

Fuzzy Logic Enabled Software Agents for Supervisory Control

Janos L Grantner
Department of Electrical
and Computer Engineering
Western Michigan University
Kalamazoo MI 49008-5329, USA
Email: grantner@wmich.edu

George A Fodor
ABB Automation Technology Products
AB, S-721 67 Vasteras, Sweden
Email: george.a.fodor@se.abb.com

Ramakrishna Gottipati
Department of Electrical
and Computer Engineering
Western Michigan University
Kalamazoo MI 49008-5329, USA
Email: r0gottip@wmich.edu

Abstract - Programs for contemporary industrial control systems are designed using Object Oriented methods and software agents. It is required that the system should reach its objectives even when unexpected events occur in an uncertain environment. A trend is that agents are becoming more autonomous, more complex and are having more responsibilities. There is a need for a high-level information representation such that a predictable behavior can be achieved in a uniform way for all agents. A fuzzy automaton-based approach offers clear benefits for developing agents of reconfigurable architecture. This paper reports on a research project that is currently underway to develop a Generic Encapsulated Fuzzy Automaton Software Agent for Object Oriented Control Systems. A laboratory has been set up to develop and evaluate the performance of new methods and architectures.

I. INTRODUCTION

A component-based program development is characterized by a number of software entities equipped with connectors. An application consists of a set of components and a method to connect them such that certain expected system behavior is achieved. There are many different possible ways to connect components, such as by initial (static) configuration, (e.g., by means of an Architecture Description Language (ADL)) or by using a dynamic configuration approach such as emerging architectures, dynamic architectures or supervised reconfigurations.

A trend in component design is that components are more autonomous, more complex, have more responsibilities and the number of components in applications increases. This trend is obvious with agent architectures but even those component designs that are not agent-based have the same characteristics.

In designing industrial applications today, there is an emerging consensus that some kind of homogenous, high-level information representation is needed for agents, such that a predictable behavior can be achieved for all agents. A fuzzy automaton-based approach offers clear benefits for developing agents of reconfigurable architecture.

The Intelligent Fuzzy Controllers Laboratory has been developed in the Department of Electrical and Computer Engineering of Western Michigan University with the help of a DURIP grant by the Department of Defense [6] and generous donations by ABB Automation Technology Products. This new lab is to support research, the development of advanced courses, and graduate projects in the area of intelligent controls for large, complex hybrid systems. One of the targeted applications is the supervisory level of flatness control for cold rolling mills.

A research project is currently underway to develop a Generic Encapsulated Fuzzy Automaton Software Agent for Object Oriented Control Systems. One of the objectives of the project is to develop and evaluate the performance of new methods and architectures for this area of research.

II. BACKGROUND

Among the problems that characterize industrial process control innovation, and which are not domain-related, the most difficult ones are as follows: (a) how can new knowledge be introduced into a system, (b) how can the system activate stored domain knowledge in an autonomous way, (c) how can the knowledge be validated (or otherwise detected as inappropriate) and (d) how can the system recover if the new, activated knowledge (or the currently active knowledge) is not suitable to handle the situation at hand.

The use of agent technology helps to answer question (a), in that paradigm an agent is defined as an architecture-neutral, mobile software entity that can act on behalf of a human and have decision making capabilities similar to a human [1]. The theory of software dynamical architectures describes the dynamics of the environment in which agents can act. The software architecture, by using an architecture broker, mediates the information flow among agents in order to achieve overall population-level goals, and also makes various resources (computational power, sensors, and actuators) available to the agent. The activation of the appropriate knowledge is accomplished via identification of operations performed by agents that are capable of finding the right model among a set of available models [4]. Thus the

answer to problem (b) is defined as the appropriate interaction among the software architecture mechanisms and the agents. Agents with model-based target seeking algorithms utilizing fuzzy logic, interacting in a distributed large system are strong candidates to replace traditional communications channels among units of a distributed system at a higher system level.

