Discrete Approximation and Supervisory Control of Continuous Systems

Jörg Raisch and Siu D. O'Young

***Abstract*— This contribution addresses the following hybrid control problem: a continuous plant (its state evolving in Euclidean space) is to be controlled via symbolic output feedback—both measurement and control signal "live" on finite sets of symbols. We adopt the following approach: the hybrid problem is first translated into a purely discrete problem by approximating the continuous plant model by a (nondeterministic) finite-state machine. By taking into account past measurement and control symbols, approximation accuracy can be improved and adjusted to the specification requirements. Supervisory control theory for discrete-event systems (DES) is then applied to find the optimal controller which enforces the specifications. As the behavior of the approximating automaton is guaranteed to contain the behavior of the underlying continuous plant model, the controller also forces the latter to obey the specifications.**

***Index Terms*— Discrete approximation of continuous systems, hybrid control systems, timed supervisory control.**

I. Introduction

We consider a special class of hybrid control problems: the plant state evolves in $\mathbb{R}^n$ and is affected by real-valued unknown but bounded disturbances, whereas control input and measurement signals are discrete-valued or symbolic.[1] One is then concerned with finding an appropriate feedback structure mapping symbolic measurement signals into (sets of) symbolic control inputs. Such problems frequently arise in chemical process control: process models typically have a continuous state set but continuous *and* discrete control inputs and measurement signals. Often, there is also a clearly defined hierarchical structure where feedback loops from continuous measurement to continuous control variables are interpreted to be "low level" (dealing, for example, with set-point regulation), and discrete signals are used for "high level," or supervisory, control. Examples for the latter are start-up/shut-down procedures and the handling of "irregularities" (discrete alarm signals). In this paper, we will take plant model and continuous control loops as an entity (simply referred to as "the plant"), for which a discrete controller has to be designed.

How to approach such problems depends crucially on specifications: if specifications are purely in terms of continuous variables, one can adopt a "continuous point of view" (e.g., [10])—under certain conditions, a controller equipped with a continuous plant model can reconstruct the plant state from symbolic measurements and use it in a Receding Horizon scheme. Hence, the hybrid problem is solved within the "traditional" framework provided by continuous systems theory.

In this paper, we follow a complementary approach, which is based on the work by Antsaklis *et al.* (e.g., [1]), and Lunze (e.g., [5]): if specifications are in terms of symbolic variables, the continuous plant model can be approximated by a (nondeterministic) finite-state machine (FSM); this converts the hybrid control problem into a purely discrete one, which can subsequently be solved using tools from discrete-event systems (DES) theory (e.g., [13]). A key requisite in this approach is that the discrete-time behavior $\mathcal{B}_c$ of the underlying continuous model must be contained in the behavior[2] $\mathcal{B}_d$ of the discrete model. If the condition $\mathcal{B}_c \subseteq \mathcal{B}_d$ were violated, the continuous system could respond to a given input signal with an unacceptable output or measurement signal which would not be predictable by the discrete approximation. Hence, this unacceptable phenomenon could not be suppressed by a control strategy based on the discrete approximation—the approximation would be useless for the purposes of control systems design. In general, $\mathcal{B}_c$ is a *proper* subset of $\mathcal{B}_d$, and the smaller the difference $\mathcal{B}_d \setminus \mathcal{B}_c$, the more accurate the discrete approximation. In the behavioral context, dynamic feedback for a given system (plant model) essentially reduces to intersecting the system behavior with the controller behavior; the intersection, assuming it is nonempty, must be contained in the specification behavior. Hence, if $\mathcal{B}_d$ is "too big" (i.e., the approximation is too coarse), there might be no control scheme which enforces the specifications. This is where the original contributions suggesting discrete approximation for continuous models (e.g., [1] and [5]) might run into problems: they are based on *partitioning* $\mathbb{R}^n$ via the measurement map q_y: $\mathbb{R}^n \rightarrow Y_d$, where Y_d is a finite set of measurement symbols; all states which correspond to the same measurement symbol $y_d \in Y_d$ are "lumped"; the cosets of the measurement map can be interpreted as the state set of a discrete approximation model. Hence, the minimal achievable $\mathcal{B}_d$ which covers $\mathcal{B}_c$ (and hence the maximal accuracy of the discrete model) is completely determined by the measurement map q_y. If the resulting model is "too coarse" to allow design specifications to be met, one has "run out of luck" (unless one can change the measurement map). We, therefore, suggest a modified approximation scheme, which is characterized by the fact that the degree of accuracy can be adjusted to suit various specifications *without* changing measurement quantization. This is achieved by identifying subsets of $\mathbb{R}^n$ which are compatible with *strings* of measurement and control symbols. As, in this approach, the controller is "responsible" for high-level tasks, we adopt a supervisory control philosophy [13]: denote the finite set of control symbols by U_d; the controller does *not* map the sequence of past and present measurement symbols into U_d to determine the current control input, but into the power set 2^{U_d}, i.e., it disables a subset of U_d and leaves the decision which of the remaining control symbols is to be chosen to a lower level agent. The controller is considered optimal if it is least restrictive while guaranteeing that design specifications are met.

