

- [8] R. Greiner, B. A. Smith, and R. W. Wilkerson, "A correction to the algorithm in Reiter's theory of diagnosis," *Artif. Intell.*, vol. 41, no. 1, pp. 79–88, 1989.
- [9] A. C. Guyton, *Textbook of Medical Physiology*, 7th ed. Philadelphia, PA: Saunders, 1981.
- [10] W. Hamscher, "Modeling digital circuits for troubleshooting," *Artif. Intell.*, vol. 51, no. 1–3, pp. 223–271, 1991.
- [11] *Readings in Model-Based Diagnosis*, W. Hamscher, L. Console, and J. de Kleer, Eds. San Mateo, CA: Morgan Kaufmann, 1992.
- [12] B. Hayes-Roth, R. Washington, D. Ash, R. Hewett, A. Collinot, A. Via, and A. Seiver, "Guardian: A prototype intelligent agent for intensive-care monitoring," *Artif. Intell. Medicine*, vol. 4, no. 2, pp. 165–185, 1992.
- [13] B. Hayes-Roth, S. Uckun, J. E. Larsson, J. Drakopoulos, D. M. Gaba, J. Barr, and J. Chien, "Guardian: An experimental system for intelligent ICU monitoring," in *Proc. 18th Annu. Symp. Computer Applications Medical Care*, Washington D. C., 1994.
- [14] S. S. Jørgensen, "Generic MFM models for use in fault diagnosis of ship engines," Ph.D. dissertation, Inst. Automatic Control Syst., Technical Univ. Denmark, Lyngby, 1993.
- [15] J. de Kleer, "How circuits work," *Artif. Intell.*, vol. 24, no. 1–3, pp. 205–280, 1984.
- [16] —, "Focusing on probable diagnoses," in *Proc. 9th Nat. Conf. Artificial Intelligence*, Anaheim, CA, 1991, pp. 842–848.
- [17] J. de Kleer and B. C. Williams, "Diagnosing multiple faults," *Artif. Intell.*, vol. 32, no. 1, pp. 97–130, 1987.
- [18] M. N. Larsen, "Strips as a planning method within abstractions and MFM modeling," Tech. Rep., Inst. Automatic Control Syst., Technical Univ. Denmark, Lyngby, 1990.
- [19] J. E. Larsson, "Knowledge-based methods for control systems," Ph.D. dissertation, TFRT-1040, Dept. Automatic Control, Lund Inst. Technol., Lund, Sweden, 1992.
- [20] —, "Diagnostic reasoning strategies for means-end models," *Automatica*, vol. 30, no. 5, pp. 775–787, 1994.
- [21] —, "Hyperfast algorithms for model-based diagnosis," in *Proc. IEEE/IFAC Joint Symp. Computer-Aided Control Systems Design*, Tucson, AZ, 1994.
- [22] —, "A toolbox for fast model-based diagnosis," in *Working Notes AAAI '94 Workshop Representing Reasoning About Device Function*, Seattle, WA, 1994.
- [23] —, "Model-based diagnosis of the human body," in *Intelligent Systems*, E. A. Yfantis, Ed. Norwell, MA: Kluwer, 1995, pp. 405–412.
- [24] —, "Diagnosis based on explicit means-end models," *Artif. Intell.*, vol. 80, no. 1, pp. 29–93, 1996.
- [25] M. Lind, "Human-machine interface for diagnosis based on multi-level flow modeling," in *Proc. 2nd European Meet. Cognitive Science Approaches Process Control*, Siena, Italy, 1989.
- [26] —, "Representing goals and functions of complex systems—An introduction to multilevel flow modeling," Tech. Rep. 90-D-38, Inst. Automatic Control Syst., Technical Univ. Denmark, Lyngby, 1990.
- [27] —, "Abstractions version 1.0—Descriptions of classes and their use," Tech. Rep. 90-D-380, Inst. Automatic Control Syst., Technical Univ. Denmark, Lyngby, 1990.
- [28] —, "An architecture for real-time MFM diagnosis," Tech. Rep. Inst. Automatic Control Syst., Technical Univ. Denmark, Lyngby, 1990.
- [29] —, "Abstractions for modeling of diagnostic strategies," in *Proc. IFAC Workshop Computer Software Integrating AI/KBS Systems Process Control*, Bergen, Norway, 1991.
- [30] —, "On the modeling of diagnostic tasks," in *Proc. 3rd Europ. Conf. Cognitive Science Approaches Process Control*, Cardiff, Wales, 1991.
- [31] —, "A categorization of models and its application for the analysis of planning knowledge," in *Proc. Post ANP '92 Conf. Human Cognitive Cooperative Activities Advanced Technological Systems*, Kyoto, Japan, 1992.
- [32] —, "Modeling goals and functions of complex industrial plants," *Appl. Artif. Intell.*, vol. 8, no. 2, pp. 259–283, 1994.
- [33] M. Lind, and M. N. Larsen, "Planning support and the intentionality of dynamic environments," in *Expertise and Technology, Cognition and Human Computer Cooperation*, J.-M. Hoc, P. C. Cacciabue, and E. Hollnagel, Eds. Hillsdale, NJ: Lawrence Erlbaum, 1995, pp. 255–278.
- [34] K. Monta, J. Takizawa, Y. Hattori, T. Hayashi, N. Sato, J. Itoh, A. Sakuma, and E. Yoshikawa, "An intelligent man-machine systems for BWR nuclear reactor power plants," in *Proc. AI91—Frontiers Innovative Computing Nuclear Industry*, Jackson, WY, 1991.
- [35] R. L. Moore, H. Rosenof, and G. Stanley, "Process control using a real-time expert system," in *Proc. 11th Triennial IFAC World Congr. 1990*, Tallinn, Estonia, pp. 241–245.
- [36] A. Osman, "Graphical control environment (Grace)," Ph.D. dissertation, Inst. Automatic Control Syst., Technical Univ. Denmark, Lyngby, 1992.
- [37] O. O. Oyeleye, "Qualitative modeling of continuous chemical processes and applications to fault diagnosis," Ph.D. dissertation, Mass. Inst. Technol., Cambridge, 1989.
- [38] T. F. Petti, "Using mathematical models in knowledge-based control systems," Ph.D. dissertation, Univ. Delaware, Newark, 1992.
- [39] R. Reiter, "A theory of diagnosis from first principles," *Artif. Intell.*, vol. 32, no. 1, pp. 57–96, 1987.
- [40] J. M. A. Sassen, "Design issues of human operator support systems," Ph.D. dissertation, Delft Univ. Technol., Faculty of Mechanical Engineering Marine Technol., Delft, The Netherlands, 1993.
- [41] G. J. Sussman and G. L. Steele, "Constraints: A language for expressing almost-hierarchical descriptions," *Artif. Intell.*, vol. 14, no. 1, pp. 1–40, 1980.
- [42] D. Taverner, *Taverner's Physiology*, 4th ed. London, U.K.: Hodder and Stoughton, 1983.
- [43] P. Torasso and L. Console, *Diagnostic Problem Solving*. London, U.K.: North Oxford Academic, 1989.
- [44] J. Walseth, "Diagnostic reasoning in continuous systems," Ph. D. dissertation, Div. Eng. Cybernetics, Norwegian Inst. Technol., Trondheim, 1993.
- [45] J. Å Walseth, B. A. Foss, M. Lind, and O. Øgaard, "Models for diagnosis—Application to a fertilizer plant," in *Proc. IFAC Symp. On-Line Fault Detection Supervision Chemical Process Industries*, University of Delaware, Newark, 1992.
- [46] *Readings in Qualitative Reasoning About Physical Systems*, D. S. Weld and J. de Kleer, Eds. San Mateo, CA: Morgan Kaufmann, 1990.