A typical agent-based application consists of an environment in which the agents can communicate and a library of agents activated either by already active agents or by a specialized supervisory agent. Some of the typical properties of such systems are:

1. Transparency for distributed, networked operations. For example, activation of the remote agents is done in the same way as for local agents, i.e., a redistribution of agents is done seamlessly and without functional degradation. The interconnection of local agents is done in the same way as for remote agents, etc.
2. Architecture integrity supervision. The system has built-in functionality that checks that all necessary agents are active and reachable. If this is not the case, the supervisor can restart missing components, or agents.
3. Distributed basic functionality. This is available in the environment such as access to the file system, database, timers, etc. across the distributed system in a homogenous way.

One of the useful functions, which the research presented here is focusing on, is a state machine model that exists identically in each agent. Solutions of this type are common today and have been described earlier, e.g., in [7]. This is the supervisory state level and all agents have the same set of states. At any time instance each agent can have a different active state. The purpose of this state machine is that other agents, or a supervisor can query if an agent is in a proper active state and if it is not then will react appropriately. Each agent is responsible for the state transitions of its own state machine. External agents can only query the current state.

A fuzzy automaton [2] can implement new knowledge by means of the states of the goal path of an event-driven, sequential control algorithm while providing an effective approximation method to model continuous and discrete signals in a single theoretical framework. With respect to problem (c) knowledge validation is achieved by quantifying the degree of deviation from the nominal operating conditions due to unexpected events caused by either abrupt, or gradual changes in the system, or in the environment of the system. With respect to problem (d) these properties can facilitate the development of computationally inexpensive fault detection and identification (FDI) algorithms, and automated recovery from faults (subject to further research). Regarding to FDI, the evaluation of the state transitions between states of a large, complex system will be done by

focusing only on clusters of relevant states along the goal path. A reconfigurable virtual fuzzy automaton will be used to model those clusters of states [3].

III. THE SOFTWARE AGENT GEFASA

The development of a software environment for the Intelligent Fuzzy Controllers Laboratory has been divided into three phases. The primary objective of Phase 1 is to create a program referred to as the Generic Encapsulated Fuzzy Automaton Software Agent (GEFASA) aimed for Object Oriented Control Systems using Java. This program is then loaded as a reconfigurable agent in the FSA (Flatness Server Architecture) broker running on the Stressometer System by ABB. The Stressometer System includes several industrial computers and PLCs (Programmable Logic Controllers) along with ABB's software development environment for complex control systems.

The notion of fuzzy automaton in this research is based upon the premises as follows: it can stay in more than one crisp (i.e., Boolean) state at once, to a certain degree in each. Those degrees are defined by a state membership function. That is, a fuzzy state is made of some crisp states. For each fuzzy state there is just one dominant crisp state, though, for which the state membership is a 1 (full membership). Each dominant state is associated with a linguistic model of the plant for inference. For each fuzzy state a composite linguistic model is devised by the composition of the linguistic models of all contributing crisp states. A set of associated state membership degrees are used for the composition. The transitions between fuzzy states are based upon the transitions defined between their dominant crisp states. There is an underlying Boolean finite state machine to implement the fuzzy automaton. The states of this Boolean automaton are the dominant (crisp) states of the fuzzy automaton. Fuzzy (i.e., continuous signal) inputs (and outputs) are mapped to sets of two-valued logic variables. In addition, other input signal types include two-valued ones and continuous signals with a threshold. Conditions for state transitions are defined by any combination of signals of these three types. Defuzzified (and fuzzy) outputs are obtained by computing the compositional rule of inference using the composite linguistic models. Two-valued (Boolean) outputs are devised in the usual manner [5].