This contribution is based on previous work by the authors [11]. There, the continuous model was translated into a discrete

Manuscript received September 23, 1997. This work was supported by Deutsche Forschungsgemeinschaft under Grants Ra 516/2-2 and Ra 516/3-1, by "Sonderforschungsbereich" SFB 412, and by the Information Technology Research Centre, Ontario, Canada.

J. Raisch is with the Institut für Systemdynamik und Regelungstechnik, Universität Stuttgart, D-70550 Stuttgart, Germany (e-mail: raisch@isr.uni-stuttgart.de).

S. D. O'Young is with the Faculty of Engineering and Applied Science, Memorial University of Newfoundland, St. John's, Newfoundland, Canada A1B 3X5.

Publisher Item Identifier S 0018-9286(98)02807-4.

[1] This paper is set in a discrete-time framework, i.e., the domain of all signals is $\{t_0, t_1, \cdots\}$, with $t_i - t_{i-1}$ being constant. This understood, the adjectives "discrete" and "continuous" will in the sequel only be used to refer to the *codomain* of signals: the codomain of a discrete, or discrete-valued, signal is a set of symbols which, for our purposes, will be assumed to be finite (e.g., {"valve open," "valve closed"} or {"liquid level too low," "ok," "too high"}). The codomain of a continuous, or continuous-valued, signal gives the real numbers.

[2] Loosely speaking, the behavior of a model is the set of all time-trajectories of its external signals; for a survey on "behavioral systems theory," see [14] and the references therein.

decision tree, which was subsequently used to determine limited lookahead control policies. The approximation step employed in the present paper is described in more detail in [12]. Beside the references given above, the following papers deal with closely related topics; Pappas and Sastry [9] consider *continuous* abstractions of continuous dynamical systems. Caines and Wei [4] propose a lattice of dynamically consistent partition machines associated with a given continuous system. Niinomi *et al.* [6] design supervisory control on the basis of approximating automata; in contrast to our work, their approximations are purely logical, i.e., timing information (apart from the temporal order of events) is *not* retained.

This paper is organized as follows: In Section II, the continuous plant model is introduced. In Section III, it is shown how to approximate it by a nondeterministic FSM. Section IV treats the synthesis of an optimal supervisory controller for the resulting FSM model. In Section V, this controller is shown to force the continuous plant model to obey the specifications.

Finally, a short remark concerning notation: signals are represented by lower case letters, their codomains by the corresponding upper case letters. Discrete-valued signals are characterized by the subscript "d." For example, $u_d\colon \{t_0, t_1, \cdots\} \to U_d$ is the input signal, which is defined on the sampling grid $\{t_0, t_1, \cdots\}$ and "lives" (takes values) in the discrete set U_d.

II. The Continuous Plant Model

The plant is modeled as a discrete-time dynamic system:

$$x(t_{k+1}) = f(x(t_k), w(t_k), u_d(t_k)) \tag{1}$$
$$z_d(t_k) = q_z(x(t_k)) \tag{2}$$
$$y_d(t_k) = q_y(x(t_k)) \tag{3}$$

where $k \in \{0, 1, 2, \cdots\}$ is the time index, $x(t_k) \in \mathbb{R}^n$ the state at time t_k, and $w(t_k) \in \mathbb{R}^r$ an unknown but bounded disturbance: $w(t_k) \in W := \{w \mid w \in \mathbb{R}^r, \|w\|_\infty \le 1\}$, where $\|w\|_\infty = \max_i |w_i|$. $u_d(t_k) \in U_d$, $z_d(t_k) \in Z_d$, and $y_d(t_k) \in Y_d$ are control, output, and measurement symbols, respectively. U_d, Z_d, and Y_d are finite sets without any algebraic structure

$$U_d = \{u_d^{(1)}, \cdots, u_d^{(\alpha)}\}, \qquad Z_d = \{z_d^{(1)}, \cdots, z_d^{(\beta)}\}$$
$$Y_d = \{y_d^{(1)}, \cdots, y_d^{(\gamma)}\}.$$

Output symbols are used to specify desired performance.[3] The only requirement on $f\colon \mathbb{R}^n \times \mathbb{R}^r \times U_d \to \mathbb{R}^n$ is that its restriction to $\mathbb{R}^n$ be invertible. Therefore, (1) can be solved for the first argument on the right-hand side, and the solution is denoted by $x(t_k) = \tilde{f}^{-1}(x(t_{k+1}), w(t_k), u_d(t_k))$. Both output and measurement map, $q_z\colon \mathbb{R}^n \to Z_d$ and $q_y\colon \mathbb{R}^n \to Y_d$, are onto. They induce equivalence relations on $\mathbb{R}^n$; their cosets are referred to as z_d-cells and y_d-cells, respectively.