A Framework for Hybrid Control Design

Rafael Fierro and Frank L. Lewis

Abstract— This paper presents a hybrid system framework which considers simultaneously the 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 algorithm for a hybrid control design is provided and illustrated through a mobile manipulator example.

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

Manuscript received July 27, 1995; revised April 14, 1996 and February 5, 1997.

The authors are with the Automation and Robotics Research Institute, The University of Texas at Arlington, Fort Worth, TX 76118-7115 USA (rfierro@arri.uta.edu).

Publisher Item Identifier S 1083-4427(97)07018-5.

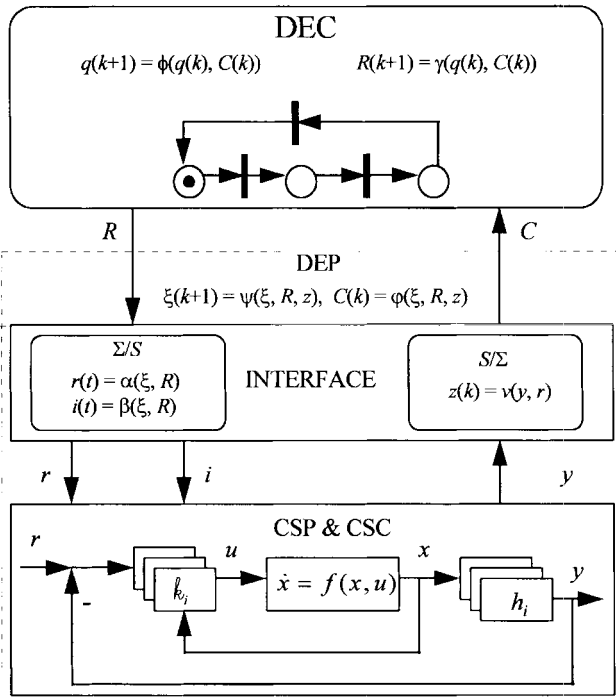


Fig. 1. Hybrid dynamical system.

systems must encapsulate the relevant *behaviors* and *events* needed for effective decision-making control. A hybrid system framework is needed for considering simultaneously the control and decision-making issues.

Hybrid systems have emerged as a technique for modeling and analyzing a class of autonomous control systems containing both continuous-state dynamics and logical clauses. In one formulation of hybrid systems, the continuous-state plant is supervised by a discrete-event controller. Such a hybrid system consists of three main components: a plant, a discrete-event controller, and an interface.

This three-layer *hybrid* framework has attracted the attention of the control community in the last few years [1]–[3], [5], [10], [11], [13]–[15]. 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 translate between the plant and the discrete-event controller. The interface converts 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. Note that *hybrid systems are very reminiscent of sampled data control*, where the plant plus *ZOH* plus sampler appears as a discrete-time system.

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 [17] is to associate discrete states with different plant outputs, controllers, and/or reference trajectories (c.f., [18]), 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 [20], mobile robots [22], intelligent vehicle highway systems (IVHS) [9], and flexible manufacturing systems [12].

The remainder of the paper is organized as follows. In Section II we present a mathematical model and some definitions of the hybrid framework introduced herein. Section III discusses the discrete-event controller. Section IV provides a hybrid control design algorithm which is illustrated through a mobile manipulator (i.e., mobile base plus onboard arm) example performing a sequential task in a manufacturing environment. Since discrete state changes correspond to changes in the structure of the continuous-state plant, guaranteeing the plant stability becomes a major concern. The *stability problem* is briefly addressed in Section V. Some connections between hybrid systems and fuzzy logic systems are established in Section VI. Finally, Section VII gives some concluding remarks.

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, \mathbb{R} , \mathbb{R}^+ , \mathbb{Z} , \mathbb{N} , denote the reals, nonnegative reals, i.e., $\mathbb{R}^+ = [0, +\infty)$, integers, and nonnegative integers, respectively. \underline{N} denotes the set $\{1, 2, \dots, N\}$. Let \mathbb{R}^n and $\mathbb{R}^{n \times m}$ denote the real n -space and the set of $n \times m$ real matrices, respectively. If $A \in \mathbb{R}^{n \times m}$, then A^T denotes the transpose of A .

Let Ω be a subset of \mathbb{R}^n . $\bar{\Omega}$ represents the closure of Ω , Ω° its interior, and $\partial\Omega$ its boundary.

B. Hybrid System Model

A class of HDS, commonly found in the literature, considers the three-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

$$\text{Discrete-event controller: } \begin{cases} q(k+1) = \phi(q(k), C(k)) \\ R(k) = \gamma(q(k), C(k)) \end{cases} \quad (1)$$

where $q(k) \in Q$ is the state after the k th event, $C \in \mathcal{C}$, $R \in \mathcal{R}$, $\phi: Q \times \mathcal{C} \rightarrow Q$, $\gamma: Q \times \mathcal{C} \rightarrow \mathcal{R}$, $k(t) \in \mathbb{N}$.

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

$$\text{Discrete-event plant: } \begin{cases} \xi(k+1) = \psi(\xi(k), R(k), z(k)) \\ C(k) = \varphi(\xi(k), R(k), z(k)) \end{cases} \quad (2)$$

where $\xi \in \mathcal{X}$, $\Lambda \subseteq \mathcal{R} \times \mathcal{Z}$, $z \in \mathcal{Z} \subset \mathbb{R}^b$, $\psi: \mathcal{X} \times \mathcal{R} \times \mathcal{Z} \rightarrow \mathcal{X}$, $\varphi: \mathcal{X} \times \mathcal{R} \times \mathcal{Z} \rightarrow \mathcal{C}$. $z(k)$ contains the event information from

the CSP, i.e., a signal that enables the state transition function in the DEP. The interface equations are

$$\text{Interface } (\Sigma/S): \begin{cases} r(t) = \alpha(\xi(k), R(k)) \\ i(t) = \beta(\xi(k), R(k)) \end{cases} \quad (3)$$

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

where $r \in \mathbb{R}^p$ is the reference trajectory, $i \in \underline{N}$ is a deterministic scalar index, $y \in Y \subset \mathbb{R}^p$. Functions $\alpha: \mathcal{X} \times \mathcal{R} \rightarrow \mathbb{R}^p$ and $\beta: \mathcal{X} \times \mathcal{R} \rightarrow \underline{N}$ are mappings in Σ/S , and function $v: Y \times \mathbb{R}^p \rightarrow Z$ 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

Continuous-state plant:

$$\begin{cases} \dot{x}(t) = f(x(t), u(t)) \\ y(t) = h(x(t), i(t)) \end{cases} \quad (5)$$

Continuous-state controller:

$$u(t) = \mathcal{K}(x(t), y(t), r(t), i(t)) \quad (6)$$

where $x \in X \subset \mathbb{R}^n$, $u \in U \subset \mathbb{R}^m$, $f: X \times U \rightarrow \mathbb{R}^n$ is assumed to be Lipschitz continuous, $h: X \times I \rightarrow Y$, $\mathcal{K}: X \times Y \times \mathbb{R}^p \times I \rightarrow U$. 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.

