Execution Monitoring of Domain-Independent Controllers for Hybrid Systems

Janos L. Grantner[1] and George Fodor[2]

[1] Western Michigan University, 3058 Kohrman Hall, Kalamazoo, MI 49008-5066, USA
grantner@unix.cc.wmich.edu

[2] ABB Automation Products AB, S-721 67 Vasteras, Sweden
george.fodor@seipr.mail.abb.com

Abstract

An autonomous control system working in a real-world environment may encounter events that are outside the bounds of the established safety criteria. In the case of a hybrid system, unexpected situations in its environment may violate assumptions that have been used to design the continuous and/or the discrete segment. Hence, it is difficult to design a control unit that can detect and recover from such errors. An execution monitoring algorithm for this type of faults will draw practical interest only if it is application independent. This paper presents an execution monitoring and fault detection method for hybrid systems along with its limitations. This approach addresses a new class of problems encountered with hybrid systems that needs to be considered in the design and analysis phases.

Introduction

A safety critical system is designed with safety criteria established before the system is commissioned. As it has been extensively described in literature, a system working in a real environment may encounter events that are outside of the bounds of the established safety criteria. This phenomenon is well known both for continuous (e.g. robust control) and discrete systems (e.g. execution monitoring). In particular, a safety critical hybrid system (i.e. a proven correct hybrid system) working in a real environment may encounter situations in which either the continuous, or the discrete part of the system acts outside of its expected operating ranges (statically, or dynamically). Hence, it is difficult to design a control unit that can cope with both problems. Moreover, an execution monitoring system will draw practical interest only if it is domain independent in the following restricted sense:

- An algorithm is developed for execution monitoring such that it is applicable for a large class of systems.
- The particular data the algorithm uses can be extracted before the control goes into effect.
- The data extraction uses yet another algorithm that is based upon a formal description (model) of the system, or heuristics.

In other words, the run-time execution-monitoring unit is application independent, but the input it uses is application-dependent and extracted off-line from a model of the process.

In this paper, we present a method based upon a hybrid process representation. Although other modeling formalism may be conceived as well, we argue that this method provides a good balance between a full model-based approach that can provide sufficient error detection capability and a domain-independent approach that may not detect accurately application-dependent faults. The results are illustrated using an example of industrial-strength.

Cellier has introduced numerical methods (Cellier et al. 1993) for simulating numerically hybrid models. His method requires a completely separate description of the continuous segment (using sets of differential equations) from the description of the accompanying events. Often it is assumed that the separation between the continuous and discrete segments is fixed and given by a model. However, in complex applications the separation of the description of the continuous segment is not practical due to its size, the lack of accompanying models and due to the programming method employed. Even in the case when a system has formal continuous and discrete models, when implemented on a real-time digital computer, a large chunk of the continuous model becomes discrete implementations. The discrete segment may consist of numerical algorithms, initialization and reset values for integrators, range checks, parameter changes using rules, etc. It is also true that some discrete model may become continuous when implemented. Thus there is a blurred boundary between the continuous/discrete segments of the model and of the implementation. We will refer to this "moving" boundary as the hybrid model slip (HMS).

Another special case of execution monitoring is about system stability. In a number of papers, instability in hybrid systems has been considered using energy transfer models that can be analyzed with Lyapunov functions. The

HMS problem makes the application of the Lyapunov functions difficult. In addition to the traditional instability, as it is shown in (Fodor 1997) and in this paper, instability at discrete state level may develop that is not related to energy transfer (the control system may be in a steady state from a continuous perspective, but may not be stable).

This paper proposes a uniform execution modeling of the continuous and discrete segments using a fuzzy finite state machine. The fuzzy automaton monitors the deviation from expected next states in order to detect failures in the continuous segment, and an ontological (supervisory) controller monitors failures in the discrete segment. Moreover, the fuzzy automaton can provide information for the supervisory controller to attempt a recovery from faults originated from either the continuous or the discrete segment.

Execution Monitoring Principle

In many complex real-time industrial applications such as chemical, pulp and paper, or marine vessels, the control system is made up of a large number of smaller control units (e.g., Programmable Logic Controllers-PLCs) integrated into an overall control architecture. When controllers act in a sequential fashion, the output of one controller may be among the input signals of another controller. It is often recognized that these types of systems are complex due to the size, and the number of the possible state combinations in the total state space. However, when controllers of different makes and heterogeneous types are connected together, even the knowledge about the total state set may not be sufficient for a correct supervision. There are always assumptions, often undocumented, about the conditions under which a controller algorithm can be used such that the intended control goals will be reached (Fodor 1997).