The fuzzy automaton model implemented in the FSA broker along with an input stimuli generator is shown in Fig.1. The GEFASA model is made up of three fuzzy agents referred to as FSMSIM, FSM, and FSMPanel, respectively. The FSMSIM module in the GEFASA generates random fuzzy inputs and Boolean inputs for the FSM module to perform fuzzy model building and inference operations. The block diagram outlining the FSMSIM agent in the FSA architecture is as shown in Fig. 2. The FSM module generates the composite linguistic model $R^* = G * R_s$ once for a given model and then performs the fuzzy model building and inference and the necessary state change

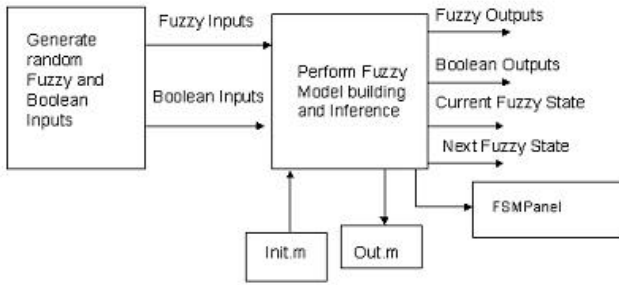


Fig. 1. Functional block diagram of the GEFASA model

computations. It also sends the results to the FSMPanel module for display purposes. The basic flowchart for the FSM agent is shown in Fig. 3. These three agents have been developed and loaded successfully into the FSA broker.

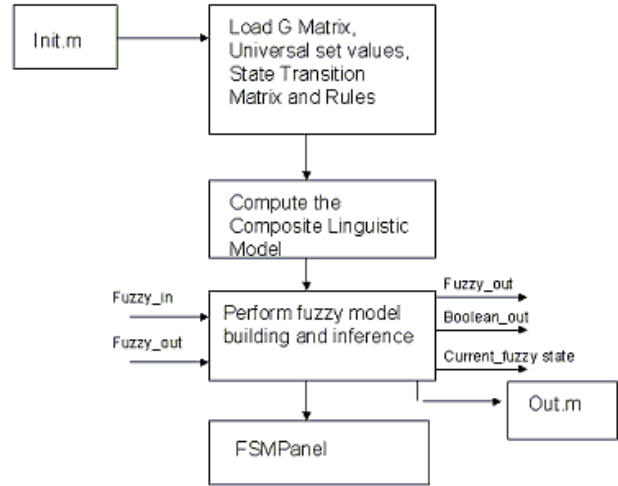


Fig. 3. Flowchart for the FSM agent

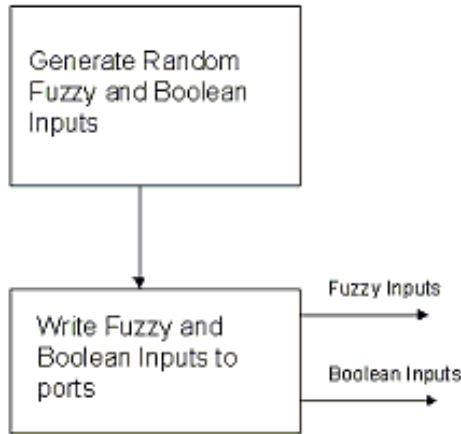


Fig. 2. Block diagram outlining the FSM agent in the FSA broker

IV. THE MILL MODEL

Phase 2 involves the development of a Mill model by making use of the GEFASA agent created in the Phase 1. The idea is to provide Flatness Control in a closed loop using a PID controller to provide for a reference to the fuzzy automaton-based control. The Mill model is shown in Fig. 4. Mill1 is a function that models a cold rolling mill with 3 actuators and 32 measurement zones. PID stands for an industry-standard Proportional-Integral-Derivative Controller that is modeled as a simple recursive function.

It has two parts:

- static computation that has an inverted model $ex=A \setminus e$ (pseudo-inverse function),
- dynamic section that includes a PID, or alike.

The objective is to replace the computation for the pseudo-inverse function (i.e., replace $ex = A \setminus e$ with a fuzzy

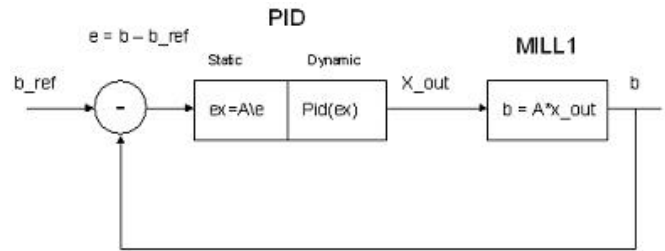


Fig. 4. Mill model for Flatness Control in close

approach) and leave the PID section alone. The flatness control program for the PID section is responsible for two tasks:

- convert the error into an actuator setup, e.g., the least square of $A \setminus b$,
- implement the PID algorithm that makes a smooth conversion between different actuator setup points.