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, \mathcal{K}, r) , where $u(t) \in U$ and $y(t) \in Y$ are vectors of inputs and outputs, $\mathcal{K} \in \mathcal{K}$ is a library of stabilizing tracking controllers (adaptive, neural net, PID, etc.), and $r(t)$ is the reference trajectory. The controller \mathcal{K} should be designed so that the closed-loop system is stable with suitable robustness and performance properties. All the techniques of control theory can be used. (Compare with work by Shin [18] where the reference trajectory is selected to improve plant performance.)

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,’’ ‘‘exploration,’’ ‘‘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 $(u_i, y_i, \mathcal{K}_i, r_i)$, z is a symbol indicating event occurrence. A DEC or supervisor (c.f., [16]) monitors C and provides R . The closed-loop plant plus interface in Fig. 1 can represent, for instance a *workcell* in an 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

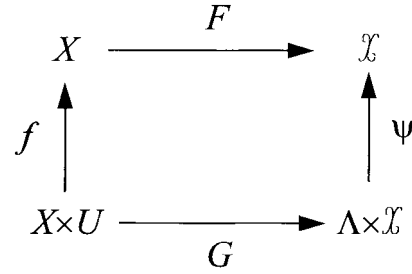


Fig. 2. Hybrid system’s commutative diagram.

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 the problem is to determine, given a continuous description of the plant, a discrete-time description that includes the sampler plus the hold mechanism.

To see what is involved, consider the following simplified problem. Let there be prescribed the continuous-state system (X, U, f) with X the state-space, U the input-space, and $f: X \times U \rightarrow X$ a state transition function. In this representation X, U are function spaces whose elements are functions of a scalar parameter t in some specified set T , and operation related to f are specified in some algebra. An example is the linear time-invariant system

$$\dot{x} = Ax + Bu \equiv f(x, u) \quad (7)$$

where $x(t), u(t)$ with the time $t \in \mathbb{R}$ and operations carried out over the standard $(+, \cdot)$ matrix algebra.

Given a set of discrete states \mathcal{X} , with a set of specified operations, and an input alphabet Λ , define an operator (‘‘generator’’) $F: X \rightarrow \mathcal{X}$ and an operator (‘‘inverse actuator’’) $G: U \rightarrow \Lambda$ as shown in the commutative diagram of Fig. 2. It is now desired to determine a finite state description $(\mathcal{X}, \Lambda, \psi)$ such that

$$\psi = F \circ f \circ G^{-1} \quad (8a)$$

$$\psi(F(x), G(u)) = F(f(x, u)) \quad (8b)$$

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 $\psi: \mathcal{X} \times \Lambda \rightarrow \mathcal{X}$. 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 $X = \mathcal{X} = \mathbb{R}^n$ and F is the sampling operator

$$F(x) = \sum_k x(k\tau)\delta(t - k\tau) \quad (9)$$

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 VI.

III. UPPER-LEVEL DE DESIGN—LINKED FSM

We have just shown how to approach the design and analysis of the lowest level hierarchical step in a hybrid system architecture. Using

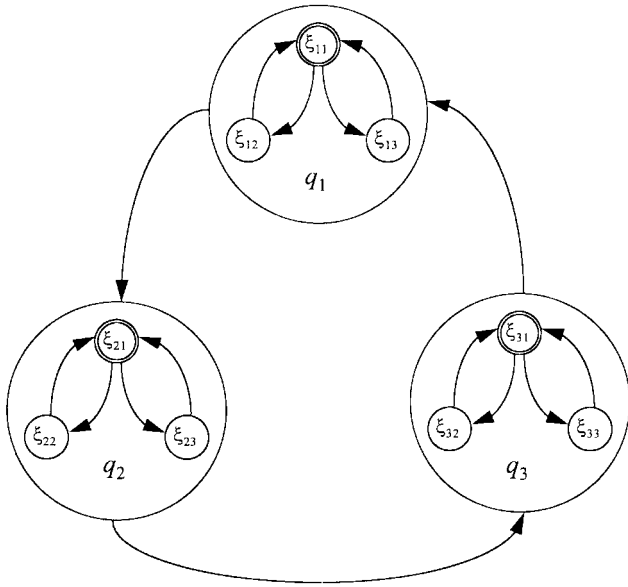


Fig. 3. FSM interconnected hierarchically.

the following structure, it is not difficult to formalize the design of the higher DE levels. The structure addresses considerations of various researchers who are by now contending that *strict hierarchies* are unsuitable for real-world planning and control problems.

It is well-known that FSM can be used to model plans, tasks, behaviors, etc. Formally, a FSM is defined as $M = (Q, \Sigma, \delta, q_0, F)$, where Q is the set of states, Σ is an input alphabet, δ is a transition function, q_0 is the initial state, and F is the set of final states. A FSM is a special case of a Petri net, wherein each transition has a unique place both upstream and downstream. Unfortunately, FSM have some limitations for intelligent control (IC) purposes as they cannot accommodate shared resources or task choices. That is, the interconnections of FSM provide insufficient generality for control interconnections in real-world systems. FSM can be formally interconnected in several ways, including the hierarchical method in Fig. 3. In this figure, each FSM contains another FSM with well-defined entry and output states. The composite FSM here is also a FSM. However, the more general *linked* FSM structure depicted in Fig. 4 is more suitable for control purposes.

In Fig. 4, it is obvious that the state transitions are activated by *control links* into PN-like transitions, and that each transition causes some “outputs” or *events* to occur. The control links and event links are connected to either upper-level FSM, the event links become “status signals,” when connected to lower level FSM they become “commands.”

The overall linked FSM is not a FSM, but has a more general PN structure that can accommodate shared resources, conflicts, and decisions. Thus, the techniques, for instance in [16], can be used to define a FSM_2 (the PN supervisor) in Fig. 4. The linked FSM structure is nothing more than a formal PN representation of *Mealy machines*, where each state has a single associate input and output.

The hybrid (low level) in an IC hierarchy can also be represented as in Fig. 4. Then, the states in the FSM_1 represent the operational plant behavior elements just defined. That is, each state represents a specific closed-loop plant behavior $(u_i, y_i, \mathcal{K}_i, r_i)$. With each change of state, FSM_1 would send commands R to the closed-loop system to change the controller, the reference trajectory, or the output measured or control inputs employed.

