

The Virtual Fuzzy Automaton Approach to the Problem of Global State Evaluation in Multitask Control Systems

Janos L. Grantner

Department of Electrical and Computer Engineering
Western Michigan University, Kalamazoo, MI 49008-5066, USA
Email: grantner@unix.cc.wmich.edu

George A. Fodor

ABB Automation Products AB, S-721 67 Vasteras, Sweden
and
Department of Electrical and Computer Engineering
Western Michigan University, Kalamazoo, MI 49008-5066, USA
Email: george.fodor@se.abb.com

Abstract

Real-world applications are often controlled with systems capable of executing several simultaneous control tasks. Each control task works in parallel with other control tasks to achieve common control goals. A desirable property for the design, analysis and supervision of such systems is the representation of an encapsulated (overlaid) state set which extends over the states of several tasks and characterizes the current control situation in application-related terms. The paper describes the reason why traditional power-set state based approaches are not enough practical in the case of large, complex systems, and suggests a solution based upon the concept of a virtual fuzzy finite state machine.

Keywords: fuzzy control, fuzzy automata, discrete event systems, ontological control, programmable controllers.

1. Introduction

Control applications in the areas of robotics, process control, manufacturing and the chemical industry are specified in terms of a large number of discrete states [1]. Continuous processes are normally embedded in the discrete state representation by using a suitable discretization of the continuous signal intervals. Even in the case of relatively small systems, the number of discrete states is often very large. In practice, system engineers make the size of the state space manageable by introducing a number of parallel tasks such that each task controls certain aspects of a plant. Each task has a set of discrete states that can be represented in ways equivalent to Petri nets [2], discrete event systems [3], or hybrid systems [4]. An on-line supervisory system is normally employed to

translate the current information of the control system into some other form of information that is meaningful in terms of the application specification. Since the tasks need not always to be synchronized, it is very difficult to determine what is the current global state. Often the number of states is quite large, hence, a characterization of the global state that uses all states of all tasks is very difficult to process by a supervisory system. It would be more desirable to have an application specific method for state characterization in terms of the process signals alone. The solution proposed in this paper is based upon the following premises:

1. All tasks are normally processing data placed in a real-time database. The global control state is fully characterized by data in that database.
2. A mapping of the states of the continuous signals stored in the database into a discrete state representation that corresponds to the given system specification can be accomplished by using a virtual fuzzy automaton.
3. The state transitions allowed at the fuzzy state machine level are made explicit.
4. A supervisory control unit backed up by a virtual fuzzy automaton can determine (a) the current global state and (b) whether the current state transition is one among the state transitions allowed.

Section 2 outlines the mapping principle. Section 3 gives an example, followed by the theory of the Virtual Fuzzy State Machine in Section 4.

2. Determination of the Discrete to Fuzzy State Mapping

We consider a control system with m tasks T_i ($i=1, \dots, m$). Each task has a state set $S_i = \{s_{1,i}, s_{2,i}, \dots, s_{n_i,i}\}$, where m is the total number of tasks, i is the index of an

arbitrary task and n_i is the number of states of Task i . Each state set S_i can be further divided in two state sets: an application state set S_i' and an implementation state set S_i'' . An application state identifies a particular, specified state of the environment under control, e.g., "the water in the tank is at the maximum level", whereas an implementation state identifies a particular state of the controller, e.g., "regulator R1 is acting".

The global state set of the system can be given as a specified subset S^p of the power set $S^* = \{S_1 \times S_2 \times \dots \times S_m\}$. One power-state in S^p contains all task states that can possibly be true simultaneously with a given set of sensors and actuators, in an application. As for states of S_i , the states of S^p have either implementation or application classification. Correspondingly, the states in S^p can be one of the three kinds:

S^a – consists of those power states that have the maximum number of application states from S_i' .

S^q – consists of power states with a non-maximum number of application states from S_i' .

S^p – consists of all power states that can possibly be reached by the control system.

In what follows "control state" denotes a state s_{ij} of a task T_i in the system. An "application state" is a state in S^a . A "global state" is a state in the set S^p . Clearly, $S^a \subseteq S^q \subseteq S^p \subseteq S^*$.