In the case of high-speed networked control architectures, the types of information that can be exchanged between control units have been extended to state traces obtained in real-time, process models, and real-time diagnostic information. However, only a few research results have been reported so far that take advantage of the additional information transmitted among control units.

Industrial-strength complex control systems are required to act consistently relative to the initial goals when meeting unexpected situations in their environment. The capacity of a system to identify and recover from an error after meeting an unexpected situation is regarded in industry as a very important property. However, a recovery mechanism will be of practical interest only if it is designed application-independent, (i.e. it can be implemented, say, in the operating system of a PLC rather than being re-designed for each application). That means the recovery mechanism cannot be based upon a model of the environment. Moreover, the error detection level is required to be application-independent as well.

The research reported here proposes a solution to the problem of extending the safety and recovery capacity of complex control systems by introducing a new type of execution monitoring. The solution employs the theories of the Hybrid Fuzzy-Boolean Finite State Machine (HFB-FSM) (Grantner, Fodor, Driankov 1998), and Ontological Control (Fodor 1997). The main points of the approach are as follows:

- It has been shown in the theory of ontological control that problematic control situations at the reactive level (referred to as *state de-synchronization*) can be formally represented and classified. Moreover, it has also been shown that causes for state de-synchronization can be identified up to the so called *violations of the ontological assumptions* only when the state set of the reactive controller has certain syntactical properties (the state set is referred to as *well determined*).
- When the state set is well determined, a recovery operation is possible within certain constraints. The constraints are given in terms of an event-driven dynamic linguistic model implemented by the HFB-FSM. These boundaries can be used to specify the recovery capacity of a control system with a given set of actuator and sensor equipment.
- It has been shown that the recovery operation cannot be performed using the state of the reactive-level controller that needs to be recovered. Thus the method exploits a connected supervisory controller for the recovery operation.
- The state boundaries for the HFB-FSM are devised using the continuous model of the system

When an unexpected change occurs in the environment of a controller A, an execution monitoring unit which is connected to A can detect that by using the theory of ontological control. The execution-monitoring unit will then invoke the fuzzy specification of the discrete states that are involved in the erroneous situation. That includes a set of fuzzy states of the HFB-FSM, and an algorithm for triggering transients of fuzzy states. Then the execution monitoring unit uses the information on the next fuzzy state to determine if a recovery is possible, that is, the HFB-FSM represents the specification of the bounds within the recovery is possible. If the HFB-FSM enters a suitable fuzzy state, it returns the particular control action that will achieve the recovery of controller A. The following two sections present the two main theoretical tools involved, and then the method is illustrated by an example.

HFB-FSM Model

The HFB-FSM is an extension of the former Fuzzy-State-Fuzzy-Output Finite State Machine (FSFO FSM) model (Grantner, Patyra 1994). It is implemented by a Boolean automaton based upon two-valued logic and 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 ∘ 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.

$$
\begin{aligned}
&Z_F = X_F \circ R^* \\
&R^* = G(R_S) \\
&z_c = DF(Z_F) \\
&U_B = f_u(y_B) \\
&X_B = B(X_F) \\
&Z_B = B(Z_F) \\
&Y_B = f_y(X_B, W_B, Z_B, y_B)
\end{aligned}
\qquad (1)
$$

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

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

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 FSM 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 early concept of a fuzzy FSM based on a Boolean FSM and the State Membership Functions, in this context, have been introduced in (Grantner, Patyra, Stachowicz 1995). A formal representation was given in (Grantner 1994).

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

$$
\begin{aligned}
R_k^* = \max[&\min(\beta_1^k, R_{S_1}), \min(\beta_2^k, R_{S_2}), \ldots, \\
&\ldots, \min(\beta_k^k, R_{S_k}), \ldots, \min(\beta_p^k, R_{S_p})]
\end{aligned}
\qquad (3)
$$

where $\beta_1^k, \beta_2^k, \ldots, \beta_p^k$ stand for the degrees of state membership function g_{S_k}, and $R_{S_1}, R_{S_2}, \ldots, R_{S_p}$ are the overall rules in crisp states $S_1, S_2, \ldots, S_p$, 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 transition between active composite linguistic models is 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 (Grantner 1994). In this domain, those changes are joined by the state changes of the two-valued inputs. On the basis of this combined Boolean input/output sequence specification the crisp automaton section of the HFB-FSM will then be synthesized. 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.

