A Framework for Hybrid Control Design and its Relations with a Class of Fuzzy Logic Control Systems

R. Fierro, F. L. Lewis, and K. Liu
Automation and Robotics Research Institute. The University of Texas at Arlington
7300 Jack Newell Blvd. South. Fort Worth, Texas 76118-7115
rfierro, flewis, kliu@arri.uta.edu

***Abstract*— This paper presents a Hybrid System Framework which considers simultaneously the lower-level control and decision-making issues. This reconfigurable framework can accommodate a wide range of situations, from aircraft control systems to mobile manipulators. A continuous-state plant is supervised by a discrete-event system which is based on a theory of *linked* finite state machines. The composite system is viewed as an iterative process where a task is carried out by changing the structure of the continuous-state plant. An application of this framework is illustrated through a mobile manipulator example. Finally, some connections between hybrid systems and a class of fuzzy logic control systems are established.**

I. INTRODUCTION

In intelligent control of complex systems, one is faced with the problem of providing stability, performance, and robustness at the close-loop real-time level, as well as decision-making control with guaranteed performance at the supervision level. Unfortunately, it is difficult to formally derive a finite state representation of a closed-loop continuous-state dynamical system that is suitable for upper-level decision-making functions with close-loop stability of the entire system. Any discrete-event representation of dynamical systems must encapsulate the relevant *behaviors* and *events* needed for effective decision-making control.

Hybrid systems have attracted the attention of the control community in the last few years [1]-[3], [9], [12]. The plant is often described in terms of differential/difference equations and has the continuous-state conventional in system and control theory. In an effort to impart more intelligence and decision-making capabilities to the system, it may be desired for a rule-based discrete-event controller to determine the control and performance objectives of the real-time plant. This requires an *interface* to convert continuous-time signals to discrete-event symbols and vice versa. When viewed from above, this interface, together with the continuous-time plant, becomes a *Discrete-Event Plant* modeled by a finite automaton.

To define the states of the Discrete-Event Plant, the 'events' that cause changes of state must be defined. One way of accomplishing this is to partition the continuous-state space into discrete regions by various methods. In the work of Antsaklis [1], the continuous-state space is partitioned into regions using 'event boundary functions', and a nice theory is developed that includes notions of reachability, etc. Another possibility is to associate discrete states with different plant outputs, controllers, and/or reference trajectories, so that changing discrete states corresponds to changing the plant performance and control objectives.

This paper considers a class of hybrid dynamical system where a given task is executed as a sequence of closed-loop continuous plant behaviors. That is, a discrete-event controller selects a continuous-state controller, a reference trajectory, and an output function from a library of plant control algorithms, input functions, and output functions, respectively. The selected plant behavior is kept active for a certain time until a prescribed plant event or controller event occurs.

This class of decision-making controllers is common, for instance in modern aircraft controls, mobile robots, Intelligent Vehicle Highway Systems (IVHS) [8], and Flexible Manufacturing Systems [10].

II. PRELIMINARIES

A. Notation

We collect some notation and abbreviations used in this work: continuous-state plant (CSP), discrete-event (DE), discrete-event plant (DEP), continuous-state controller (CSC), discrete-event controller (DEC), hybrid dynamical system (HDS), finite state machine or finite automaton (FSM), Intelligent Control (IC), Flexible Manufacturing System (FMS), and Petri nets (PN).

Throughout, $\Re$, $\Re^+$, $\mathbb{Z}$, $\mathbb{N}$, denote the reals, nonnegative reals, i.e., $\Re^+ = [0,+\infty)$, integers, and nonnegative integers, respectively. $\underline{N}$ denotes the set

{1, 2, ..., N}. Let $\Re^n$ and $\Re^{n\times m}$ denote the real n-space and the set of $n\times m$ real matrices, respectively. If $A \in \Re^{n\times m}$, then A^T denotes the transpose of A.

Let Ω be a subset of $\Re^n$. $\overline{\Omega}$ represents the closure of Ω, Ω^o its interior, and $\partial\Omega$ its boundary.