As it is shown in [5][9], the states in $(S^p \setminus S^q)$ (and which have no application states) belong to the so called goal-seeking operation (GSO) that determines the actual goal path for a given control situation. The GSO states are not of interest for an external observer of the system who supervises the process since these states implement the internal chore of a control task.

It can be shown that a typical state trajectory in the space S^p starts with a state in S^a , continues in S^q , then possibly in $(S^p \setminus S^q)$, and finally returns via S^q to some state of S^a (Figure 1).

The following problems are encountered when working with the global state set S^p :

1. The states in the parallel tasks are not necessarily synchronized, thus several distinct global states in S^p may represent the same application state (the application state set is sparse relative to S^p).
2. Distinct application states may be represented by the same state in S^p . The difference appears in the form of particular state sequences in which the component states have been materialized.
3. The transitions at the global state level are not easily tractable in the system since they are not represented.

An often encountered problem is that although particular states and state transitions at the task level are right, the transitions at the global state level may not be correct, and it is very difficult to show that this kind of error has occurred in a working system. The lack of tractability is correlated with the complexity of the system, and with problems 1 and 2 above.

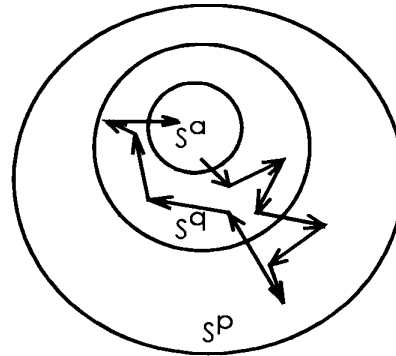


Figure 1 - State trajectory

The solution proposed in this paper is based upon the property of the Virtual Fuzzy Finite State Machine (VF-FSM) to identify discrete states using continuous, fuzzified intervals. The application state set is normally well specified in terms of the signals stored in the database. The distribution of the application states over various tasks may be less specific, since it is implementation-dependent. Therefore, it seems advantageous to map the application states into the global state set using the real-time database signals. Since these signals are of a mix of continuous and discrete type, a fuzzy state machine is employed to model the global states and the state transitions of the system.

The transitions of the application power states can be captured by "locking-in" the virtual fuzzy state machine when the execution is in the S^q set and then tracking all the state transitions reachable from those states. The states are monitored using direct sensor and actuator information from the database. The supervision of the states continues all the time, even when the states are in the space $(S^p \setminus S^q)$, but in this case the partial state S^q (i.e. the part of S^q which contains the monitored signals) should stay unchanged.

In order to clarify the environment within the approach can be applied, we shall consider the architecture of a typical control system given in Figure 2. A control system, that is implemented using a programmable controller (PLC) or an open control system (OPC), has a real-time database which can be accessed to by all tasks. In this real-time database, both sensory data (inputs) and actuator commands (outputs) are stored. In addition, the system has also dynamic data. A real-time operating system launches and synchronizes the tasks. In more advanced systems, a hierarchy of control levels is implemented.

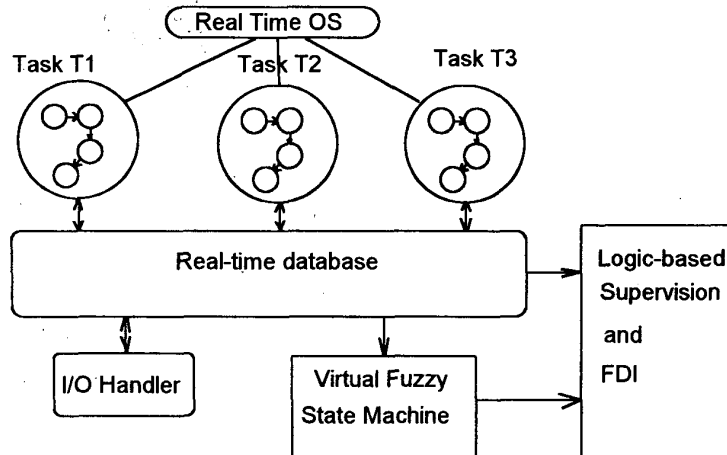


Figure 2 - Control Architecture with Virtual Fuzzy State Machine

Since the algorithm works with sequences of states of different types, we introduce a notation that gives the sequence of states and their types only. We consider the time as an integer representing the interpretation sequence of the control program.

a_k = is a state in S^a which is true at some time k .

q_k = is a state in S^q which is true at some time k .

i_k = is a state in $(S^p \setminus S^q)$ which is true at some time k .