A hardware accelerator of pipeline architecture for high-speed applications was presented in (grantner, Patyra 1994) for the earlier fuzzy automata model. It can be extended to adopt the new features of the HFB-FSM.

The theory of HFB FSM will be used in the sequel to model the changes in the control algorithm of a low-level reactive system recovering from a violation of the ontological assumptions (VOA).

Ontological De-synchronization

When a control system such as an autonomous agent is designed, the modeling assumptions for its control algorithm are inherently extended by additional assumptions about the complex environment in which the control will take place. These assumptions are not

represented by formal means, hence, an agent cannot verify whether they are true. Ontological control investigates the case when these assumptions are violated, situations in which a controller acts under *violations of the ontological assumptions.*

The architecture of an Ontological Controller (OC) capable to detect violations of ontological assumptions (VOA) in the context of a de-synchronization from the execution of a goal path was shown in (Fodor 1995). However, the OC is unable to recover from a VOA by itself (Fodor 1995). The concept of a state is defined in the OC by (4):

$$S_i = (y_i, u_{i,j}) \; (i,j=1,...,n) \qquad (4)$$

where y_i stands for a two-valued Boolean formula (referred to as a *plant formula*) showing the condition that is true at a given time in the controlled plant. Each relevant plant situation has a corresponding plant formula in some state. A control action denoted as $u_{i,j}$ is executed when y_i is true. The expected outcome of this action is that the plant changes such that at the next time instance y_j will be true. However, if some external action (disturbance) occurs, the expected change in the plant does not take place but a new plant state, y_k, will materialize instead. The disturbance is considered as an external action and, if known in advance, is denoted as $u_{i,k}^{ext}$ where the subscripts refer to the respective two plant formulas before and after the external action. The new state can be denoted as some $S_k = (y_k, u_{k,l})$. The control will then proceed in a succession of states. The states that can materialize from an arbitrary state S_i by external actions (disturbances) are referred to as *collateral states* to S_i and the set of such states is denoted as $K_{\_}(S_i)$.

An example is given below to illustrate the problem of ontological de-synchronization and the communication between control units. A control system is used to control the gap between two rolling mills for a metallurgic application. The system consists of two controllers, A and B. Controller A gives the setpoint for the gap and makes sure that all conditions are satisfied for Controller B that performs the control algorithm. Controller B, when enabled by A, controls the size of the gap such that it measures the position of the actuator and performs corrective actions to bring the gap size to the setpoint value. The control action is implemented by a pneumatic device that moves the rolls, and thus modifies the gap. The relevant parts of the state set and data exchanged between regulators is shown in Figure 1.

In state S_0, an electric motor is started which builds up the air pressure. In state S_1, a pump is started. It is assumed that at the nominal speed of the electric motor, the air pressure is large enough to start a pressure controller in state S_2, and state S_3 will materialize, in which the nominal value of the pressure is maintained. In state S_3, if the pressure decreases due to a high air load, corrective actions are taken such as cooling the pump. A violation of the ontological assumptions (VOA) occurs at a state $S_i = (y_j, u_{j,k})$ if $u_{i,j}$ is executed in state S_i, but the expected y_j does not materialize in the plant, although no external action has occurred. It has been shown in (Fodor 1995) that a VOA manifests always as a state transition from S_i to a state in $K_{\_}(S_j)$ where S_j is the consecutive (next expected) state to S_i. This type of transition is referred to as an ontological desynchronization. It has also been shown in (Fodor 1995) that for certain state sets, a VOA has the effect such that the controller enters a cycle. This cycle repeats indefinitely such that the ontological de-synchronization will be repeated as well.