B. Hybrid System Model

A class of HDS, commonly found in the literature, considers the 3-layer structure depicted in Fig. 1: a CSP described by differential/difference equations, a DEC, and an interface. The interface maps signals from the continuous-state domain S to symbols in the discrete-state domain Σ and vice versa. The CSP plus the interface are viewed as a DEP, which is controlled by the DEC.

The DEC is a five-tuple $(Q, \mathcal{C}, \mathcal{R}, \phi, \gamma)$, consisting of the finite state set, input set, output set, state transition function and output function, respectively. The DEC is described by the equation

$$\textit{DEC:} \quad \begin{aligned} q(k+1) &= \phi(q(k), C(k)), \\ R(k) &= \gamma(q(k), C(k)), \end{aligned} \tag{1}$$

where $q(k) \in Q$ is the state after the k^{th}-event, $C \in \mathcal{C}$, $R \in \mathcal{R}$, ϕ: $Q\times\mathcal{C}\to Q$, γ: $Q\times\mathcal{C}\to\mathcal{R}$, $k(t) \in \mathbb{N}$.

The DEP is a five-tuple $\Sigma = (\mathcal{X}, \Lambda, \mathcal{C}, \psi, \varphi)$, consisting of the finite state set, input set, output set, state transition function and output function, respectively. The DEP equation is

$$\textit{DEP:} \quad \begin{aligned} \xi(k+1) &= \psi(\xi(k), R(k), z(k)), \\ C(k) &= \varphi(\xi(k), R(k), z(k)), \end{aligned} \tag{2}$$

where $\xi \in \mathcal{X}$, $\Lambda \subseteq \mathcal{R}\times Z$, $z \in Z \subset \Re^h$, ψ: $\mathcal{X}\times\mathcal{R}\times Z\to\mathcal{X}$, φ: $\mathcal{X}\times\mathcal{R}\times Z \to\mathcal{C}$. $z(k)$ contains the event information from the CSP i.e., a signal that enables the state transition function in the DEP. The interface equations are

$$\textit{Interface } (\Sigma/S): \quad \begin{aligned} r(t) &= \alpha(\xi(k), R(k)), \\ i(t) &= \beta(\xi(k), R(k)). \end{aligned} \tag{3}$$

$$\textit{Interface } (S/\Sigma): \quad z(k) = \nu(y(t), r(t)), \tag{4}$$

where $r \in \Re^p$ is the reference trajectory, $i \in \underline{N}$ is a deterministic scalar index, $y \in Y \subset \Re^p$. Functions α: $\mathcal{X}\times\mathcal{R}\to\Re^p$ and β: $\mathcal{X}\times\mathcal{R}\to\underline{N}$ are mappings in Σ/S, and function ν: $Y\times\Re^p \to Z$ is a mapping in S/Σ.

The CSP is a five-tuple $S = (X, U, Y, f, h)$, consisting of the state set, input set, output set, state transition function and output function, respectively. The CSP and CSC equations are

$$\textit{CSP:} \quad \begin{aligned} \dot{x}(t) &= f(x(t), u(t)), \\ y(t) &= h(x(t), i(t)). \end{aligned} \tag{5}$$

$$\textit{CSC:} \quad u(t) = \Bbbk(x(t), y(t), r(t), i(t)), \tag{6}$$

where $x \in X \subset \Re^n$, $u \in U \subset \Re^m$, f: $X\times U \to\Re^n$ is assumed to be Lipschitz continuous, h: $X\times I \to Y$, $\Bbbk$: $X\times Y\times\Re^p\times I \to U$. For simplicity we assume that the continuous-state $x(t)$ can be measured. If this is not the case an *observer* may be required to estimate the continuous-state. We will not pursue this.