The separation algorithm by type, for each state in each task, is accomplished in two steps:

Step 1: the type of each state is determined, q or i , by analyzing the signals that appear in the state formula. If in a state no sensory signals are used, then the state is of type i else it is of type q .

Step 2: A state is characterized by a Boolean formula made up of a mix of dynamic data and sensory signals. Among the states of type q , it can be determined which state has the largest number of sensory variables. Those states are then marked as of type a .

The execution of these steps yields a timed sequence. A sample for a task T_i may take the form:

$$T_i = a_1, i_2, q_3, q_4, a_5, q_6, a_7$$

For a control system with m tasks, there are m such sequences. Next, each sequence is further modified using the following rules: if in a sequence there is a pair a_j, i_{j+1} or q_j, i_{j+1} then it can be replaced with a_j, a_{j+1} or q_j, q_{j+1} , respectively. The reason for this rule is that a state of i -type does not modify sensory data, hence, the previous sensory data is still valid.

In this way, by using recursion, each sequence has got finally states of only type a and q , respectively. For each of these states, one considers only the sequence of changes of the sensory and actuator signals. Some of the signal ranges

may overlap. Each transition in the sequence is associated with an index. In this way, a specification of the state transients is obtained that will be used to create an instance of the VF-FSM.

3. Example

A cooling system used for cold-roll milling consists of a number of nozzles placed along the width of the metal strip being rolled. The cooling operation is controlled by a complex system. Each nozzle can be individually turned on or off, or several nozzles can cool the same spot of the roll such that a specific level of cooling can be reached. The aim of the control system is to achieve a number of simultaneous goals as follows:

1. Basic cooling: the strip needs a basic cooling, and lubrication with a relatively low level of lubricant is applied uniformly along the strip. The basic cooling is applied when the roll's temperature T is relatively cold after a other type of cooling operation has been applied for a longer time: $T \leq T_0$.
2. Cooling: it is applied when the roll is too hot: $T > T_0$.
3. Bending support: in order to achieve a certain profile of the strip, a mechanical bending is applied on the roll. The mechanical bending is limited by mechanical constraints, therefore a thermal support provides for the same effect like a mechanical bending. This is realized using a so called *thermal profile* (i.e. cooling applied more intensively at the middle of the roll and less towards the edges of the roll). If ΔB is the required bending (mechanical plus thermal), the bending support is applied when $\Delta B > B_{\text{mech}}$.
4. Flatness control: by applying cooling at certain points, a uniform flatness of the strip can be achieved. The flatness is measured on-line with a measurement roll.

If ΔF_k is the flatness error in certain zone k , then cooling is used as a flatness actuator when $\Delta F_k > F_0$.

These conditions are shown in the table below:

Power State	Meaning	ΔT	ΔF	ΔB
S_1^q	Basic Cooling	$\leq T_0$	any	any
S_2^q	Cooling	$> T_0$	=0	=0
S_3^q	Bending Support	$> T_0$	=0	$> B_0$
S_4^q	Flatness Actuator	$> T_0$	$> F_0$	=0

Table 1.

The states of this table can appear in any combination. If a state has less than all of the three variable types (ΔT , ΔF , ΔB) then is of type "q", else it is of type "a". All these state combinations make up the *application state set*. There are many other states in this control system that appear as a result of component faults, measurement requirement, speed conditions, manual operations, etc. An implementation of such a cooling control system may have as many as 25 different tasks, each of them has got several thousand states. However, when using the VF-FSM only these four variables are relevant. Thus, an application can be made more robust and easier to track if the application state set is determined independently of the rest of the control states. Then, by correlating the actual states with the application states represented by a virtual fuzzy automaton, logical supervision, fault detection and isolation can be achieved.