Within this framework, where DE state changes correspond to changes in the *structure* of the continuous-state plant (including

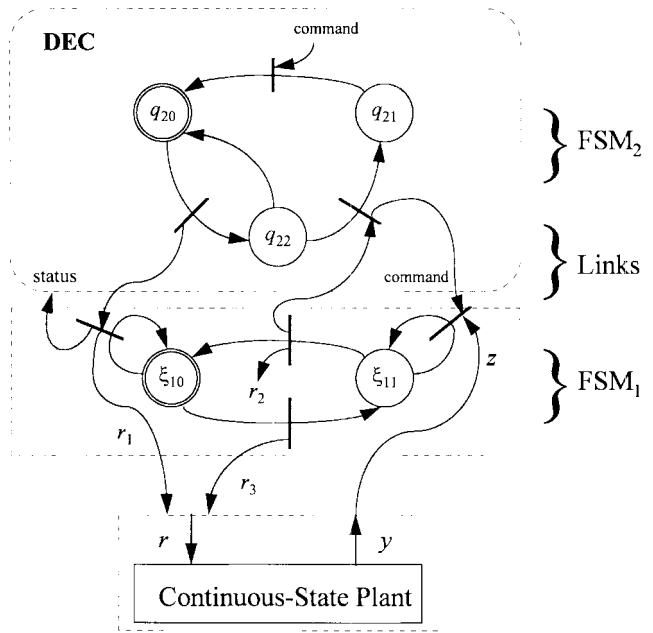


Fig. 4. FSM with control and event links.

reference controllers and reference trajectories), it will be important to study the *stability* of the continuous-state part of the system as the changes in DE states occur.

Finally, it is observed that linked FSM can be represented as the *logical matrix controller equations*. In fact, linked FSM yield a *structure block matrix form* of rule-based controller. This allows matrix analysis and refinement of the design of the deadlock removal, and show how dispatching rules can be used in outer control loops for conflict resolution (see [12] for details).

It should be clear that linked FSM include hierarchical structures, but also *nonhierarchical structures* since FSM can be linked in parallel at the same level.

A. Task Definitions and Specification as FSM

A plan represents a sequence of actions that achieves a given goal. A ‘feasible goal’ is an ordered sequence of ‘primitive goal actions’ i.e., closed-loop behaviors. Plans can be represented by a finite state machine. The arcs in the FSM represent an ordering in the states required to accomplish a goal state.

Depending on the task, the upper-level FSM in our linked FSM structure are generated and nested into the system with control and event links (which are transparent to the task planner). Thus, in our “hybrid structure,” the lower level DE systems represent the fixed plant-plus-controllers (the resources), while the upper-level DE systems are evanescent or “virtual” and are reconfigured as the task change.

IV. A HYBRID CONTROL DESIGN ALGORITHM

In this section, we apply the hybrid framework given by (1)–(6) to a mobile manipulator performing a task in a manufacturing plant. Consider the system shown in Fig. 5. A mobile manipulator [22] 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 \mathcal{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

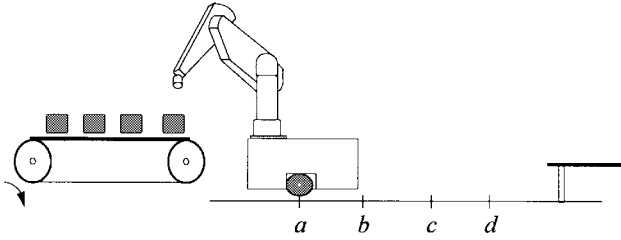


Fig. 5. Mobile manipulator.

controller \mathcal{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 \mathcal{K}_3 to drive the onboard arm to pick/release the object. This example is an extension of a case study presented in [6].

The design algorithm considers the three layers of the hybrid system in Fig. 1, separately. In each layer we define states, inputs, outputs, and the corresponding transition and output functions. Then the appropriate interfaces and connections between layers are defined. The algorithm 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)–4).

1) *Abstract representation of the CSP:* The continuous-state system, plant plus low-level servo controller, is represented by an abstract model which captures the main features of the original system. In this example, we describe the position of the mobile base by a first order differential equation given by

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

If the mobile base goes to the right the controller \mathcal{K}_1 is used and $\lambda_1 = 2.5 \text{ (s}^{-1}\text{)}$. On the other hand, if the mobile base goes to the left the controller \mathcal{K}_2 is used and $\lambda_2 = 5 \text{ (s}^{-1}\text{)}$. Therefore, the mobile base moves faster after releasing the object. 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.

2) *Discretization of the continuous-state plant:* 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 $\xi_2(k) = \{0 \equiv \text{pick_object}, 1 \equiv \text{release_object}\}$. The DEP inputs (i.e., commands) are given by the position command $U_1(k) = \{0 \equiv \text{right}, 1 \equiv \text{left}\}$ and the onboard arm command $U_2(k) = \{0 \equiv \text{hold}, 1 \equiv \text{release}\}$. The DEP dynamics are given by

$$\begin{aligned} \xi_1(k+1) &= \psi_1(\xi_1(k), U_1(k)) \\ \xi_2(k+1) &= U_2(k) \\ C_1(k) &= \xi_1(k) \\ C_2(k) &= \xi_2(k). \end{aligned} \quad (11)$$

The state transition function of the mobile base position is given in Fig. 6. From this figure we can infer the oscillating nature of the mobile base position when it performs the task repeatedly (c.f., switching trajectories in *bang-bang* control). Although some information details are lost in the abstraction process, the fundamental properties of the system behavior do not change.

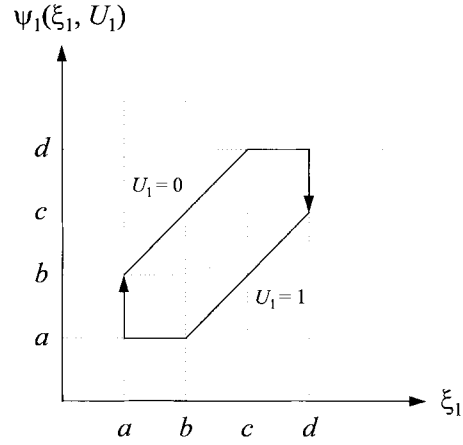


Fig. 6. ‘Next’ mobile base position.

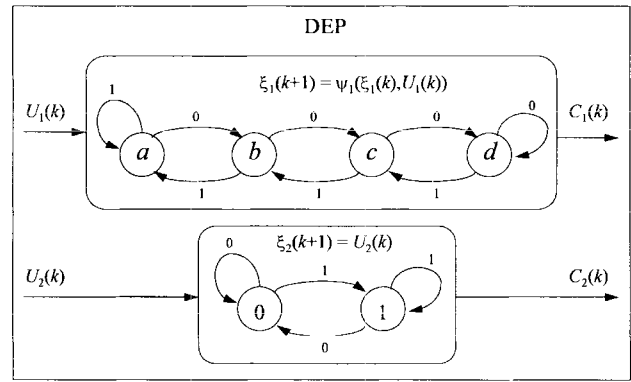


Fig. 7. FSM representation of the DEP.

Whenever the mobile base crosses certain boundaries an event is generated and the *event function* $k(t)$ is increased by one. The event function has the same meaning as time in discrete-time systems. However in DES the sampling interval is not uniform. The FSM representation of the DEP is showed in Fig. 7.

3) *DEC design:* The DEC supervises the performance of the DEP and chooses the inputs to perform the given task. When the mobile base goes to the right, the end-effector holds the object. The end-effector releases the object in position d and the mobile manipulator returns to position a .