III. Nondeterministic Automata as Discrete Approximations

In this section, it will be shown how to generate a nondeterministic automaton as an approximation, or abstraction, for the continuous plant model. First of all, choose a nonnegative integer v—this will turn out to be a design parameter which determines accuracy of the discrete abstraction. Then, the state of the approximating automaton at time t_k is defined as

$$x_d(t_k) := \begin{cases} ([y_d(t_k), \cdots, y_d(t_0)], [u_d(t_{k-1}), \cdots, u_d(t_0)]), \\ \qquad \text{if } k = 0, 1, \cdots, v-1 \\ ([y_d(t_k), \cdots, y_d(t_{k-v})], [u_d(t_{k-1}), \cdots, u_d(t_{k-v})]), \\ \qquad \text{if } k \ge v. \end{cases} \tag{4}$$

Hence, the current state of the approximation consists of a recorded string of measurements and control symbols reaching back over a certain interval $\min(k, v)$. This choice is reminiscent of observer canonical realizations in (continuous) control theory—the state is made up of known quantities and hence is trivially observable. As both U_d, the set of control symbols, and Y_d, the set of measurement symbols, are finite, the number of different strings (4) one gets by exhaustive permutation of measurement and control symbols is also finite.

It is obvious, however, that the continuous system (1), (3) cannot generate *all* these strings. In the next step, one therefore eliminates all strings which are not compatible with the continuous model (1), (3) and the disturbance assumptions: let $0 \le \rho \le v$ and denote $\tilde{f}^{-1}(x, w, u_d^{(k)})$ by $\tilde{f}_k^{-1}(x, w)$. Then, $([y_d^{(i_0)}, \cdots, y_d^{(i_\rho)}], [u_d^{(k_1)}, \cdots, u_d^{(k_\rho)}])$ is an element in the state set of the discrete abstraction if and only if

$$y_d^{(i_0)} = q_y(x) \tag{5}$$
$$y_d^{(i_1)} = q_y(\tilde{f}_{k_1}^{-1}(x, w_{j_1})) \tag{6}$$
$$\vdots$$
$$y_d^{(i_\rho)} = q_y(\tilde{f}_{k_\rho}^{-1} \cdots (\tilde{f}_{k_1}^{-1}(x, w_{j_1}), \cdots, w_{j_\rho})) \tag{7}$$

subject to

$$\|w_{j_l}\|_\infty \le 1, \qquad l = 1, \cdots, \rho \tag{8}$$

has a nonempty solution set for $[x', w_{j_1}', \cdots, w_{j_\rho}']'$. This "compatibility check" becomes a numerically straightforward and reliable procedure, if the right-hand side of (1) is affine in the state $x(t_k)$ and the unknown disturbance vector $w(t_k)$, and the measurement map has the form $q_y = Q_y \circ C_y$, where $C_y\colon \mathbb{R}^n \to \mathbb{R}^p$ is a linear map and the "quantizer" $Q_y\colon \mathbb{R}^p \to Y_d$ partitions $\mathbb{R}^p$ into finitely many rectangular boxes with edges parallel to the coordinate axes. In this case, (5)–(8) reduces to a set of *linear inequalities* in $[x', w_{j1}', \cdots, w_{j\rho}']'$. Details can be found in [12].

This second step boils down to "weeding out" nonreachable states; the remaining state structure is therefore guaranteed to be minimal. The number of elements in this minimal state set X_d is denoted by N_v: $X_d := \{x_d^{(1)}, \cdots, x_d^{(N_v)}\}$. The state $x_d^{(i)}$ belongs to the set X_{d0} of possible initial states, if it contains exactly one measurement symbol (and no control symbols), i.e., $X_{d0} = Y_d$. This reflects the fact that, prior to time t_0, we do not have any control over the plant. Hence, $x(t_k)$ can be anywhere in $\mathbb{R}^n$, and any measurement symbol in Y_d is possible at time t_0.

