured in that the approach is feasible. The response times for the knowledge-based system were acceptable. The sampling rate for the numerical algorithm process was 1 second. It took approximatively 2–3 sampling periods from that a message was sent to the until a responding message was returned. A second positive result was a clean implementation of a relay auto-tuner that clearly benefited from the separation of logic and algorithms. The time and effort to make extensions to the controller were significantly smaller than for comparable implementations in conventional languages.

The negative experiences all concerned OPS4. It became clear that a simple forward-chaining rule system is not sufficient. An expert system framework must allow for a modular decomposition of both the rulebase and the database. For the database, this could be achieved by a frame system. The rulebase must be decomposable into rule groups for the different subtasks. Further it was found that one knowledge representation technique is not enough. Some subtasks contains large sequential elements. These are more naturally represented procedurally than

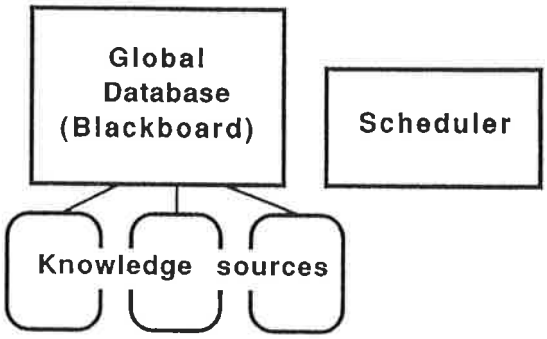

Figure 3. Knowledge based control structure

with rules. It was also clear that backward chaining inferencing would be useful for some problems. The most important drawback with OPS 4 was, however, that it is not designed for real-time operation. It has, e.g., no possibilities to have time-outs associated with database elements, no means for halting the rule execution for a certain time, and no possibilities to check rules at given time intervals.

A blackboard system

Based on the experiences of the prototype, a real-time expert system framework has been developed. The reasoning model chosen as the basis for framework is the blackboard model, (Nii, 1986). A global database, the blackboard, is available to different, co-operating knowledge sources. The database allows for frame structures for storing associated information. The knowledge sources can be thought of as different actors, each of which solves some subtask of the problem. The knowledge sources also have their own local databases. Knowledge sources can be rule-based with either forward or backward chaining and procedural. The structure of the framework is shown in Fig. 3. A knowledge source implements the domain knowledge for a certain task. It is often associated with one or more numerical algorithms. It could for example contain the heuristic logic surrounding an algorithm.

The knowledge sources have primitives for adding, modifying, and deleting frames both globally and locally. They also have primitives to halt their execution for a certain time or until a certain database element is added to the blackboard. It is possible to have forward chaining rules that are tested with specific time intervals and to associate validity intervals with database elements.

The operation of the knowledge-based controller involves the activation of different knowledge sources both in sequence and in parallel. A typical case when knowledge sources are active in parallel is during the steady state control of the process. One knowledge source takes care of the actual control algorithm while other knowledge sources implement different monitoring aspects.

A separate rule-based module schedules the selection of knowledge sources at two different levels. The first level involves the sequential activation of different knowledge sources. The second level involves the scheduling between different knowledge sources that are active simultaneously. This resembles the scheduling in an ordinary real-time, multi-tasking operating system where the knowledge sources are the equivalents of concurrent processes. A knowledge source runs until it explicitly returns control to the scheduler, e.g., if it has to wait for some information or if it is finished. A simple extension which allows interrupts among the knowledge sources is described in Årzén (1987). With this extension, priorities can be associated with knowledge sources. The scheduler contains frames with information of the state of the different knowledge sources and frames which contains information about the conditions on which a knowledge source is waiting.

The primitives that involves waiting a certain time are implemented with the help of the timer process. A primitive that causes a knowledge source to wait a certain time gives rise to a message to the timer process. The message contains the desired wakeup time and a unique identifier for the waittime request. A high-priority message is returned to the scheduler when the waiting time has elapsed. This message causes the state of the waiting knowledge source to be changed to ready.

Knowledge source combination

The operation of the knowledge-based controller typically consists of a sequence, with parallel parts, of knowledge source activations. Three different methods for combining knowledge sources into sequences have been implemented.

The most straightforward way is to use primitives that let knowledge sources activate and deactivate each other. A knowledge source has the possibility to wait until another knowledge source is finished. Procedural knowledge sources also have the possibility to call other procedural knowledge sources, and await and use their returned result.

Another alternative is to have a number of pre-stored sequences. One example of a sequence could be the initial tuning sequence. Other sequences could be used to return to steady-state control when different alarm conditions have been detected. Combination of knowledge sources into sequences is basically a procedural operation. It is therefore natural to express it with procedural knowledge sources. In order for this to be possible, wait primitives that allows waiting for conjunctions and disjunctions of multiple events have been implemented.