But instead of using the least square method the program follows the following rule for converting the error into an actuator setup: the actuator that takes out more of the error is given more power to do it. The input to the algorithm is e , referred to as the flatness error. It is a vector of the size 32×1 . The output from the algorithm is ex that is of size 3×1 . It is the set point for the actuators.

The algorithm has the steps as follows:

Step1.

Compute the actuator parameters as if each actuator compensated alone for the error defined as $x1$, $x2$, and $x3$, respectively.

Step2.

Compute the residue $r = e - Ax$. The actuator that works the best will have the smallest residue. Factor A that determines the percent of the error compensated by each actuator must be determined in this step. Let $A1, A2$, and $A3$ be columns in matrix A that describes each actuator, $x1, x2, x3$ be the parameters of each actuator, and the factors $k1, k2$, and $k3$ are as follows:

- $k1$ - part of the error taken by Actuator 1,
- $k2$ - part of the error taken by Actuator 2,
- $k3$ - part of the error taken by Actuator 3, and
- $K1+K2+K3 = 1$.

Then

$$\begin{aligned} A1 * x1 &= k1 * b \\ A2 * x2 &= k2 * b \\ A3 * x3 &= k3 * b \end{aligned}$$

Step3.

Compute factors $k1, k2, k3$ as follows:

1. Compute the Euclidian norm of the residue $r = A1 * x1 - b$ which provides the error left if Actuator 1 acted alone. Do the same for Actuators 2 and 3.
2. The ‘best’ actuator is the one with the lowest residue but the actuator with the lowest residue must have the highest factor. That is obtained by computing $q1 = (r1+r2+r3)-r1$ and then $k1 = q1/(q1+q2+ q3)$.

Step4.

Compute the effective actuator parameter by making use of the following rule: the actuator that can compensate more of the error will work alone until some other actuator becomes more prevalent.

Hence, the program is state-based with three states, each of which corresponds to one specific actuator working alone.

The fuzzy logic section of the program consists of the following steps:

Step1.

Define crisp states. Set up conditions for triggering a state change. Define fuzzy rules and their relationship with each crisp state

Step2.

It has three parts:

1. the initial part where the membership functions are computed and rules are given,
2. the dynamic part where the fuzzy inference is done, and
3. the interface to the program for actuator selection

Step3.

It is the implementation step:

1. Compute the flatness error norm: $\|e\| = \|b-b_{ref}\|$ and fuzzify it.
2. Compute the inverse of the norm of the residue from each actuator (if it acted alone) $\|r_i\| \ i = 1,2, 3$ and fuzzify them.
3. Compute the parameter of each actuator $x1, x2, x3$ and fuzzify them.

This step results in 7 fuzzy inputs $\|e\|, \|1/r1\|, \|1/r2\|, \|1/r3\|, \|x1\|, \|x2\|, \|x3\|$.

Some of these fuzzy inputs determine the states as shown in Table I below.

TABLE I
DEFINITION OF FUZZY STATES

State	Norm(e) – in(1)	Norm(1/r1) – in(2)	Norm(1/r2) – in(3)	Norm(1/r3) – in(4)
1	High	High	Any	Any
2	High	Any	High	Any
3	High	Any	Any	High
4	Low	Any	Any	Any

The state transitions are as shown in Fig. 5.