Denote the strings of control and measurement symbols associated with a particular $x_d^{(i)}$ by $u_d^*(x_d^{(i)})$ and $y_d^*(x_d^{(i)})$, respectively, and introduce a "forgetting operator" $\mathcal{F}$ which deletes the "oldest" symbol from strings $y_d^*(x_d(t_k))$ and $u_d^*(x_d(t_k))$, if $k \ge v$

$$\mathcal{F}(y_d^*(x_d(t_k))) := \begin{cases} [y_d(t_k), \cdots, y_d(t_0)], & \text{if } k = 0, 1, \cdots, v-1 \\ [y_d(t_k), \cdots, y_d(t_{k-v+1})], & \text{if } k \ge v \end{cases}$$
$$\mathcal{F}(u_d^*(x_d(t_k))) := \begin{cases} [u_d(t_{k-1}), \cdots, u_d(t_0)], & \text{if } k = 0, 1, \cdots, v-1 \\ [u_d(t_{k-1}), \cdots, u_d(t_{k-v+1})], & \text{if } k \ge v. \end{cases}$$

[3] Clearly, the choice of the outputs is part of the design process. For example, if safety is our only concern, the codomain of the output map q_z is $Z_d = \{\texttt{forbidden}, \texttt{legal}\}$.

Now, writing down the transition structure of the discrete approximation is trivial; $(x_d^{(i)},\ u_d^{(j)},\ x_d^{(k)})$ is a transition iff there exists a $y_d^{(l)} \in Y_d$ such that

$$y_d^*(x_d^{(k)}) = [y_d^{(l)},\ \mathcal{F}(y_d^*(x_d^{(i)}))] \tag{9}$$

and

$$u_d^*(x_d^{(k)}) = [u_d^{(j)},\ \mathcal{F}(u_d^*(x_d^{(i)}))]. \tag{10}$$

$x_d^{(i)}$ and $x_d^{(k)}$ are called the exit state and the entrance state of the transition; the control symbol $u_d^{(j)}$ is its transition label. Each state $x_d^{(i)}$ has an associated unique (measured) output, which is simply the leftmost symbol in $y_d^*(x_d^{(i)})$. Hence, for each $v \geq 0$, we get a finite Moore automaton as a plant approximation. For a given $x_d^{(i)}$ and $u_d^{(j)}$, more than one $y_d^{(l)}$ (hence more than one $x_d^{(k)}$) may satisfy (9) and (10); the resulting automaton will be nondeterministic—this is further illustrated below.

By construction, $x_d^{(j)}$ is an element in X_d if and only if the set of solutions $[x',\ w'_{j_1},\ \cdots,\ w'_{j_\rho}]'$ for (5)–(8) is nonempty. The projection of this set onto its first n elements can be interpreted as the set of continuous plant states x which are compatible with the automaton state $x_d^{(j)} = ([y_d^{(i_0)},\ \cdots,\ y_d^{(i_\rho)}],\ [u_d^{(k_1)},\ \cdots,\ u_d^{(k_\rho)}])$—this set is denoted by $X(x_d^{(j)})$. In other words, if the approximating automaton is in state $x_d^{(j)}$, $j = 1, \cdots, N_v$, we know that the state of the underlying continuous plant is in the set $X(x_d^{(j)})$—the set $X(x_d^{(j)})$ can be interpreted as a set-valued estimate for the continuous plant state. Clearly, $\bigcup_{j=1}^{N_v} X(x_d^{(j)}) = \mathbb{R}^n$, and, in general, $X(x_d^{(i)}) \cap X(x_d^{(j)}) \neq \emptyset$. Hence, the sets $X(x_d^{(j)})$ form a cover for $\mathbb{R}^n$. Obviously, a set $X(x_d^{(j)})$ never increases (and, in general, decreases) in "size" if the input and measurement strings embodied in $x_d^{(j)}$ are extended further into the past—this is an immediate consequence of the "triangular" structure of (5)–(8). Increasing v, the maximum length of strings, is therefore equivalent to generating a finer "granularity" for the finite cover of $\mathbb{R}^n$—the nonnegative integer v can be seen as a design parameter, which may be used to improve the accuracy of the discrete model. This of course implies that the number of states, and hence the complexity of the discrete model, also increases. This mental picture is also helpful for understanding the intrinsic nondeterminism of discrete abstractions: a state $x_d(t_k) = x_d^{(i)}$ of the discrete abstraction corresponds to a set $X(x_d^{(i)}) \subset \mathbb{R}^n$. An input $u_d(t_k) = u_d^{(j)}$ then maps this set onto another set. Only under very restrictive assumptions will the latter be contained in exactly one $X(x_d^{(k)})$ and not intersect any $X(x_d^{(l)})$, $l \neq k$, i.e., only in exceptional cases will the discrete state $x_d(t_k) = x_d^{(i)}$ have a unique successor state $x_d(t_{k+1}) = x_d^{(k)}$ under input $u_d^{(j)}$. Existence of unknown disturbances in the "base" model merely increases the "level of nondeterminism" in its discrete abstraction.

As controller synthesis is to be based on the discrete approximation, we also have to provide it with z_d-outputs. But this is easy; simply define a symbol $z_d^{(j)}$ to be an output of state $x_d^{(i)}$, if $X(x_d^{(i)})$ intersects the z_d-cell $\{\zeta \in \mathbb{R}^n | z_d^{(j)} = q_z(\zeta)\}$. Note that $X(x_d^{(i)})$ can intersect several z_d-cells, hence a state $x_d^{(i)}$ can be associated with a (nonempty) *set* of possible output symbols $z_d^{(j)}$.