The last and most complex method is to dynamically generate sequences. This is accomplished by associating goal states, i.e., post-conditions, and initial states, i.e., pre-conditions, with each knowledge source. Each knowledge source can be viewed as an operator that transforms the state of the system from its initial state to its goal state. A sequence is recursively generated by comparing the desired goal and the current state with the pre- and post-conditions of the operators. This formulation turns the problem into a planning problem. The scheduler generates a

plan which then is executed.

The possibility for different knowledge representation techniques allows the user to choose the technique most natural for each sub-problem. The various methods of combining knowledge sources give a rich and flexible structure. For instance, it is possible to have one knowledge source that contains monitoring rules which are checked periodically. If something erroneous is detected the rules can invoke other knowledge sources that focuses on the problem. These knowledge sources could, e.g. be backward chainers that tries to verify some hypothesis concerning the error or procedural knowledge sources that performs some procedural tests. Meta-knowledge sources with knowledge about the applicability of other knowledge sources are also easy to implement.

A knowledge source in a knowledge-based control application could be, e.g., contain knowledge about design of different controllers. Another knowledge source could contain knowledge about modelling and model validation. Other examples could contain knowledge of different monitoring aspects of the controller. The possibility to refer to past signal values is important in a real-time environment. This is possible through statistics knowledge sources that computes signal statistics over different time horizons. These knowledge sources are associated with numerical algorithms that collect the signal values.

The described framework has been used for the design of elaborate extensions of relay auto-tuning. This is described in Årzén (1987).

4. Implementation

The implementation of the expert system framework is built on the object-oriented system Flavors (Cannon, 1982) and the forward-chaining production system YAPS (Allen, 1983). The YAPS system is a pattern-matching system in the same spirit as the OPS family with a similar optimized, incremental matching algorithm. The important difference is that YAPS is written in Flavors and allows Flavor instances in its database. These Flavor instances can be instances of other YAPS systems.

The YAPS system originally only allows arbitrarily nested list structures of containing numbers, atoms, and Flavor instances as database elements. The system has been modified to allow frame structures. The system has also been extended to allow for automatic explanations of how database elements have been added to the system.

The Scheduler is implemented as a flavor which inherits a YAPS flavor. The scheduling strategy is represented with rules. The different types of knowledge sources are implemented as different flavors. Each individual knowledge source is an instance of the corresponding flavor. Each knowledge source is represented as a frame in the scheduler database. The frame contains slots for the type of knowledge source,

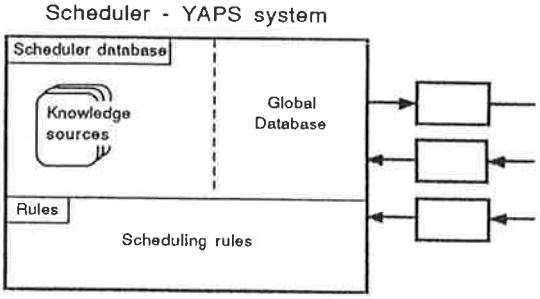

Figure 4. Implementation structure

e.g., forward or procedural, for the state of the knowledge source, and for the actual flavor instance that implements the knowledge source.

The implementation structure is illustrated in Fig. 4. The actual interface between the knowledge sources and the scheduler consists of a relatively small set of messages for which the knowledge source flavors should supply methods. This makes it easy to add new types of knowledge sources to the system.

A slightly simplified example of a rule in the scheduler is given below.

```
(p schedule1
  "If a knowledge source is ready
   and no other knowledge source is
   running then run this knowledge source"
   (frame knowledge-source
      status active
      state ready
      instance -x)
(- (frame knowledge-source
      state running))
-->
   (modify 1 state running)
   (<- -x 'run))
```

The forward chaining knowledge sources are implemented as instances of a flavor that inherits a YAPS flavor. This gives a structure where several YAPS systems reside as database elements inside the Scheduler YAPS system.

The backward chaining knowledge sources are based on an small expert system example in Winston and Horn, (1981), which has been extended and embedded in Flavors. Currently they can only ask questions to the operator when an answer cannot be automatically deduced. A possible extension would be to, in that case, allow a backward chaining knowledge source to invoke a forward or procedural knowledge source that returns the needed answer.

The procedural knowledge sources consists of Lisp functions. In order to allow interrupts of these functions at arbitrary places the entire state of the Lisp computation must be saved. This is not possible with the Franz Lisp of the basic system. Instead Lisp has been used to write a register machine based interpreter for a procedural Lisp-like language that allows the computation to be suspended.

5. Summary

An general expert system framework for real-time applications has been presented. It has been developed for knowledge-based control applications but is not restricted to it.

The framework has real-time facilities. It is modularized into knowledge sources that can be compared with concurrent processes. This is similar to the Muse system. In the current version, however, the knowledge sources cannot interrupt each other. With a simple extension this is possible. The knowledge sources have primitives to wait a certain time or for a certain database element. These primitives are used to implement periodic rule testing in a way similar to G2 and Picon.