4. Virtual Fuzzy Finite State Machine

The Virtual Fuzzy Finite State Machine (VF-FSM) is built upon the model of the Hybrid Fuzzy-Boolean Finite State Machine (HFB-FSM) [8]. The HFB-FSM is given by the formulas (1), where X_F and Z_F stand for a finite set of fuzzy inputs and outputs, respectively, W_B and U_B stand for a finite set of two-valued logic inputs and outputs, respectively. Defuzzified outputs are denoted by z_C , R^* is a composite linguistic model (3), and \circ is the operator of composition. Each crisp state of the HFB-FSM is characterized by an overall linguistic model R_s , or by a set of linguistic sub-models in the case of multiple-input-single-output (MISO), and multiple-input-multiple-output (MIMO) systems.

$$\begin{aligned} Z_F &= X_F \circ R^* \\ R^* &= G(R_s) \\ z_C &= DF(Z_F) \end{aligned}$$

$$U_B = f_u(y_B) \quad (1)$$

$$X_B = B(X_F)$$

$$Z_B = B(Z_F)$$

$$Y_B = f_y(X_B, W_B, Z_B, y_B)$$

A fuzzy state is defined by a crisp (Boolean) state and a state membership function

$$S_{F_k} : S_k, g_{S_k} \quad (2)$$

where S_{F_k} stands for fuzzy state k , S_k represents crisp state k , and g_{S_k} is the state membership function associated with S_k . G stands for the matrix of state membership functions, X_B , Z_B , Y_B , and y_B are two-valued Boolean input, output and state variables, respectively. B stands for a Fuzzy-to-Boolean transformation algorithm to map a change in the status of a fuzzy variable into state changes of a finite set of corresponding Boolean variables. The z_C crisp values of the fuzzy outputs are obtained by evaluating a defuzzification strategy, DF. On the basis of the concept of a fuzzy state, the automaton stays in a number of crisp states simultaneously, to a certain degree in each. One of these states is referred to as a dominant state for which the state membership function is a 1 (full membership). The initial concept of a fuzzy FSM based on a Boolean FSM and the State Membership Functions, in this context, have been introduced in [6] and [7], respectively. A formal representation was given in [11].

For each fuzzy state of the HFB FSM model, a R_k^* composite linguistic model is created from the finite set of R_{S_i} overall linguistic models ($i=1, \dots, p$). Let the HFB FSM be in fuzzy state S_{F_k} , then

$$R_k^* = \max[\min(\beta_1^k, R_{S_1}), \min(\beta_2^k, R_{S_2}), \dots, \quad (3)$$

$$\dots, \min(\beta_k^k, R_{S_k}), \dots, \min(\beta_p^k, R_{S_p})]$$

where $\beta_1^k, \beta_2^k, \dots, \beta_p^k$ stand for the degrees of state membership function g_{S_k} , and $R_{S_1}, R_{S_2}, \dots, R_{S_p}$ are the overall rules in crisp states S_1, S_2, \dots, S_p , respectively. With (3), a SISO system is assumed. In adaptive systems R_k^* is not stored in memory, it is dynamically created by computing (3), instead. By modifying the β degrees of the state membership functions on-line, new R^* composite linguistic models can be created under real-time conditions. The transitions between active composite linguistic models are determined by the state transients of the HFB-FSM.

The state transients of the HFB FSM are specified by means of a sequence of changes in the states of the fuzzy inputs and outputs, as well as of the two-valued inputs. The changes in the states of the fuzzy inputs and outputs are mapped into the corresponding sequence of changes of Boolean input and output variable sets, respectively, using the B algorithm [11]. In this domain, those changes are joined by the state changes of the two-valued inputs. This combined Boolean input/output sequence specification is used to synthesize the crisp FSM section of the HFB-FSM. Hence, the HFB-FSM model allows the integration of fuzzy and two-valued logic specifications to describe a system's behavior. The integrated treatment of fuzzy and two-valued signals is of great importance for designing complex systems in which, in fact, many signals are of two-valued and need to be dealt with as such.