The Moore automaton evolves on the same sampling grid $T = \{t_0, t_1, \cdots\}$ as the underlying continuous system: the time needed for a transition is the sampling interval $t_{i+1} - t_i$. Hence, an (implicit) notion of time is retained in our discrete approximation. Denote the set of all functions from T into $(U_d \times Y_d \times Z_d)$ by $(U_d \times Y_d \times Z_d)^T$. The behaviors $\mathcal{B}_c \subseteq (U_d \times Y_d \times Z_d)^T$ and $\mathcal{B}_d(v) \subseteq (U_d \times Y_d \times Z_d)^T$ are the sets of control/measurement/output signals which are compatible with the continuous model (1)–(3) and its discrete abstraction. The following key result has been proven in [12] and formalizes our previous discussion of approximation accuracy.

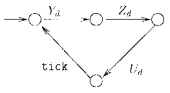

Fig. 1. Clock process.

Theorem 1: The behavior of the continuous plant model (1)–(3) and the behaviors of the discrete abstractions obtained for $v = 0, 1, \cdots$ form a totally ordered set, where ordering is in the sense of set inclusion: $\mathcal{B}_c \subseteq \mathcal{B}_d(v_i) \subseteq \mathcal{B}_d(v_j);\ v_i, v_j = 0, 1, \cdots;\ v_i \geq v_j$.

Hence, as required, the approximation behavior contains the continuous model behavior for any nonnegative v. Moreover, as expected, approximation accuracy is monotone in v.

Finally, for the purposes of control systems design, we convert the Moore automaton into an equivalent[4] transition structure without state outputs. This is done by adding a Y_d-labeled selfloop to each state and one or more Z_d-labeled selfloops (depending on the number of z_d-symbols attached to the state). To ensure "correct" ordering of transitions, we compose this transition structure with the simple clock process shown in Fig. 1. This also makes time explicit—the `tick` event represents the passage of one sampling interval and can be thought of as being synchronized with an external clock. The arrows labeled by U_d, Z_d, and Y_d represent the occurrence of any symbol from U_d, Z_d, and Y_d, respectively. The resulting nondeterministic transition structure is referred to as a *discrete-time discrete-event process (DTP)*. It includes explicit timing (`tick`-events) and a control mechanism (transitions labeled by a control symbol can be disabled). Its behavior is defined as follows: count the `tick`-events; denote transitions occurring before the first `tick` by $u_d(t_0),\ y_d(t_0),\ z_d(t_0)$, transitions occurring between the first and the second `tick` by $u_d(t_1),\ y_d(t_1),\ z_d(t_1)$ etc.; in this way, the DTP generates a behavior identical to the one of the underlying Moore automaton.

IV. Supervisory Control for the Approximating Automaton

We adopt a supervisory control philosophy [13] to suitably modify the behavior of the discrete plant model. We use the state transition-based framework suggested in [7] which is well suited to deal with nondeterminism in the discrete plant model and which can also handle time.[5]

The departure point is a discrete approximation of the plant model in the form of a DTP. The mechanism of control is via a subset of transitions that can be disabled (prevented from happening). These are the transitions labeled by a control symbol. Specifications are also formulated as DTP's. The most straightforward example is the case where one is only concerned with safety: then, the codomain of the output map q_z is chosen as $Z_d = \{$`forbidden`, `legal`$\}$, and the specification DTP simply states that after each `tick`, $z_d =$ `legal` has to occur. It is obvious how more complex dynamic specifications can be coded as DTP's. Forming the parallel composition $P = M \parallel S$ of the plant DTP, M, and the specification DTP, S, formally removes all transitions which violate the specifications—but this is done without caring for realizability. For example, a transition can only be eliminated if it is labeled by a control symbol; a transition cannot be eliminated if this implies that the process can reach a state where no further `tick`-event can be executed—stopping time is impossible. The optimal supervisor's job can then be thought of

[4] In the sense of having the same behavior as the Moore automaton.

[5] An extension of [13] to (deterministic) DES with timing features is described in [3].

as implementing the "least restrictive," but realizable substructure of P. This is formalized in the following paragraph.

Let Q and Σ be the (finite) sets of states and event (or transition) labels of P. A transition is represented by a triplet (q_e, σ, q_i), with $q_e, q_i \in Q$, and $\sigma \in \Sigma$. q_e and q_i are called *exit* and *entrance state*; σ is the *event label*. Transitions $(q_{e1}, \sigma_1, q_{i1})$ and $(q_{e2}, \sigma_2, q_{i2})$ are called *partners*, if $q_{e1} = q_{e2}$ and $\sigma_1 = \sigma_2$ (they have the same exit state and event label). A DTP can be represented by a pair (Δ, Q_0) where Δ and Q_0 are the (finite) sets of transitions and initial states, respectively.