The DEC state is given by $q_1(k) = \{0 \equiv \text{to_right}, 1 \equiv \text{to_left}\}$, and the DEC input, $U_{C1}(k)$, is the mobile base position. Note that we just need one state in the DEC because the action of the end-effector depends on what direction the mobile base is moving. The DEC dynamics are given by

$$\begin{aligned} q_1(k+1) &= \phi(q_1(k), U_{C1}(k)) \\ R_1(k) &= q_1(k) \\ R_2(k) &= \gamma(q_1(k), U_{C1}(k)) = q_1(k+1). \end{aligned} \quad (12)$$

The closed-loop DE system is defined as

$$\begin{aligned} U_{C1}(k) &= C_1(k) \\ U_1(k) &= R_1(k) \\ U_2(k) &= R_2(k) \end{aligned} \quad (13)$$

and the dynamics of the closed-loop DE system become

$$\text{DEC: } \begin{cases} q_1(k+1) = \phi(q_1(k), \xi_1(k)) \\ R_1(k) = q_1(k) \\ R_2(k) = \gamma(q_1(k), \xi_1(k)), \end{cases} \quad (14)$$

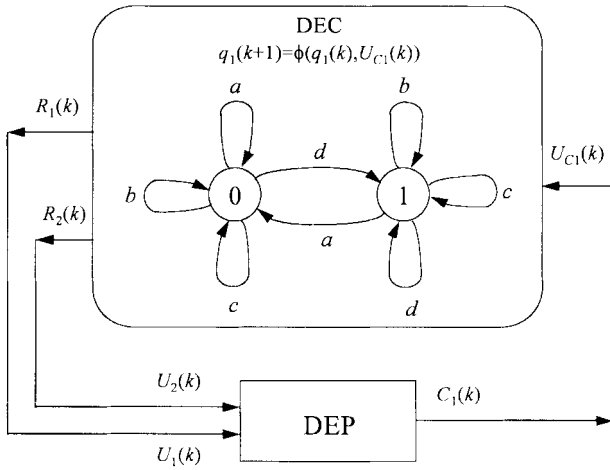


Fig. 8. The closed-loop DE system.

$$\text{DEP: } \begin{cases} \xi_1(k+1) = \psi_1(\xi_1(k), q_1(k)) \\ \xi_2(k+1) = \gamma(q_1(k), \xi_1(k)) \\ C_1(k) = \xi_1(k) \\ C_2(k) = \xi_2(k). \end{cases} \quad (15)$$

The closed-loop DE system is presented in Fig. 8.

4) *Interface design* ($S/\Sigma, \Sigma/S$): 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

$$u(t) = \alpha(\xi_1(k), R_1(k)) = \alpha(\psi_1(\xi_1(k), R_1(k)))$$

$$u = \begin{cases} 0 & \text{if } \psi_1(\cdot) = a \\ 2 & \text{if } \psi_1(\cdot) = b \\ 4 & \text{if } \psi_1(\cdot) = c \\ 6 & \text{if } \psi_1(\cdot) = d, \end{cases} \quad (16)$$

$$i(t) = \beta(\xi_1(k), R_1(k)), \quad i = \begin{cases} 1 & \text{if } R_1(k) = 0 \\ 2 & \text{if } R_1(k) = 1. \end{cases} \quad (17)$$

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

$$z(k) = v(y, u), \quad z(k) = \begin{cases} 1 & \text{if } |u(t) - y(t)| < \varepsilon \\ 0 & \text{otherwise.} \end{cases} \quad (18)$$

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

$$\xi_1(k+1) = \psi_2(\xi_1(k), q(k), z(k)) \quad (19)$$

$$\psi_2(\cdot) = \begin{cases} \xi_1(k) & \text{if } z(k) = 0 \\ \psi_1(\cdot) & \text{if } z(k) = 1 \end{cases} \quad (20)$$

$$k(t) = \begin{cases} k+1 & \text{if } z(k) = 1 \\ k & \text{if } z(k) = 0. \end{cases} \quad (21)$$

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. 9 depicts the closed-loop HDS. Finally, Fig. 10 shows a simulation of the hybrid system. Note that the mobile manipulator moves faster when it goes to the left.

V. THE STABILITY PROBLEM IN HYBRID DYNAMICAL SYSTEMS

In this section we shall consider the stability of the hybrid system introduced in Section II. The discrete-event dynamics is abstracted

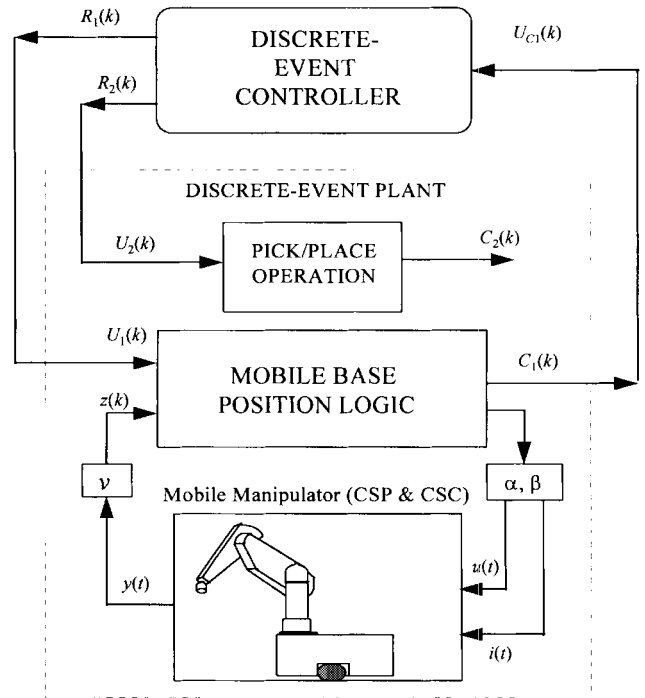


Fig. 9. The closed-loop HDS.

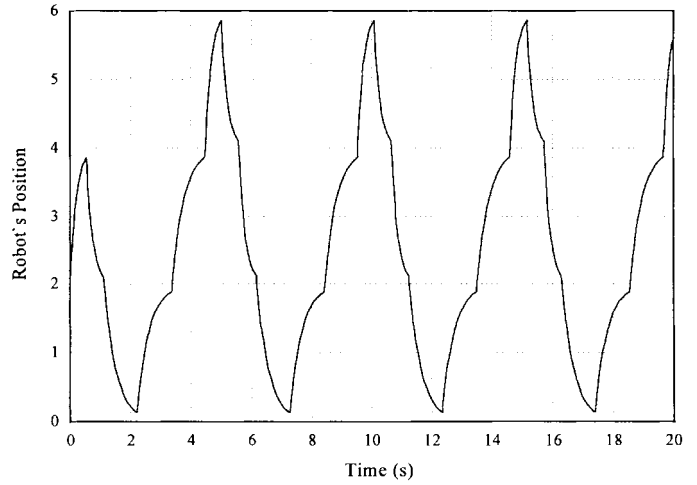


Fig. 10. Mobile base position.

away in order to study the stability of the continuous-state part of the hybrid system. The hybrid system \mathcal{H} given by (1)–(6) is *embedded* (c.f., [23]) into a switched system $\tilde{\mathcal{H}}$ given by

$$\dot{x}(t) = f_i(x(t)), \quad i \in \{1, 2, \dots, N\} \equiv \underline{N} \quad (22)$$

and

$$i(t) = g(i(t^-)), \quad x(t) = j, \quad \text{if } i(t^-) = i \text{ and } x(t) \in \Omega_{ij} \quad (23)$$

where $x \in X \subseteq \mathbb{R}^n$ is the continuous-state, and $i \in \underline{N}$ is the discrete-state. The evolution function $f(\cdot)$ is assumed to be globally Lipschitz, i.e., $f(\cdot) \in C^\infty(\mathbb{R}^n)$. Let Ω_{ij} denote a region of \mathbb{R}^n such that for any $i \in \underline{N}$ we have: 1) Ω_{ij} with $j \in \underline{N}$ cover the state space \mathbb{R}^n , and 2) Ω_{ij} and Ω_{ik} with $j \neq k$ do not overlap. It is assumed that if the continuous-state hits a certain boundary $\partial\Omega_{ij}$ (i.e., a switching

event) the discrete-state is given by (23) where $g: \underline{N} \times X \rightarrow \underline{N}$ is the *simplified* discrete-state dynamics.