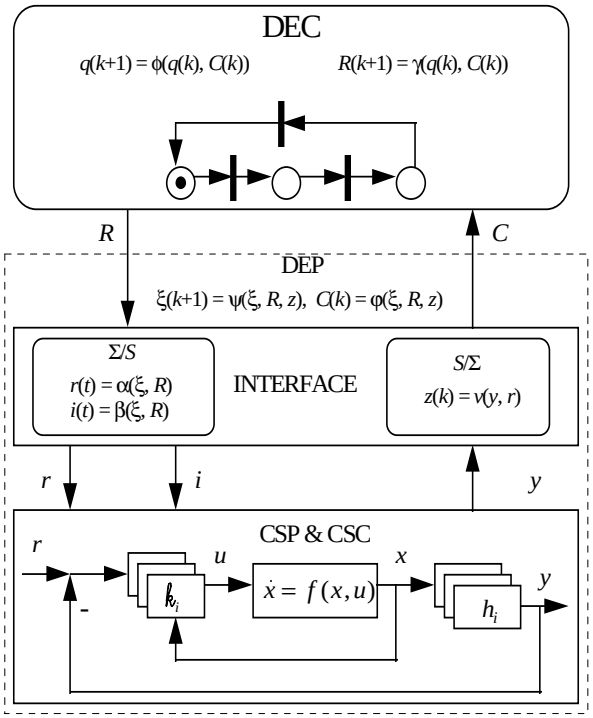

Fig. 1. Hybrid dynamical system.

C. Definitions of Operational Level Behavior Elements for the Plant

Some notions of behaviors and events are needed in any theory of hybrid systems. Given a plant (e.g., aircraft, mobile robot), with a set of outputs Y (e.g., airspeed, altitude, pitch rate) a set of control inputs U (e.g., aileron, elevator, rudder, throttle), we define a *closed-loop behavior* of the plant as $(u, y, \Bbbk, r)$, where $u(t) \in U$ and $y(t) \in Y$ are vectors of inputs and outputs, $\Bbbk \in K$ is a library of stabilizing tracking controllers (adaptive, neural net, PID, etc.), and $r(t)$ is the reference trajectory. The controller $\Bbbk$ should be designed so that the closed-loop system is stable with suitable robustness and performance properties.

In an aircraft, sample behaviors are 'altitude hold mode', 'pitch rate command following mode', 'glide slope coupler mode', etc. In a mobile robot the behaviors include 'wall following', 'obstacle avoidance', etc. It is seen that by suitable selection of the controller, plant inputs, plant outputs, and reference trajectories the full range of *natural* closed-loop

behaviors of any given plant may be obtained. This is defined as the *library of closed-loop behaviors*.

The *events* are next defined as occurrences of interest to the next higher-level controller. Events may be formally defined using 'event boundary functions' in the continuous-state space of the plant (c.f., work by Antsaklis [1]). We define the behaviors to include the plant plus controller— that is, we are discussing closed-loop behaviors. Thus, the controller guarantees stability of the plant and suitable performance of the behaviors. In Fig. 1 R is a command from a selected language for the closed-loop system to implement a specific choice of (u_i, y_i, k_i, r_i), z is a symbol indicating event occurrence. A DEC or supervisor monitors C and provides R. The closed-loop plant plus interface in Fig. 1 can represent, for instance a *workcell* in a FMS, an aircraft, a mobile robot, etc.

D. The State Discretization Problem and Homomorphism Theory

Referring to Fig. 1, the fundamental issue in hybrid-state systems is that the plant evolution equation and the evolution description of the DEP are over two different algebras. The *State Discretization* problem addresses the following issue: given a continuous-state plant description (either continuous-time or discrete-time) determine a DEP description over a specified algebra. The state discretization problem is akin to the *time* discretization problem in control systems, where a discrete-time plant includes a *ZOH* mechanism.

Given a set of discrete states $\mathcal{X}$, with a set of specified operations, and an input alphabet Λ, define an operator ('generator') F: $X \to \mathcal{X}$ and an operator ('inverse actuator') G: $U \to \Lambda$. It is now desired to determine a finite state description $(\mathcal{X}, \Lambda, \psi)$ such that

$$\psi = F \circ f \circ G^{-1}, \tag{7}$$

$$\psi(F(x), G(u)) = F(f(x,u)), \tag{8}$$

for every $x \in X$, $u \in U$. It is now clearly seen that the problem resides precisely in defining the DE state evolution function ψ: $\mathcal{X} \times \Lambda \to \mathcal{X}$. This is intimately related to the theory of *homomorphisms*. If the DEP has a rule-base transition function, then $\mathcal{X}$ is the set of logical state vectors, with operations of (AND, OR), and we are dealing with semiring homomorphisms. That is, the underlying group structure is defined by the selection of X, U and the operations of interest.