Since the state set of a complex system may consist of thousands of states, it would be impractical to design a fuzzy automaton of that size. However, only a small partition of the total state space of each task is relevant at any particular state transition, as it outlined in Section 2. A supervisory controller monitoring the flow of states determines a segment of the transition graph of the relevant states prior to the decision on the next state transient, and creates an instance of the VF-FSM to handle it. The following information will be downloaded to the VF-FSM: the active sets of fuzzy and Boolean inputs and outputs, respectively, state membership function degrees, the actual mapping scheme between fuzzy and Boolean subintervals of the B algorithm, the initial state, the state transition graph along with the conditions, the overall linguistic models if inference will be needed, and the status of the active fuzzy and Boolean signals [10]. The VF-FSM will then decide on the transient to the next state, infer outputs (if needed), and pass all of these information back to the supervisory controller. The configuration process will repeat each time when a new instance of the VF-FSM should be created to track the current segment of the state graph.

5. Conclusion

The method suggested here employs the Virtual Fuzzy Finite State Machine to the problem of state supervision for real-time control systems. The results are of immediate interest for controlling large systems that require operator decisions in complex plant situations. From a research perspective, the method provides for a new approach to fault detection and isolation (FDI), program specification and synthesis for large, complex systems, and the design of fuzzy hardware accelerators for hybrid systems.

6. Acknowledgement

This research is supported by ABB Automation Products AB, Sweden.

7. References

- [1] A. Sanchez, Formal Specification and Synthesis of Procedural Controllers for Process Systems. Springer Verlag, 1996.
- [2] R.G. Willson, B.H. Krogh, Petri Net Tools for the Specification and Analysis of Discrete Controllers, IEEE Trans. on SW Eng. Vol. 16. no. 1, January 1990.
- [3] Wonham, W. M., A control theory for discrete event systems, in *Advanced Computing Concepts and Techniques in Control Engineering*, M.J. Denham, A.J. Laub (editors), pp. 129-169, Springer-Verlag, 1988.
- [4] T. A. Henzinger, Z. Manna, A. Pnueli, Refining Temporal Specifications into Hybrid Systems, Workshop on Theory of Hybrid Systems, Lyngby, Denmark, Oct. 1992.
- [5] G. A. Fodor, *Ontologically Controlled Autonomous Systems, Principles, Operations and Architecture*, Kluwer Academic Publishers (1997).
- [6] J. Grantner, M. Patyra, Synthesis and analysis of fuzzy logic finite state machine models, WCCI'94, Proceedings of the FUZZ-IEEE'94, Orlando, FL, June 26-29, Vol. 1, 205-210, (1994).
- [7] J. Grantner, M. Patyra, M. Stachowicz, Intelligent fuzzy controller for event-driven real-time systems and its VLSI implementation, in *Fuzzy Control Systems* (Eds. A. Kandel, G. Langholz), CRC Press, Boca Raton, FL, USA, 161-179, (1994).
- [8] 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, Anchorage, Alaska, May 4-9, 1998.
- [9] G. Fodor, D. Driankov, A New Approach to On-line Fault Identification in PLC Control, Proceedings of the SAFEPROCESS'97, The University of Hull, UK, pp. 597-602, August 26-28, 1997.
- [10] G. Fodor, J.L. Grantner, D. Driankov, Modeling the Real-Time Recovery of Complex Control Systems – A Fuzzy Approach, Proceedings of the IEEE International Conference on Systems Man and Cybernetics, Oct. 12-15,

1997, Orlando, Florida, USA.

[11] J. Grantner, Design of Event-Driven Real-Time Linguistic Models Based on Fuzzy Logic Finite State Machines for High-Speed Intelligent Fuzzy Logic Controllers, Thesis for the Degree Candidate of Technical Science, Hungarian Academy of Sciences, Hungary (1994).