Some definitions are required in order to proceed.

1) A *valid switching sequence* \mathcal{S} , for a given initial continuous-state x_0 , is defined by the pair (i_k, t_k) as

$$\mathcal{S} = \{(i_0, t_0), (i_1, t_1), \dots, (i_{k-1}, t_{k-1}), (i_k, t_k), \dots\} \quad (24)$$

where $t_0 < t_1 < \dots < t_{k-1} < t_k$, and $i_k \in \underline{N}$. As it can be seen, the evolution of the system is governed by the vector field $f_{i_k}(\cdot)$ on the interval $t_k \leq t < t_{k+1}$. We shall call t_k a *switching time*.

2) We define an increasing time sequence T_k

$$T_k = \{t_0, t_1, t_2, \dots, t_{k-1}, t_k, t_{k+1}, t_{k+2}, \dots\}. \quad (25)$$

3) If in a ball $B(r_0) = \{x \in \mathbb{R}^n | x^T x < r_0^2\}$, the function $V(x)$ is positive definite and has continuous partial derivatives, and if its time derivative along any state trajectory $x(\cdot)$ of system $\dot{x} = f(x)$, is negative semi-definite, i.e., $\dot{V}(x) \leq 0$, then $V(x)$ is said to be a *Lyapunov function* for the system $\dot{x} = f(x)$ [19].

4) x_e is an equilibrium point of the hybrid system (22) if $f_i(x_e) = 0$ for all $i \in \underline{N}$.

5) The equilibrium point of the hybrid system (22) is *stable* if for every $\varepsilon > 0$, and $t_0 \in T_0$ there exists a $\delta = \delta(\varepsilon, t_0) > 0$ such that for any $\|x(t_0) - x_e\| < \delta$ and for any $i(t_0) \in \underline{N}$, we have $\|x(t) - x_e\| < \varepsilon$ for all $t \geq t_0$. The equilibrium point is said to be *uniformly stable* if $\delta = \delta(\varepsilon)$. Moreover, the equilibrium point is said to be *asymptotically stable* if it is stable and $\|x(t) - x_e\| \rightarrow 0$ as $t \rightarrow \infty$. \square

In the framework presented in Section II, the decisions of the DEC are associated with changes in the closed-loop plant behavior. Therefore, the stability of the continuous-state part of the hybrid system depends on the decisions of the DEC. If the *behaviors* share a common Lyapunov function, then any arbitrary sequence between them can be proven to be stable. In this case, a *separation principle* holds; that is, the design of the DE controller and the servo controller can be carried out independently [8].

In general finding a common Lyapunov function that represent a set of behaviors is not an easy task. *Multiple Lyapunov function theory* may be used to study stability of hybrid systems [4].

We should like to study the stability of a hybrid system where switching sequences may be periodic, aperiodic, finite or infinite, and include some unstable modes. In the next theorem we provide some stability criteria for hybrid dynamical systems.

Theorem: Assume that there exists a finite number of scalar positive definite functions $V_i(x): X \rightarrow \mathbb{R}^+$, with $i = 1, 2, \dots, N$ and continuous first order partial derivatives, corresponding to the continuous-state vector fields $\dot{x} = f_i(x)$ with $f_i(0) = 0$, for all i .

a) Let \mathcal{S} be the set of all valid switching sequences associated with the system, and t_k be the switching times if the following holds.

- i) There are no *jumps*, i.e., $x(t_k^-) = x(t_k^+) = x(t_k), \forall k$.
- ii) There exists a positive definite function $\phi(x)$ such that

$$DV(x(t_j)) \leq -\phi(x) \quad (26)$$

where

$$DV(x(t_j)) \equiv V_{i_{j+1}}(x(t_{j+1})) - V_{i_j}(x(t_j)), \quad \forall i_j \in \underline{N}. \quad (27)$$

iii) V_i is radially unbounded, i.e., $\lim_{\|x\| \rightarrow \infty} (V_i(x)) = \infty$.

iv) $t_{\min} \leq t_{k+1} - t_k < \infty$ with $t_{\min} > 0 \forall t_k \in T_k$, then the state of the system globally $\|x(t)\| \rightarrow 0$ as $t \rightarrow \infty$ over \mathcal{S} .

b) If conditions i)–iv) in part a) above are satisfied, and in addition we have

v) V_i is nonincreasing and $\dot{V}_i < 0 \forall i$, then the origin of the continuous-state space of the hybrid system is a globally uniformly asymptotically stable equilibrium point over \mathcal{S} .

c) Let $\mathcal{S}_N = \{(i_0, t_0), (i_1, t_1), \dots, (i_N, t_N), (i_0, t_0 + \Delta), \dots\}$ be a *periodic* valid switching sequence with period Δ . If assumptions iii) and iv) hold, and ii) is modified as follows.

ii*) There exists a positive definite function $\phi(x)$ such that

$$V_{i_j}(x(t_j + \Delta)) - V_{i_j}(x(t_j)) \leq -\phi(x), \quad \forall i_j \in \underline{N} \quad (28)$$

then the state of the system globally $\|x(t)\| \rightarrow 0$ as $t \rightarrow \infty$ over the periodic sequence \mathcal{S}_N .

d) Given a hybrid system \mathcal{H} whose dynamics are governed by (22) and (23). If assumptions iii) and iv) hold, and v) are valid for all regions Ω_{i_j} and in addition we have

vi) the Lyapunov functions have the same value on the boundaries $\partial\Omega_{i_j}$, that is $V_1(x) = V_2(x) = \dots = V_N(x)$ for all $x \in \partial\Omega_{i_j}$, then the origin of the continuous-state space of the hybrid system is a globally asymptotically stable equilibrium point (c.f., [6]).