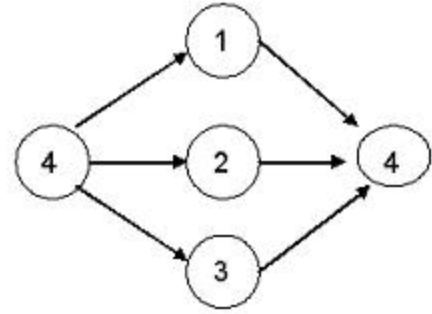


Fig. 5. State Transition Diagram

V. THE MILL MODEL IN THE FSA

The MillModel in the Flatness Server Architecture (FSA) consists of two fuzzy agents referred to as MillFSMSIM and MillFSM. The whole model is implemented in a closed loop control in the Flatness Server Architecture as shown in Figure 6.

The module MillFSMMSIM computes the MillModel that models the mill with 3 actuators and 32 measurement zones. The input to this module is the vector x_{out} that is obtained as a defuzzified value from the fuzzy inference process. Vector x_{out} is the command vector for some mechanical action. It is the input to the plant model that will again compute the error e, and the cycle will continue in this way equal to the number of cycles given by T_MAX (default is 100). The MillFSMSIM module consists of the following inputs and outputs:

1. input: x_{out} (3x1 matrix),
2. output: e (32x1 matrix), and
3. b_{ref} (32x1 matrix) that is the set point for the “reference profile” for flatness. The mechanical process that is being controlled must achieve a result equal to this matrix. The current profile b is calculated and is compared with b_{ref} to determine the type of mechanical action required.

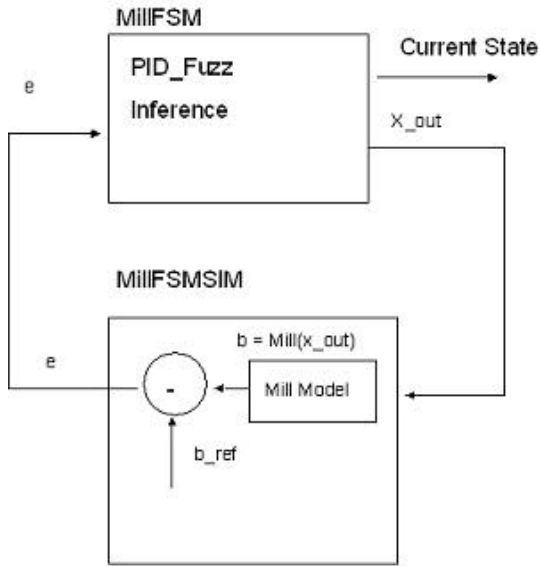


Fig. 6. Mill Model in the FSA architecture

The difference $e = b - b_{ref}$ is an input to the control system implemented by the fuzzy automaton. The flowchart for the MillFSMSIM agent is shown in Fig. 7.

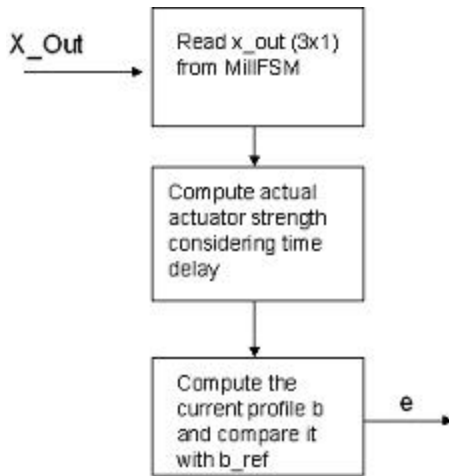


Fig. 7. Flow chart for the MillFSMSIM agent

The MillFSM module computes the composite linguistic model $R^* = G \cdot R_f$ once for a model and then performs the fuzzy model building and inference computations as well as evaluates the necessary state changes. The module has one input vector e (32 x 1) which is the error calculated in the MillFSMSIM module, one output vector x_{out} (3x1) which is given as input to the MillFSMSIM module, and one output value $current_state$ which depicts the necessary state changes. This module provides both fuzzy outputs as function of fuzzy inputs and Boolean outputs as function of Boolean inputs. The MillFSM module has the following inputs and outputs:

1. input e (32x1 vector), the error value computed in the MillFSMSIM module by making use of the above mentioned mill model,
2. output x_{out} (3x1 vector), obtained as a defuzzified value from the fuzzy inference process that acts as the command vector for some mechanical action.
3. output $Current_State$, describes the necessary state changes.