This framework seems to provide a scheme of attack that includes as well the time discretization problem, where $X = \mathcal{X} = \Re^n$ and F is the sampling operator

$$F(x) = \sum_k x(k\tau)\delta(t - k\tau), \tag{9}$$

where $\delta(\cdot)$ is the Kronecker delta and τ is the sampling time. Moreover, the widely used *fuzzy logic control* scheme is related to this framework in the sense that F, G^{-1}, and ψ denote fuzzification, defuzzification and inference map, respectively. Some connections between hybrid control systems and a class of fuzzy logic control systems are established in Section IV.

III. A HYBRID CONTROL DESIGN EXAMPLE

In this section, we apply the hybrid framework given by equations (1)-(6) to a mobile manipulator performing a task in a manufacturing plant. The mobile manipulator has to pick and move an object from the conveyor belt in position a to position d. It releases the object in position d and repeats the task. When the mobile base goes from a to d (right), the DEC selects the controller k_1 which considers the mass of the object and drives the mobile base with linear velocity $v_1(t)$. On the other hand, if the mobile manipulator goes from d to a (left), the DEC selects the controller k_2 which drives the mobile base faster i.e., $v_2(t) > v_1(t)$. When the mobile base reaches a or d, the DEC selects the controller k_3 to drive the onboard arm to pick/release the object. This example is an extension of a case study presented in [5].

The design method consists of five steps: (1) find an abstract model representing the continuous-state system and define the 'important' events in the CSP, (2) define the DEP, (3) design the DEC, (4) design the interface between the DEP and the CSP, and (5) verify the performance of the entire hybrid system. If the performance is not satisfactory, then return to steps (1), (2), (3) or (4). Details are discovered in [6].

In this example, we describe the position of the mobile base by a first order differential equation given by

$$\begin{aligned} \dot{x} &= \lambda_i(-x + u), \qquad i = 1,2 \\ y &= x. \end{aligned} \tag{10}$$

If the mobile base goes to the right the controller k_1 is used and $\lambda_1 = 2.5$ (s^{-1}). On the other hand, if the mobile base goes to the left the controller k_2 is used and $\lambda_2 = 5$ (s^{-1}). The time signal $u(t)$ represents the desired position of the mobile base. We neglect the dynamics of the manipulator, assuming that the time for picking and releasing the object is very short.

The DEP is a qualitative representation of the system with respect to a given task. It should give the required information to the DEC in order to accomplish the task successfully. The continuous-state space $x(t)$ is partitioned in a number of regions and a plant event is defined for each region. In this example, the DEP states

are given by the mobile base position $\xi_1(k) = \{a, b, c, d\}$, and the end-effector action $\xi_2(k) = \{0 \equiv$ pick_object, $1 \equiv$ release_object$\}$. The DEP inputs (i.e., commands) are given by the position command $U_1(k) = \{0 \equiv$ right, $1 \equiv$ left$\}$ and the onboard arm command $U_2(k) = \{0 \equiv$ hold, $1 \equiv$ release$\}$.

Whenever the mobile base crosses certain boundaries an event is generated and the *event function* $k(t)$ is increased by one. The DEC supervises the performance of the DEP and chooses the inputs to perform a given task. The DEC state is given by $q_1(k) = \{0 \equiv$ to_right, $1 \equiv$ to_left$\}$, and the DEC input, $U_{C1}(k)$, is the mobile base position.