Definition 1: Let $P = (\Delta, Q_0)$ be a DTP. The DTP $\tilde{P}$ modeled by the pair $(\tilde{\Delta}, \tilde{Q}_0)$ is called a *discrete-time discrete-event subprocess* (DSP) of P (denoted by $\tilde{P} \subseteq P$), if $\tilde{\Delta} \subseteq \Delta$, $\tilde{Q}_0 \subseteq Q_0$, and a transition $\delta \in \Delta$ can only be an element in $\tilde{\Delta}$, if all its partners are also contained in $\tilde{\Delta}$.

A state $q_2 \in Q$ is *reachable* from a state $q_1 \in Q$ (or, equivalently, q_1 is coreachable from q_2) if there is a sequence of transitions from Δ connecting q_1 with q_2. A DTP P is called reachable if every element of its state set is reachable from an initial state.

Definition 2: Let $P = M \| S$ and U_d be the set of transition labels of M (and hence P) which can be disabled by a control agent. Let $\tilde{P} = (\tilde{\Delta}, \tilde{Q}_0)$ be a reachable DSP of P with state set $\tilde{Q}$ and with $\tilde{Q}_0 = Q_0$. Then, $\tilde{P}$ is said to be *controllable w.r.t. to* M if $(q_{Me}, \sigma, q_{Mi}) \in \Delta_M$ (the transition set of M), $(q_{Me}, q_{Se}) \in \tilde{Q}$, and $((q_{Me}, q_{Se}), \sigma, -) \notin \tilde{\Delta}$ implies that $\sigma \in U_d$ ($-$ means "don't care").

Clearly, a DSP $\tilde{P}$ of P can only be realized by a controller if it is controllable w.r.t. M. Another condition for realizability is that the progress of time can never be stopped.

Definition 3: A state $q_1 \in Q$ is said to be `tick`-coreachable if there exists a state $q_2 \in Q$ where a `tick`-transition can occur and which is reachable from q_1. The DTP P is called `tick`-coreachable or *temporally nonblocking*, if every state in Q is `tick`-coreachable.

Let $\tilde{P}_1 = (\tilde{\Delta}^1, \tilde{Q}_0^1)$ and $\tilde{P}_2 = (\tilde{\Delta}^2, \tilde{Q}_0^2)$ be two DSP's of P. Then the *union* of $\tilde{P}_1$ and $\tilde{P}_2$, denoted by $\tilde{P}_1 \cup \tilde{P}_2$, is a transition structure represented by the pair $(\tilde{\Delta}^1 \cup \tilde{\Delta}^2, \tilde{Q}_0^1 \cup \tilde{Q}_0^2)$. It is immediately clear that $\tilde{P}_1 \cup \tilde{P}_2$ is another DSP of P. The relation $\subseteq$ induces a partial ordering on the set of all DSP's of P. Let $\{\tilde{P}_{CN}\}$ be the set of all DSP's of P which are controllable w.r.t M and temporally nonblocking. In [8], it has been shown that $\{\tilde{P}_{CN}\}$ is closed under union. Hence, if nonempty, $\{\tilde{P}_{CN}\}$ forms an upper-semilattice (with the join operation being $\cup$). Clearly, $\{\tilde{P}_{CN}\}$ is finite. Therefore, the following holds.

Theorem 2: If $\{\tilde{P}_{CN}\}$ is nonempty, there exists a (unique) greatest DSP of P (w.r.t. the ordering via $\subseteq$) which is controllable w.r.t. M and temporally nonblocking.

If $\{\tilde{P}_{CN}\}$ is nonempty, denote its supremal element by $\tilde{P}_S$. It can be interpreted as the transition structure of P that survives under the least restrictive realizable supervisory control policy which guarantees the specifications to be met. $\tilde{P}_S$ (and hence the least restrictive control strategy) can be synthesized formally in a computer-aided design environment. If $\{\tilde{P}_{CN}\}$ is empty, the supervisory control problem has no solution. This implies that either the approximating automaton is too coarse (we need to provide a finer approximation by increasing the parameter v), or the specifications are too strict (they cannot be met no matter how accurate our approximation is) and need to be relaxed.