For the system in Figure 1, let us consider the following ontological violation: the pump has a leak such that at nominal engine speed in state S_2 the air pressure stays at a level lower than the nominal one. That means a state transition from S_2 will occur directly to state S_4, actually bypassing the intermediate state S_3. However, since the cause for the current situation is not the high temperature of the motor, states S_4, S_5 and S_2 are passed quickly and the loop continues indefinitely by a new jump from state S_2 to S_4, without any apparent error is being detected. The recovery operation can be performed if two conditions are met:

(i) The cause for the loop is recognized. That is done by the theory of ontological control.

(ii) Overload limits can be specified for the engine such that the condition for nominal air pressure is satisfied. That is, the boundaries of a state can be extended. That is done by using the theory of the HFB-FSM.

The recovery from this situation requires the determination of a state that corresponds to the current plant situation, and it has a control action that can "break" the cycle. For fuzzy controllers, there can be found recovery solutions by adding extra rules to the rule base (Driankov, Fodor 1996). However, for controllers such as PLCs that have discrete states, the problem of recovery becomes more difficult since there is no state with acceptable properties in the state space of the controller that can materialize at a VOA. Thus the recovery algorithm suggested in this paper relies on techniques that can accommodate fuzzy states and yet produce two-valued outputs.

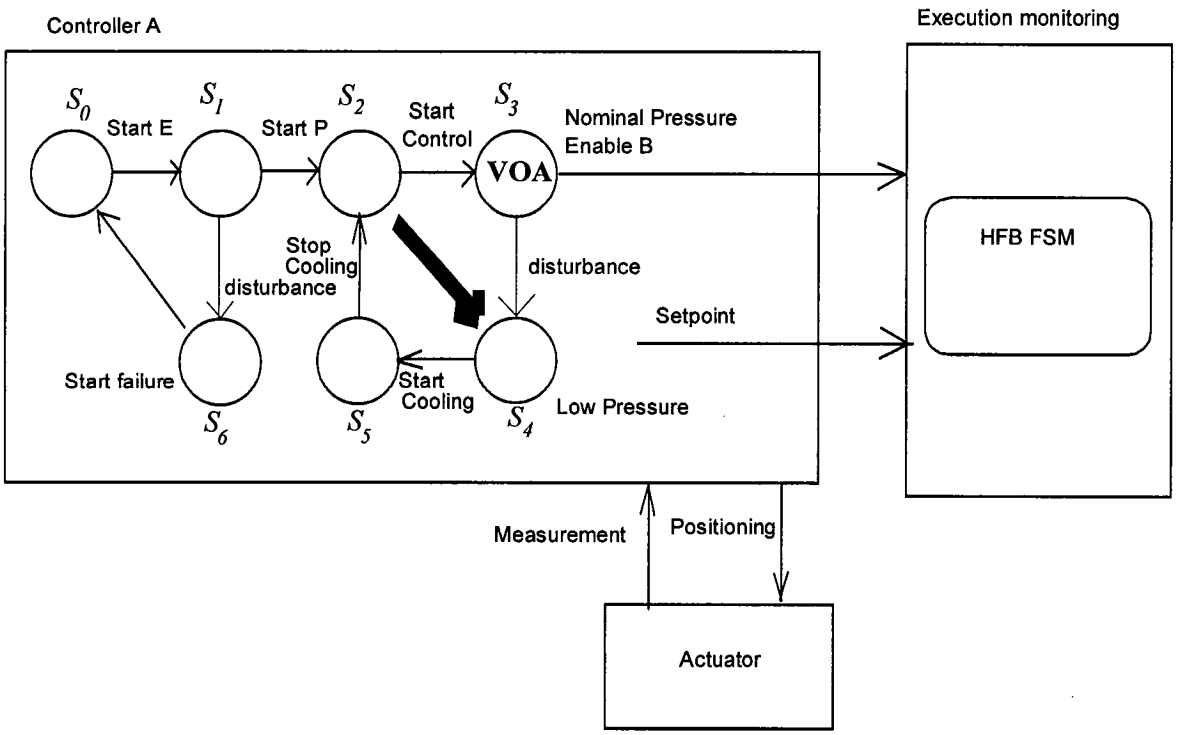

Figure 1 - Example of Control System

Recovery Using HFB-FSM

The use of the HFB-FSM will be outlined here for a single state transition at a VOA only. At design time, each plant formula is associated with linguistic intervals. In general, after an ontological de-synchronization is detected, the reactive controller provides the HFB-FSM with the following information:

(i) the relevant fuzzy input set X_F to the FSFO FSM that includes the fuzzified plant formulas of y_i (the last state materialized as expected), y_j (the state with the VOA), and the fuzzified plant formulas of the states in $K\_(S_j)$,

(ii) the fuzzy output set Z_F that consists of the fuzzified control actions of the expected next state, and in addition, the fuzzified plant formulas of the collateral states of the expected next state,

(iii) the set of two-valued inputs (if any)

(iv) state membership function degrees that are set such that the expected next state (dominant state) is assigned the degree of 1,

(v) the actual mapping scheme between fuzzy and Boolean subintervals of the B algorithm,

(vi) the relevant section of the fuzzy state transition graph along with conditions, in order to devise the next state of the HFB-FSM.

Due to a VOA, a fuzzified plant formula *in-between* the expected y_j (leading to a state of the HFB FSM that corresponds to state S_3) and the fuzzified plant formulas of the states in $K\_(S_j)$ will be passed to the HFB FSM. The situation will be dealt with as a trigger condition for the next fuzzy state of the HFB-FSM. The conditions for state transients are specified in terms of membership functions. If the received fuzzified plant formula is *not too far* from the one for y_j (i.e. despite the leak, the nominal air pressure is still in the work range of the pump) then the HFB-FSM will enter to a state corresponding to S_3. A two-valued control action signal asserted in this state will cause the system to recover from the VOA. If the condition above is not met, the HFB FSM will move to a state corresponding to one of the states in $K\_(S_j)$. That means the controller cannot recover from the VOA, hence it should be stopped.

The signals representing intervals in the plant formulas for the discrete state sets of the PLCs are of, in fact, two-valued. Fuzzification makes the boundaries between states continuous such that a state preserves its properties beyond its two-valued limits. Hence, fuzzification "balances" the fuzzy boundaries of the states against the degree of the ontological violation and, if it is possible, helps to choose a control action to recover from a VOA.

Conclusions

Current research shows that by using a fuzzy state machine along with an ontological controller in a proper architecture, the error detection and recovery capability of a hybrid system can be substantially improved. The detection architecture and algorithm is application independent. The method provides a uniform formalism for execution-monitoring of hybrid systems.

A further research problem is to develop an improved mapping method between the HFB-FSM model and the continuous/discrete models of the system, and imposing timing constraints on the changes of the linguistic models.

Acknowledgement

This research was supported by the Michigan Space Grant Consortium (Grant #: 9633401), Western Michigan University, ABB Industrial Systems AB, Sweden, and The Swedish Research Council for Engineering Sciences (271/96-134).

References

Cellier, F.E., H. Elmqvist, M. Otter, and J.H. Taylor 1993. Guidelines for Modeling and Simulation of Hybrid Systems, in Procedings of the 1993 IFAC World Congress, Sydney, Australia, Vol. 8, pp. 391-397.

G. A. Fodor 1997. *Ontologically Controlled Autonomous Systems: Principles, Operations and Architecture.* Kluwer Academic Publishers, Boston/Dordrecht/London.

J.L. Grantner, G. Fodor, Dimiter Driankov 1998. Hybrid Fuzzy-Boolean Automata for Ontological Controllers. In Proceedings of the 1998 World Congress for Computational Intelligence, WCCI'98, Anchorage, Alaska, USA, Vol. I, pp. 400-404.

D. Driankov, G. Fodor, 1996. Fuzzy control under violations of ontological assumptions, invited plenary talk. In Proceedings of the FLAMOC'96 Conference, Sydney, Australia, pp. 109-115.

G. Fodor, 1995. Ontological Control: Description, Identification and Recovery from Problematic Control Situations. PhD diss., Dept.of Computer Science, University of Linkoeping, Sweden

J. Grantner, M. Patyra, 1994. Synthesis and analysis of fuzzy logic finite state machine models. In proceedings of the FUZZ-IEEE'94/WCCI'94 Conference, , Orlando, FL, Vol. I, pp. 205-210.

J. Grantner, M. Patyra, M. Stachowicz, 1995. Intelligent fuzzy controller for event-driven real-time systems and its VLSI implementation. In the book *Fuzzy Control Systems* (Eds. A. Kandel, G. Langholz), CRC Press, Boca Raton, FL, USA, pp. 161-179.

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