The flowchart for the MillFSM agent is shown in Fig. 8.

VI. EXPERIMENTAL SYSTEM

In Phase 3 the system will be connected to an ABB R&D Lab in Sweden through the Internet and will periodically receive input files that have been passed to the controllers of an actual cold rolling mill, as well as files describing the performance of the plant. This system will process the input files for the controllers using the Mill model developed in the Phase 2 and the fuzzy automaton model.

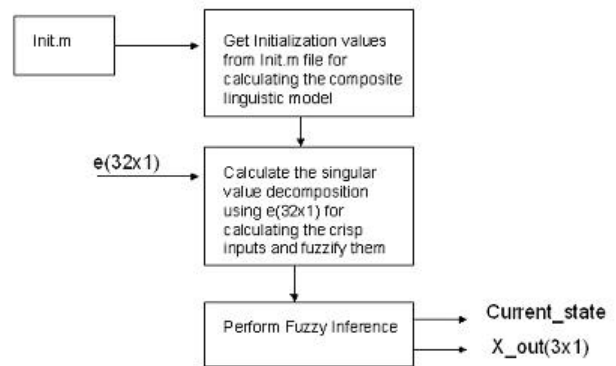


Fig. 8. Flowchart of the MillFSM agent

Its results will be compared with the output results produced by the controllers of the cold rolling mill to evaluate the performance of the method using the fuzzy automaton. Phase

3 will create an experimental system to conduct research on algorithms and architectures for supervisory control using fuzzy automata.

VII. CONCLUSION

The purpose of this paper is to explore the possibilities of using fuzzy automata at the supervisory state machine level and to increase functionality. The potential benefits are self-configuring (emergent) agent populations, improved semantics and ontology content of the information passed between agents and improved prediction for advanced fault detection, safety and active redundancy. The drawbacks of the approach are a large number of parameters and rules that need to be generated and tuned. Thus a self-tuning and easy to generate rule set would give benefits (subject of further research). Another limitation is that the system, as investigated by us, is restricted to mainly numerical applications such as real-time control and modeling. Systems which are heavy in symbolic / text processing can be also adapted for this type of agent architectures, but are currently not in the scope of this research.

REFERENCES

- [1] M. Knapik, J. Johnson, "Developing Intelligent Agents for Distributed Systems, Exploring Architecture, Technologies and Applications", McGraw-Hill, 1998.
- [2] J.L. Grantner, G. Fodor, D. Driankov, "Hybrid Fuzzy-Boolean Automata for Ontological Controllers", *Proceedings of the 1998 IEEE World Congress on Computational Intelligence, FUZZ-IEEE'98*, Vol. I, pp. 400-404, Anchorage, Alaska, May 4-9, 1998.
- [3] J. L. Grantner, G. A. Fodor, D. Driankov, "The Virtual Fuzzy State Machine Approach - A Domain-Independent Fault Detection and Recovery Method for Object-based Control Systems", *18th International Conference of the North American Fuzzy Information Processing Society - NAFIPS'99*, June 10-12, 1999, New York, NY, Proceedings, pp. 158-162.
- [4] "MAS'99 Multi-Agent Systems in Production", *Proc. of the First IFAC Workshop*, Dec. 2-4, 1999, Vienna, Austria, Elsevier Science, 2000.
- [5] J. L. Grantner, G. A. Fodor, "Fuzzy Automaton for Intelligent Hybrid Control Systems", *2002 World Congress on Computational Intelligence, WCCI'02/FUZZ-IEEE'02*, May 12-17, 2002, Hilton Hawaiian Village Hotel, Honolulu, HI, in the CD Proceedings, ISBN: 0-7803-7281-6
- [6] J. L. Grantner, "Intelligent Control Laboratory for Complex Hybrid System", DURIP'2001 Award (DAAD19-01-1-0431)
- [7] Selic, B., Gullekson, G., Ward, P.T., "Real-Time Object-Oriented Modelling", John Wiley & Sons Ltd., New York, 1994.