Proof: (Outline)

a) Since $\{V_{i_j}(x(t_j))\}$ is a strictly decreasing sequence and lower bounded by zero, then $\lim_{k \rightarrow \infty} V_{i_k}(x(t_k)) = L \geq 0$ exists [21] and

$$\lim_{k \rightarrow \infty} [DV(x(t_k))] \leq \lim_{k \rightarrow \infty} [-\phi(x)] = 0 \quad (29)$$

this implies that $\lim_{k \rightarrow \infty} x(t_k) = 0$.

b) Considering ii) and v) it can be shown that for any initial time $t_0 \in T$, for a given $\varepsilon > 0$, and for any $i(t_0) \in \underline{N}$, there exists a $\delta = (\varepsilon, t_0) > 0$ such that for any $\|x(t_0)\| < \delta$, we have $\|x(t)\| < \varepsilon$ for all $t \geq t_0$. Clearly, the system is stable in the sense of Lyapunov. Convergence was proved in a), and by using iii) we can conclude that the equilibrium point of the hybrid system is *asymptotically stable in the large*. Note that the *energy* of the hybrid system can be thought of being bounded by a nonincreasing sequence $\{V_{i_j}(x(t_j))\}$ for all $t > t_0$.

c) Similar to a).

d) Suppose the continuous dynamics is governed by $f_{i_k}(\cdot)$, and the continuous-state hits a boundary $\partial\Omega_{i_k i_{k+1}}$ at t_{k+1} , then the discrete-state becomes i_{k+1} and the continuous-state dynamics switch to $f_{i_{k+1}}(\cdot)$. Let \tilde{H} denote the *embedded* switching system of the original hybrid system \mathcal{H} , and $\tilde{\mathcal{S}} = \{(i_0, t_0), (i_1, t_1), \dots, (i_k, t_k), (i_{k+1}, t_{k+1}), \dots\}$ denote the valid switching sequence. Considering ii), iii), v), and vi), it can be shown that the origin of the continuous-state space of \tilde{H} (\mathcal{H}) is a globally asymptotically stable equilibrium point. \square

This section provides a basic framework for the stability analysis of a class of hybrid systems. Note that in (23), the “next” $i(t)$ depends on the evolution of the continuous state $x(t)$. Therefore the switching sequence (24) does depend on the continuous state implicitly. In fact, we focus our analysis on hybrid systems with a *static* DEC, i.e., the switching device has no memory. A more elaborated stability analysis is beyond the scope of this paper. The interested reader is referred to [4], [6], [21], and [23].

VI. 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

$$\dot{x}(t) = f(x, u) \quad (30)$$

where $x \in \mathbb{R}^n$ is the state variable, $u(t) \in \mathbb{R}$ is the system input, $f: \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}^n$ is a smooth mapping defined on an open set $\Phi \subset \mathbb{R}^n \times \mathbb{R}$. The hybrid control problem consists of linearizing the system (30) 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. Define a fuzzy rule base of the form

$$\begin{aligned} R^{(i)}: & \text{IF } x_1 \text{ is } \mathbf{X}_1^{(i)} \& \cdots \& x_n \text{ is } \mathbf{X}_n^{(i)}, \\ & \text{THEN } \dot{x} = A_i x + B_i u, i = 1, 2, \dots, N \end{aligned} \quad (31)$$

where $\mathbf{X}_k^{(i)}, k = 1, 2, \dots, n$, 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

$$\omega^{(i)} = \prod_{j=1}^n \mu_{\mathbf{X}_j^{(i)}}(x_j). \quad (32)$$

The N fuzzy rules (31) partition the state space $X \in \mathbb{R}^n$ into N regions Ω_i . It is assumed that $X = \bigcup_{i=1}^N \Omega_i$ and $\Omega_i, i = 1, 2, \dots, N$ are compact sets. The partition Ω_i is defined by the set

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

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

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

Each linear model (31) can be considered as a local representation of the nonlinear system (30) in region Ω_i . Then, the following fuzzy logic rule defines what controller to connect to the actual nonlinear plant

$$\begin{aligned} R^{(i)}: & \text{IF } x_1 \text{ is } \mathbf{X}_1^{(i)} \& x_2 \text{ is } \mathbf{X}_2^{(i)} \& \cdots \& x_n \text{ is } \mathbf{X}_n^{(i)} \\ & \text{THEN } u_i = -K_i x, i = 1, 2, \dots, N. \end{aligned} \quad (33)$$

Equation (33) provides a *local* control law. The gain $K_i \in \mathbb{R}^{1 \times n}$ 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_∞).

The closed-loop local system becomes

$$\dot{x} = [A_i - B_i K_i] x \equiv H_i x, \quad x \in \Omega_i \quad \text{and } i = 1, 2, \dots, N. \quad (34)$$

Note that system (34) is compatible with the hybrid model introduced in Section V. Therefore, the stability results outlined in that section can be applied to this class of fuzzy logic systems [8].

VII. 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.

Since DE state changes correspond to changes in the structure of the continuous-state plant, guaranteeing the plant stability becomes a major concern. It is not enough to design stable and robust low-level controllers without considering the DE decisions. The same is applied for the higher-level discrete-event system; as it reasons over an approximate abstract model, the commands may not always be consistent with the actual state of the continuous plant. Section V presents the rudiments of a Lyapunov stability theory for a class of hybrid systems where a discrete-state system supervises a multimodal continuous-state plant. Stability of the continuous-state part of the system is analyzed by using multiple Lyapunov functions.

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. This idea is illustrated using the well-known Takagi–Sugeno fuzzy model.

ACKNOWLEDGMENT

The authors would like to acknowledge the anonymous referees for their useful comments and constructive suggestions.

REFERENCES

- [1] P. J. Antsaklis, M. Lemmon, and J. A. Stiver, "Modeling and design of hybrid control systems," in *Proc. IEEE Mediterranean Symp. New Directions Control Automation*, Crete, Greece, 1994, pp. 440–447.
- [2] *Hybrid Systems II, Lecture Notes in Computer Science*, P. J. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, Eds. New York: Springer, 1995, vol. 999.
- [3] M. S. Branicky, V. S. Borkar, and S. K. Mitter, "A unified framework for hybrid control," in *Proc. IEEE Conf. Decision Control*, Lake Buena Vista, FL, 1994, pp. 4228–4234.
- [4] M. S. Branicky, "Stability of switched and hybrid systems," in *Proc. IEEE Conf. Decision Control*, Lake Buena Vista, FL, 1994, pp. 3498–3503.
- [5] R. W. Brockett, "Hybrid models for motion control systems," in *Essays on Control: Perspectives in the Theory and its Applications*, H. L. Trentelman and J. C. Willems, Eds. Boston, MA: Birkhäuser, 1993, pp. 29–53.
- [6] M. Dogruel and Ü. Özgüner, "Modeling and stability issues in hybrid systems," in *Hybrid Systems II, Lecture Notes in Computer Science*, P. J. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, Eds. New York: Springer, 1995, pp. 148–165.
- [7] J. Ezzine and A. H. Haddad, "Controllability and observability of hybrid systems," *Int. J. Control*, vol. 49, no. 6, pp. 2045–2055, 1989.
- [8] R. Fierro, F. L. Lewis, and C. T. Abdallah, "Common, multiple and parametric Lyapunov functions for a class of hybrid dynamical systems," in *Proc. IEEE Mediterranean Symp. New Directions Control Automation*, Crete, Greece, 1996, pp. 77–82.
- [9] D. N. Godbole, J. Lygeros, and S. Sastry, "Hierarchical hybrid control: An IVHS case study," in *Proc. IEEE Conf. Decision Control*, Lake Buena Vista, FL, 1994, pp. 1592–1597.
- [10] A. Göllü and P. Varaiya, "Hybrid dynamical systems," in *Proc. IEEE Conf. Decision Control*, Tampa, FL, 1989, pp. 2708–2712.
- [11] *Hybrid Systems, Lecture Notes in Computer Science*, R. Grossman, A. Nerode, A. Ravn, and H. Rischel, Eds. New York: Springer, 1993, vol. 736.
- [12] F. L. Lewis, H-H. Huang, R. Fierro, and D. Tacconi, "Real-time task planning, resource allocation, and deadlock avoidance," in *Proc. ISIC Workshop Architectures Semiotic Modeling Situation Analysis Large Complex Systems*, Monterey, CA, 1995, pp. 347–355.
- [13] K. M. Passino and Ü. Özgüner, "Modeling and analysis of hybrid systems: Examples," in *Proc. IEEE Int. Symp. Intell. Control*, Arlington, VA, 1991, pp. 251–256.
- [14] P. Peleties and R. DeCarlo, "A modeling strategy with event structures for hybrid systems," in *Proc. IEEE Conf. Decision Control*, Tampa, FL, 1989, pp. 1308–1313.
- [15] A. Puri and P. Varaiya, "Modeling and verification of hybrid systems," in *Proc. Amer. Control Conf.*, Seattle, WA, 1995, pp. 4466–4470.