V. The Continuous Plant Under Discrete Control

Now, suppose we have come up with a supervisory control scheme that solves the discrete feedback problem. Does the continuous "base" model, when subject to the *same* control scheme, also satisfy the specifications? Or, in other words, does it make sense to base the design of symbolic feedback controllers for a continuous plant on a discrete approximation? This is indeed the case: denote the behaviors of the controlled continuous plant model and the controlled discrete approximations for $v = 0, 1, \cdots$, by $\mathcal{B}_{cs}$ and $\mathcal{B}_{ds}(v)$, respectively (the subscript s signifies supervision). As, in the behavioral context, the meaning of control essentially reduces to intersecting the plant model behavior and the controller behavior, the following is an immediate consequence from Theorem 1:

$$\mathcal{B}_{cs} \subseteq \mathcal{B}_{ds}(v_i) \subseteq \mathcal{B}_{ds}(v_j), \qquad v_i, v_j = 0, 1, \cdots; \ v_i \geq v_j. \tag{11}$$

Hence, $\mathcal{B}_{ds}(v) \subseteq \mathcal{B}_{\text{spez}}$—the discrete control system satisfies the specifications (for any nonnegative v)—implies $\mathcal{B}_{cs} \subseteq \mathcal{B}_{\text{spez}}$. It remains to show that $\mathcal{B}_{cs} \neq \emptyset$, i.e., the discrete controller can actually be connected to the continuous plant model (or, in the DES terminology, does not cause it to block). All that is required for this is that the controller never disables *all* control symbols simultaneously. But this is an immediate consequence from the temporal nonblocking property in the discrete case.

VI. Conclusions

We have proposed an approach to hybrid control systems design which is based on approximating the continuous plant model by a nondeterministic FSM. Two important features of the approximation procedure are: 1) it allows adjusting the accuracy of the discrete abstraction to the specification requirements and 2) the approximating automaton "lives" on the same sampling grid as the underlying continuous plant model and hence retains a notion of time. This approach "translates" the hybrid control problem into a purely discrete problem. Given the discrete approximation, the least restrictive control scheme that forces a given (discrete) set of specifications to hold can then be formally synthesized. If there is no solution, either the approximating plant automaton is too coarse (i.e., we need to increase approximation accuracy) or the specifications are too strict (for any approximation) and hence have to be relaxed. If one succeeds in solving the discrete control problem (i.e., one finds an appropriate supervisor), one has also solved the underlying hybrid control problem: the continuous plant model under discrete control is also guaranteed to obey the specifications.

References

[1] P. J. Antsaklis, J. A. Stiver, and M. Lemmon, "Hybrid system modeling and autonomous control systems," in *Hybrid Systems*, Lecture Notes in Computer Science, vol. 736, R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, Eds. New York: Springer-Verlag, 1993, pp. 366–392.

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

[3] B. A. Brandin and W. M. Wonham, "Supervisory control of timed discrete event systems," *IEEE Trans. Automat. Contr.*, vol. 39, pp. 329–342, 1994.

[4] P. E. Caines and Y.-J. Wei, "On dynamically consistent hybrid systems," in P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, Eds., *Hybrid Systems II*, Lecture Notes in Computer Science, vol. 999. New York: Springer-Verlag, 1995, pp. 86–105.

[5] J. Lunze, "Qualitative modeling of linear dynamical systems with quantized state measurements," *Automatica*, vol. 30, pp. 417–431, 1994.

[6] T. Niinomi, B. H. Krogh, and J. E. R. Cury, "Synthesis of supervisory controllers for hybrid systems based on approximating automata," in *Proc. Conf. Decision Contr.*, 1995, pp. 1461–1466.

[7] S. D. O'Young, "On the synthesis of supervisors for timed discrete event processes," *IEEE Trans. Automat. Contr.*, to be published.

[8] S. D. O'Young and J. Raisch, "Timed DES supervisory control of hybrid systems," in *Proc. Workshop on Discrete Event Systems*, Edinburgh, U.K., 1996, pp. 189–194.

[9] G. J. Pappas and S. Sastry, "Toward continuous abstractions of dynamical and control systems," Univ. California, Berkeley, Tech. Rep. UCB/ERL M96/53, Oct. 1996.
[10] J. Raisch, "Control of continuous plants by symbolic output feedback," in P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, Eds., *Hybrid Systems II*, Lecture Notes in Computer Science, vol. 999. New York: Springer-Verlag, 1995, pp. 370–390.
[11] J. Raisch and S. D. O'Young, "A DES approach to control of hybrid dynamical systems," in *Hybrid Systems III*, Lecture Notes in Computer Science, vol. 1066, R. Alur, T. A. Henzinger, and E. D. Sontag, Eds. New York: Springer-Verlag, 1996, pp. 563–574.
[12] ____, "A totally ordered set of discrete abstractions for a given hybrid or continuous system," in *Hybrid Systems IV*, Lecture Notes in Computer Science, vol. 1273, P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, Eds. New York: Springer, 1997, pp. 342–360.
[13] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event systems," *SIAM J. Contr. Optimiz.*, vol. 25, pp. 206–230 1987.
[14] J. C. Willems, "Paradigms and puzzles in the theory of dynamical systems," *IEEE Trans. Automat. Contr.*, vol. 36, pp. 259–294, 1991.