Validity intervals can be used to indicate how long database elements remain valid. In contrast with G2 and Picon the validity intervals are not propagated to inferred facts. History values of important signal values are maintained. It is not possible to store history values of arbitrary frame attributes.

The system allows for both rule-based and procedural representation which is very important. The flexible means of combining knowledge sources gives a rich structure.

There are many similarities between real-time operating systems and real-time knowledge-based systems. Real-time operating systems for process control have evolved over a long period of time. This paper indicates a new system architecture where real-time operating systems, databases, object-oriented programming, and knowledge-based systems are combined.

The execution speed of the system is of the order of one forward chaining rule per second. The system is currently being ported to a Symbolics – IBM PC environment where the knowledge-based system resides on the Symbolics and the numerical algorithms reside on the IBM PC. Preliminary results indicate a factor of 10 in increased speed. Other possible candidates for migration are systems where powerful symbolic processing capacity is combined with conventional computing. One example of this is the μ-Explorer.

Acknowledgements

The author would like to thank Professor Karl Johan Åström and Sven Erik Mattsson for many useful discussions. This work has been supported by the National Swedish Board for Technical Development (STU) under contract 85-3084.

References

ALLEN, E.M. (1983): "YAPS: Yet another production system," TR-1146, Department of Computer Science, University of Maryland.

ÅRZÉN, K-E. (1986a): "Expert systems for process control," in D. Sriram and R. Adey (Eds.): *Proc. of First International Conference on Applications of Artificial Intelligence in Engineering Practice*, Springer Verlag, Berlin, pp. 1127–1138.

ÅRZÉN, K-E. (1986b): "Use of expert systems in closed loop feedback control," *Proc. of American Control Conference*, Seattle, WA.

ÅRZÉN, K-E. (1987): "Realization of expert system based feedback control," Ph.D. thesis CODEN: LUTFD2/ TFRT-1029, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

ÅSTRÖM, K.J. and J.J. ANTON (1984): "Expert control," *Proc. 9'th IFAC World Congress*, Budapest, Hungary.

ÅSTRÖM, K.J. and T. HÄGGLUND (1984): "Automatic tuning of simple regulators," *Proc. IFAC 9'th World Congress*, Budapest, Hungary.

ÅSTRÖM K.J. and B. WITTENMARK (1989): *Adaptive Control*, To appear, Addison-Wesley, Reading, MA.

ÅSTRÖM, K.J., J.J. ANTON and K.-E. ÅRZÉN (1986): "Expert control," *Automatica*, **22**, 3, 277–286.

CANNON, H.I. (1982): "Flavors: A non-hierarchical approach to object-oriented programming,," unpublished paper.

CCL (1987): "Muse product description," Cambridge Consultants Limited, 4 pages.

CHANTLER, M.J. (1988): "Real-time aspects of expert systems in process control," *Colloquium on Expert Systems in Process Control*, IEE, Savoy Place, London UK.

FODERARO, J.K., K.L. SKLOWER and K. LAYER (1983): "The Franz Lisp Manual," UC Berkeley, California.

FORGY, C.L. (1979): "OPS4 User's manual," Technical report CMU-CS-79-132, Department of Computer Science, Carnegie-Mellon University.

FORGY, C.L. (1982): "Rete: A fast algorithm for the many pattern/many object pattern match problem," *Artificial Intelligence*, **19**, 1, 17–37.

GENSYM (1987): *G2 User's manual*, Gensym Corp., Cambridge, MA.

ISERMANN, R. and K.H. LACHMANN (1985): "Parameter-adaptive control with configuration aids and supervision functions," *Automatica*, **21**, 6, 625–638.

KASHTAN, D.L. (1982): "EUNICE: A system for porting UNIX programs to VAX/VMS," Artificial Intelligence Center, SRI International, Menlo Park, California.

LAFFEY, T.J., P.A. COX, J.L. SCHMIDT, S.M. KAO, and J. Y READ (1988): "Real-time knowledge-based systems," *AI Magazine*, **9**, 1, 27–45.

MOORE, R.L., L.B. HAWKINSON, M.E. LEVIN and C.G. KNICKERBOCKER (1985): "Expert control," *Proc. American Control Conf.*, Boston, MA, pp. 885–887.

NII, H.P. (1986): "Blackboard systems: The blackboard model of problem solving and the evolution of blackboard architectures," *AI Magazine*, **7**, 2, 38–53.

TONG, R.M. (1984): "A retrospective view of fuzzy control systems," *Fuzzy Sets and Systems*, **14**, 199–210.

WARVICK, K. (1988): *Implementation of self-tuning controllers*, Peter Peregrinus, London.

WINSTON, P.H. and B.K.P. HORN (1981): *Lisp*, Addison-Wesley, Reading, MA.