The closed-loop DE system is defined as follows

$$U_{C1}(k) = C_1(k),\ U_1(k) = R_1(k), \text{ and } U_2(k) = R_2(k), \quad (11)$$

and the dynamics of the closed-loop DE system become

$$\begin{aligned} & q_1(k+1) = \phi(q_1(k), \xi_1(k)), \\ DEC: \quad & R_1(k) = q_1(k), \\ & R_2(k) = \gamma(q_1(k), \xi_1(k)), \end{aligned} \quad (12)$$

$$\begin{aligned} & \xi_1(k+1) = \psi_1(\xi_1(k), q_1(k)), \\ & \xi_2(k+1) = \gamma(q_1(k), \xi_1(k)), \\ DEP: \quad & C_1(k) = \xi_1(k), \\ & C_2(k) = \xi_2(k). \end{aligned} \quad (13)$$

To design the interface (S/Σ, Σ/S): A real number is assigned to each symbolic desired position of the mobile base, and an integer in the set $\{1, 2\}$ is assigned to the symbolic controller selection $i(t)$ as follows

$$\begin{aligned} u(t) &= \alpha(\xi_1(k), R_1(k)) \\ &= \alpha(\psi_1(\xi_1(k), R_1(k)), \end{aligned} \quad (14)$$

$$u = \begin{cases} 0 & \text{if } \psi_1(\cdot) = a \\ 2 & \text{if } \psi_1(\cdot) = b \\ 4 & \text{if } \psi_1(\cdot) = c \\ 6 & \text{if } \psi_1(\cdot) = d \end{cases},$$

$$i(t) = \beta(\xi_1(k), R_1(k)),\ i = \begin{cases} 1 \text{ if } & R_1(k) = 0 \\ 2 \text{ if } & R_1(k) = 1 \end{cases}. \quad (15)$$

Let ε be a small positive constant. A plant event is generated whenever the actual position of the mobile base is near a predefined position in the set $\{a, b, c, d\}$, more formally

$$z(k) = v(y, u),\ z(k) = \begin{cases} 1 \text{ if } |u(t) - y(t)| < \varepsilon \\ 0 \qquad \text{therwise.} \end{cases} \quad (16)$$

The DEP representation of the CSP does not change; however, $z(k)$ *enables* the state transition function and *increases* the event function $k(t)$ as follows

$$\xi_1(k+1) = \psi_2(\xi_1(k), q_1(k), z(k)), \quad (17)$$

$$\psi_2(\cdot) = \begin{cases} \xi_1(k) & \text{if } z(k) = 0 \\ \psi_1(\cdot) & \text{if } z(k) = 1 \end{cases}, \quad (18)$$

$$k(t) = \begin{cases} k+1 & \text{if } z(k) = 1 \\ k & \text{if } z(k) = 0 \end{cases}. \quad (19)$$

Note that only when a plant event is generated, i.e., $z(k) = 1$, the discrete-state representing the robot's position ξ_1 changes to a new state. Fig. 2 depicts the closed-loop HDS.

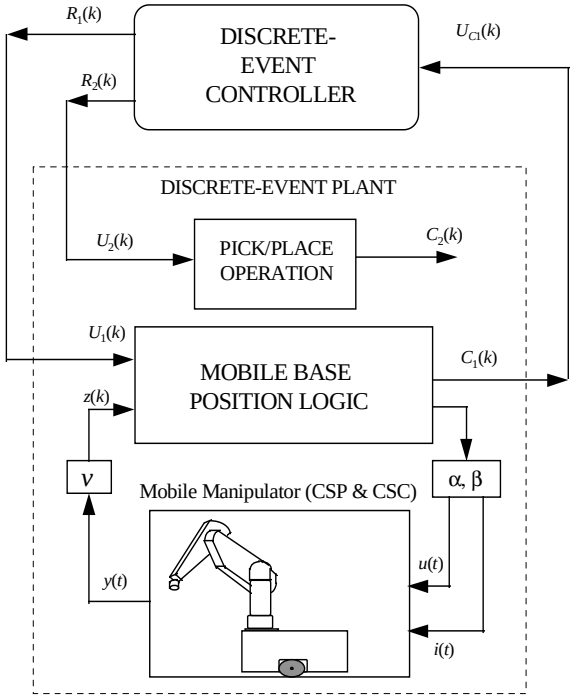

Fig. 2. The closed-loop HDS.