Differential Petri Nets: Representing Continuous Systems in a Discrete-Event World

Isabel Demongodin and Nick T. Koussoulas

***Abstract*— Differential Petri nets are a new extension of Petri nets. Through the introduction of the differential place, the differential transition, and suitable evolution rules, it is possible to model concurrently discrete-event processes and continuous-time dynamic processes, represented by systems of linear ordinary differential equations. This model can contribute to the performance analysis and design of industrial supervisory control systems and of hybrid control systems in general.**

***Index Terms*—Differential Petri nets, discrete-event dynamic systems, hybrid systems, Petri nets, supervisory control systems.**

I. Introduction

One of the most recent and most intense efforts in control theory deals with handling dynamic systems that include not only the technological process but its supervisory mechanism(s) as well. For our purposes, hybrid systems are considered to be all combinations of a continuous plant (such as a chemical process) or a mixed continuous/discrete-event plant (such as a chemical process with process-related logic), with a discrete-event supervisor that reacts to external events (planned or unforeseen). Thus, a supervisory control system of the classical hierarchically structured form of three levels (execution, supervision, and coordination) falls into this description. This kind of control system, being a mixture of continuous-time and discrete-event dynamic processes, has been termed "hybrid" or "discontinuous." Modeling, analysis, control, and synthesis of such systems pose a number of challenging problems.

Manuscript received September 23, 1997. This work was supported in part by the European Commission through the ESPRIT-8924 program SESDIP.

I. Demongodin is with the Department of Automatic Control and Production Systems, Ecole des Mines de Nantes, La Chantrerie, B.P. 20722, 44 307 Nantes, Cedex 03, France.

N. T. Koussoulas is with the Laboratory for Automation and Robotics, Electrical and Computer Engineering Department, University of Patras, 26500 Rio, Patras, Greece (e-mail: ntk@ee.upatras.gr).

Publisher Item Identifier S 0018-9286(98)02780-9.

In this initiatory phase of hybrid control systems theory, most of the discussions focus on the issue of suitable modeling approaches to the representation of hybrid systems. Different approaches of modeling have been used, and there is already an abundance of models [16], [7]. Some authors (e.g., [4]) define a homogeneous model which links the discrete-event part and the continuous part in a single formalism. Others (e.g., [8], [1], and [17]) use specific formalisms for each of the two parts or define a model based on the interface between the two parts. Perhaps naturally, most of the efforts are based on the discrete-event part and involve models for discrete-event dynamic systems (DEDS's) such as finite state machines, process algebras, Petri nets, temporal logic, etc. A nice overview, along with a first attempt for a unified model, appeared in [7].

Among the DEDS models, Petri nets proved to be a very popular model with the academic and industrial community alike. Having a dual nature of a graphical tool and a mathematical object can serve in both the practical and the theoretical camp. There is a constantly growing number of publications regarding Petri nets and their applications which can be found in quite diverse fields. A major step in the effort to enlarge the modeling power of Petri nets has been their extension known as continuous Petri nets. The motivation was the inability for successful modeling of discrete-event systems with a large number of mostly unobservable events (e.g., circulation of bottles in a bottling line). Thus, a continuous approximation was proposed instead, replacing the uncertain counting with a speed approximation, a technique quite popular in similar settings, such as queuing networks. Continuous Petri nets are thus approximations to discrete-event systems allowing faster simulation of the latter without sacrificing accuracy. The combination of continuous with ordinary Petri nets leads to the concept of hybrid Petri nets [2], where hybridity does not refer to the kind indicated above.

It is advantageous, if not indispensable, to be able to represent both continuous and discrete parts of a hybrid system in the same context. This can be problematic, however, due to the mathematical incompatibility between instantaneous events and the convenience of continuity. Given that, it appears less cumbersome to choose the discrete-event domain as the environment for this common representation. Therefore, an extension for Petri nets is necessary to allow them to represent the continuous time dynamic components. Bourjij *et al.* [6] show that it is possible to model a hybrid Petri net in a singular system, by using Euler approximation. Their approach permits us to perform diagnostics through a reference model established by an extension of a hybrid Petri net. However, this consideration does not take into account the possibility of negative values for continuous variables. To represent negative values, Saadi *et al.* [19] have developed another extension of continuous Petri nets in which a continuous place can support a negative marking. They call this extension a dynamical continuous Petri Net and merge this kind of Petri net with a regular one as defined in the hybrid Petri net framework. In [5], the simulation of ordinary differential equations is represented with predicate/transition Petri nets. While they consider only Euler integration because their work seems to be focused on real-time applications, it is in principle valid for other integration algorithms too. In a similar vein, in [15], to arrive at a unified model for the hybrid system, a Petri net equivalent to the causal graph of the continuous system is found. However, the issues of interface with the "logical" part and the issue of negative markings are not covered in both works.