- [16] P. J. Ramadge and W. M. Wonham, "The control of discrete event systems," *Proc. IEEE*, vol. 77, no. 1, pp. 81–98, 1989.
- [17] K. Ratlif, R. Fierro, R. Richardson, and F. Lewis, "Toward a framework for intelligent control," *Proc. ISPE/IFAC Int. Conf. CAD/CAM, Rob. Factories Future, CAR's FOF '94*, Ottawa, Ont., Canada, 1994, pp. 921–927.
- [18] K. G. Shin and X. Cui, "Design of a knowledge-based controller for intelligent control systems," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, no. 2, pp. 368–375, 1991.
- [19] J. E. Slotine and W. Li, *Applied Nonlinear Control*. Englewood Cliffs, NJ: Prentice-Hall, 1991.
- [20] B. L. Stevens and F. L. Lewis, *Aircraft Control and Simulation*. New York: Wiley, 1992.
- [21] M. A. Wicks, P. Peleties, and R. A. DeCarlo, "Construction of piecewise Lyapunov functions for stabilizing switched systems," in *Proc. Conf. Decision Control*, Lake Buena Vista, FL, 1994, pp. 3498–3503.
- [22] Y. Yamamoto and X. Yun, "Coordinating locomotion and manipulation of a mobile manipulator," in *Recent Trends in Mobile Robots*, Y. F. Zheng, Ed.. Singapore: World Scientific, 1993, pp. 157–181.
- [23] H. Ye, A. N. Michel, and L. Huo, "Stability theory for hybrid dynamical systems," in *Proc. IEEE Conf. Decision Control*, New Orleans, LA, 1995, pp. 2679–2684.
- [24] B. P. Zeigler, "DEVS representation of dynamical systems: Event-based intelligent control," *Proc. IEEE*, vol. 77, no. 1, pp. 72–80, 1989.

Defining Visual Languages for Interactive Computing

P. Bottoni, M. F. Costabile, S. Levialdi, and P. Mussio

Abstract— A novel definition of visual languages allows a uniform approach to satisfying the needs of visual reasoning faced in visual human–computer interaction. The way the machine associates a computational meaning with an image, and conversely, the way it generates an image on the screen from a computation are formally described. A definition of visual sentence and of visual language as a set of visual sentences is discussed. A hierarchy of visual languages is derived in relation with the requirements for intelligible, manageable and trustable interaction between humans and computers.

I. INTRODUCTION

Interactive computing is the present challenge of computer science [1]. Nowadays the computer is primarily seen as a communication tool, and modern graphical workstations have placed more emphasis on human–computer interaction (HCI) via pictorial representations, including images, sketches, diagrams, forms as well as text. Pictorial representations play two fundamental roles: communication between humans and programs and communication among humans through computers. In both cases, pictorial representations are the exchanged messages. Humans act on them to steer the computation and/or the communication. Computer systems act on them to respond to human requests and to synthesize the state of the interactive computation.

Pictorial representations have been widely used in scientific and technical communication and reasoning [2]. In HCI, pictorial repre-

Manuscript received December 19, 1995; revised November 25, 1996. This work was supported in part under Grant CNR 96.02297.07 and Murst 60%.

P. Bottoni, S. Levialdi, and P. Mussio are with the Dipartimento di Scienze dell'Informazione, Università "La Sapienza" di Roma, 00198 Roma, Italy (e-mail: bottoni@dsi.uniroma1.it).

M. F. Costabile is with the Dipartimento di Informatica, Università di Bari, 70126 Bari, Italy.

Publisher Item Identifier S 1083-4427(97)07019-7.

sentations carry a computational meaning, thus becoming a means to drive the underlying machine program.

In this paper we formalize the way the machine associates a computational meaning with an image and, conversely, the way it generates an image on the screen from a computation. To this end, we regard the messages exchanged during visual human–computer interaction as images, in that they are represented on a screen which is structured as a rectangular array of pixels, and we generalize the concept of image to any arrangement of signs appearing on the whole screen.

We define a visual sentence as a three-component structure consisting of the image on the screen, its meaning, i.e., the program managing the computation, and the relations between program and image components.

The main contribution of this paper is the definition of a set of visual sentences in a human–computer interaction as a visual language; it is shown how such a language may be constructed from a finite generator set. This provides the framework to formalize the relations between images and meanings. The meanings are formalized as strings of attributed symbols, generalizing the seminal work of Fu [3] in pattern recognition. Operations on visual sentences are defined which preserve the structure of operations on the separate components. As a consequence, pictorial and visual alphabets are defined.

A further result of this paper is a hierarchical taxonomy of visual languages with respect to the requirements for intelligible, manageable and trustable interaction between humans and computers [4]. Such a hierarchy is obtained by framing the theory of visual languages within the theory of formal languages. In particular, image components are characterized via a bidimensional generalization of the concept of string, and constructive operations on bidimensional strings are introduced.

The paper is organized as follows. In Section II, related work is briefly surveyed. Section III illustrates the basic concepts of images, descriptions and visual sentences, together with their constructive operations and the corresponding alphabets. Visual languages are defined in Section IV. Properties of visual sentences and visual languages are studied in Sections V and VI, respectively, and a hierarchy of visual languages is derived. Further developments of the proposed theory are outlined in the conclusions in Section VII.

II. RELATED WORK

It is acknowledged that a visual language is a tool for a human to interact with the computer to perform a task [5]. Several authors agree that the interaction must be controlled by the human who retains responsibility of the work [4]; moreover, the two agents, human and computer, must understand each other, and the computer must be trustable for the human. Our approach contributes to the solution of this problem by formalizing the relations established between images and meaning by each agent. This is the basis for the study of (dis)agreement between a human and a computer in interpreting an image or expressing a meaning by an image.

Evolving the model of Abowd and Beale, reported in [6], a typical interactive session is modeled as follows [7]. A human looks at a display, *interprets* the image appearing on it to understand the state of the work, decides which action to perform next, and composes (i.e., *materializes* on the screen) new messages to tell the computer the action to perform. On the computer side, the program manages the screen bitmap: the messages composed on the screen by the human