IV. FUZZY LOGIC CONTROL SYSTEMS VIEWED AS HYBRID CONTROL SYSTEMS

A fuzzy control system is a special class of hybrid dynamical systems. In fuzzy systems a finite rule base interacts with a continuous-state plant. Furthermore the communication between the fuzzy controller and the controlled plant is by using the so-called *fuzzifier* and *defuzzifier* interfaces. We can view a fuzzy logic control system as a hybrid system as follows: (1) the DEC of the hybrid system is replaced by a fuzzy logic controller, (2) the event *generator* (S/Σ) and *actuator* (Σ/S) of the hybrid system interface (Section II, Fig. 1) are replaced by a fuzzifier and a defuzzifier, respectively, and (3) in both hybrid and fuzzy structures the continuous-state plant is modeled by ordinary differential/difference equations.

Consider a single-input, single-output dynamic system of the form

$$\dot{x}(t) = f(x, u), \quad (20)$$

where $x \in \Re^n$ is the state variable, $u(t) \in \Re$ is the system input, f: $\Re^n \times \Re \to \Re^n$ is a smooth mapping defined on an open set $\Phi \subset \Re^n \times \Re$. The hybrid control problem consists of linearizing the system (20) at various operating points and designing local controllers that satisfied certain performance criteria. The next step is to realize a DEC that switches between the local controllers when the continuous-state hits some predefined boundaries.

This hybrid control problem can be solved by using fuzzy logic techniques where systematic methods are available. A set of *fuzzy logic-based linear models* can be constructed as follows [4]. Define a fuzzy rule base of the form

$$R^{(i)}: \text{ IF } x_1 \text{ is } \mathbf{X}_1^{(i)} \& \cdots \& \ x_n \text{ is } \mathbf{X}_n^{(i)}, \quad \text{THEN } \dot{x} = A_i x + B_i u, \ i = 1,2,..,N \tag{21}$$

where $\mathbf{X}_k^{(i)}$, k = 1,2, ..., n, are the fuzzy numbers corresponding to state variables $x(t)$ for the i^{th} rule and A_i and B_i are matrices of appropriate dimensions. Let $\omega^{(i)}$ denote the truth value of the i^{th} rule, and $\mu_{\mathbf{X}}(\cdot)$ denote the membership function for $\mathbf{X}$. By using the product-inference rule, it yields

$$\omega^{(i)} = \prod_{j=1}^{n} \mu_{\mathbf{X}_j^{(i)}}(x_j). \tag{22}$$

The N fuzzy rules (21) partition the state space $X \in \Re^n$ into N regions Ω_i. It is assumed that $X = \bigcup_{i \in N} \Omega_i$. The partition Ω_i is defined by the set

$\{x \mid \omega^{(i)}(x) \geq \omega^{(j)}(x)$ with $i \neq j$, and $j = 1,2,...,N\}$,

and the boundary $\partial\Omega_{ij}$ (i.e., $\Omega_i \cap \Omega_j \neq \varnothing$)

$\{x \mid \omega^{(i)}(x) = \omega^{(j)}(x)$ with $i \neq j$, and $j = 1,2,...,N\}$.

Each linear model (21) can be considered as a local representation of the nonlinear system (20) in region Ω_i. Then, the following fuzzy logic rule defines what controller to connect to the actual nonlinear plant

$$R^{(i)}: \text{ IF } x_1 \text{ is } \mathbf{X}_1^{(i)} \& \cdots \& \ x_n \text{ is } \mathbf{X}_n^{(i)}, \quad \text{THEN } u_i = -K_i x, \ i = 1,2,..,N. \tag{23}$$

Equation (23) provides a *local* control law. The gain $K_i \in \Re^{1 \times n}$ is a local state-feedback controller for each fuzzy logic-based linear model. $K_i(x)$ can be designed by using any technique (e.g., LQR, pole placement, H_∞).

A. Fuzzy Logic Interface

In the literature it is common to assume that the dynamics of the system (20) is fully known, see for instance [11], which is a major simplification. In this case the discretization of the continuous-state space, and the local matrices A_i, B_i and K_i can be computed *off-line*. In most applications, the dynamics of the system is partially known or very difficult to determine. Therefore we need a sort of *on-line* identifier to learn the partitions of the continuous-state space and estimate the local matrices $\hat{A}_i, \hat{B}_i$. This identifier has been proposed in [4], details are omitted here. In this section, we should like to use this fuzzy logic approximator as a tool to design the hybrid system interface. We also provide a preliminary framework for the stability analysis of a class of hybrid systems.

We assume that system (20) can be approximated by the set of fuzzy logic-based linear models

$$\dot{\hat{x}} = \hat{A}_i \hat{x} + \hat{B}_i u, \quad i = 1,2,...,N. \tag{24}$$

Each linear model (24) can be considered as a local representation of the nonlinear system (20) in region Ω_i. When the local control laws (consequent of (23)) have been determined, one can apply at least two control schemes:

1. A hybrid (mode switching) control law. The local closed-loop system is given by

$$\dot{\hat{x}} = [\hat{A}_i - \hat{B}_i K_i]\hat{x} \equiv \hat{H}_i \hat{x}, \tag{25}$$

where $\hat{x} \in \Omega_i$ and $i = 1,2,...,N$. In this case the control actions are discontinuous. The local controllers may be designed such that the closed-loop systems satisfy some desired properties. Note that system (25) is compatible with the hybrid model introduced in Section II, but in this case the DEC and interface are realized by using fuzzy logic techniques.

2. A smooth fuzzy control law with state-feedback control signals generated by the centroid defuzzification technique. In this case the control actions are interpolated as the state of the system is transferred from region Ω_i to Ω_j.

The work presented herein is concerned with the hybrid (mode switching) control system (25). The smooth fuzzy control law corresponds to a standard fuzzy logic approach which has been studied extensively by a number of authors. See for instance [11] and the references therein.

B. Stability Considerations

In order to guarantee the stability of system (25), we need to impose some restrictions on the system dynamics. To be specific, we shall consider that

i) The nonlinear system dynamics vary slowly, i.e., the evolution of the system can be represented by piecewise constant linear systems of the form

$$\dot{\hat{x}}(t) = \hat{H}_{i_k} \hat{x}(t), \text{ on } t \in [t_k, t_{k+1}) \text{ and } k = 0,1,2,.... \tag{26}$$

ii) There exists $t_{min} > 0$ such that $t_{min} \leq t_{k+1} - t_k < \infty$,

and iii) $\left\| \hat{H}_{i_{k+1}} - \hat{H}_{i_k} \right\| < \varepsilon \ \ \forall k$.

These assumptions are well-known for the stability analysis of a class of gain scheduling systems [13], and adaptive systems [14]. In the following proposition we

prove the stability of the hybrid system provided that the above conservative assumptions are satisfied. The key idea is to show that the same Lyapunov function $V_{i_k}(x) = x^T P_k x$ can be associated to $\hat{H}_{i_k}$ and $\hat{H}_{i_{k+1}}$.

Proposition If (i), (ii) and (iii) are satisfied the hybrid system (26) is asymptotically stable.

Proof Since $\hat{H}_{i_k}$ are stability matrices for all *k*. We can select, without loss of generality

$$\hat{H}_{i_k}^T P_k + P_k \hat{H}_{i_k} = -I. \qquad (27)$$

Consider the following Lyapunov function candidate

$$V_{i_{k+1}}(x) = x^T P_k x, \qquad (28)$$

differentiating there results

$$\dot{V}_{i_{k+1}}(x) = x^T (\Delta_k^T P_k + P_k \Delta_k - I) x, \qquad (29)$$

where

$$\Delta_k = \hat{H}_{i_{k+1}} - \hat{H}_{i_k}, \text{ and } \|\Delta_k\| < \varepsilon. \qquad (30)$$

If ε is small enough, say $\frac{1}{4\lambda_{\max}(P_k)}$. It can be shown that $\dot{V}_{i_{k+1}}(x) < 0$, then the hybrid system is asymptotically stable.

□

Remark We have shown that under certain conditions the approximate state $\hat{x}$ is asymptotically stable. We need to impose certain conditions on the robustness of the matrices $\hat{H}_i$ to guarantee the stability (not asymptotic stability) of the actual nonlinear system. A more elaborate stability analysis is beyond the scope of this paper.

V. CONCLUSIONS

We have presented a hybrid framework to model and analysis a class of IC systems where a real-time continuous-state plant is supervised by a discrete-event controller. The structure of the continuous-state plant, i.e., closed-loop behavior, changes asynchronously in response to a DEC command.

The behavior of the upper DE levels is modeled by linked FSM. It has been shown that the interconnections of linked FSM provide sufficient generality for IC purposes as they can accommodate shared resources, task choices, and conflicts.

Note that hybrid systems are similar to the widely used fuzzy logic systems. We believe hybrid systems provide a more general framework which can benefit from the results available in fuzzy theory.

REFERENCES

[1] P. J. Antsaklis, M. Lemmon, and J. A. Stiver, "Modeling and design of hybrid control systems," *Proc. IEEE Mediterranean Symp. New Directions in Control & Automation*, Krete, Greece, 1994, pp. 440-447.

[2] P. J. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, Eds., *Hybrid Systems II*, Lecture Notes in Computer Science, vol. 999, New York: Springer, 1995.

[3] M. S. Branicky, V. S. Borkar, and S. K. Mitter, "A unified framework for hybrid control," *Proc. IEEE Conf. Dec. Control*, Lake Buena Vista , FL, 1994, pp. 4228-4234.

[4] K. Liu, and F. L. Lewis, "Adaptive tuning of fuzzy logic identifier for unknown nonlinear systems," *Int. J. Adaptive Control and Signal Processing*, vol. 8, pp. 573-586, 1994.

[5] M. Dogruel and Ü. Özgüner, "Modeling and stability issues in hybrid systems," in [2], 1995, pp. 148-165.

[6] R. Fierro and F. L. Lewis, "A framework for hybrid control design," *IEEE Trans. on Syst., Man, Cyber*. To appear.

[7] R. Fierro, F. L. Lewis, and C. T. Abdallah, "Common, multiple and parametric Lyapunov functions for a class of hybrid dynamical systems," *Proc. IEEE Mediterranean Symp. New Directions in Control & Automation*, Krete, Greece, 1996, pp. 77-82.

[8] D. N. Godbole, J. Lygeros, and S. Sastry, "Hierarchical hybrid control: An IVHS case study," *Proc. IEEE Conf. Dec. Control*, Lake Buena Vista, FL, 1994, pp. 1592-1597.

[9] R. Grossman, A. Nerode, A. Ravn, and H. Rischel, Eds., *Hybrid Systems*, Lecture Notes in Computer Science, vol. 736, New York: Springer, 1993.

[10] F. L. Lewis, H-H. Huang, R. Fierro, and D. Tacconi, "Real-time task planning, resource allocation, and deadlock avoidance," *Proc. ISIC Workshop on Architectures for Semiotic Modeling and Situation Analysis in Large Complex Systems*, Monterey, CA, 1995, pp. 347-355.

[11] H. O. Wang, K. Tanaka, and M. F. Griffin, "An approach to fuzzy control of nonlinear systems: stability and design issues," *IEEE Trans. on Fuzzy Syst.*, vol. 4, no. 1, pp. 14-23, Feb. 1996.

[12] A. Puri and P. Varaiya, "Modeling and verification of hybrid systems," *Proc. American Control Conf.*, Seattle, 1995, pp. 4466-4470.

[13] J. Shamma, and M. Athans, "Guaranteed properties of gain scheduled control for linear parameter-varying plants," *Automatica*, vol. 27, no. 3, pp. 559-564, 1991.

[14] W. A. Sethares, B. D. O. Anderson, and C. R. Johnson, Jr., “Adaptive algorithms with filtered regressor and filtered error,” *Math. Control Signals and Systems*, no. 2, pp. 381-403, 1989.