INT. J. CONTROL, 1996, VOL. 63, NO. 4, 767–797

State-space supervisory control of reconfigurable discrete event systems

HUMBERTO E. GARCIA† and ASOK RAY‡

The discrete event theory of supervisory and state feedback control offers many advatages for implementing supervisory systems. Algorithmic concepts have been introduced to ensure that the supervising algorithms are correct and meet the specifications. In the current methodology, it is, in general, assumed that the supervisory specifications are invariant during the operation of the system or, at least, until a given supervisory task is completed. However, there are many practical applications where the supervising specifications need to be updated in real time. For example, when dealing with complex processes, the tasks of supervisory systems analysis and synthesis can be facilitated by partitioning the controlled Discrete-Event System (DES) into several subprocesses. This partitioning is based on operational or physical considerations and a unique supervisor is assigned to control each subprocess at a given instant of time. When a decision maker at a higher level of hierarchy decides to change the supervising algorithm, switching to a new supervisor takes place. For this adaptive implementation, the decision-maker or *coordinator* first decides the set of acting supervisors based on the requested supervisory tasks or current system performance requirements and then exercises control over the enabled supervisors in real time. Specifically, in a Reconfigurable Discrete Event System (RDES) architecture, a bank of supervisors is defined to accommodate each identified operational condition or different supervisory specifications. This adaptive supervisory control system can change its supervisory configuration to accept coordinator commands or to adjust for changes in the controlled process. This paper addresses reconfiguration at the supervisory level of hybrid systems along with the underlying architecture of RDES. In particular, the paper reviews the supervisory control theory in the state-based framework and extends it to the paradigm of RDES, considering process control applications. The paper addresses theoretical issues with a limited number of practical examples. This control approach is particularly suitable for hierarchical hybrid implementations with the capability of reconfiguration at both the control and supervisory levels.

1. Introduction

Traditionally, control theory has been applied to systems whose dynamic behaviour can be modelled by difference or differential equations. These systems evolve continuously in time and satisfy the basic principles of physics. However, modern technology has created man-made dynamic systems with no invariant physical laws to constrain their configurations, and these systems may not be described by ordinary or partial differential/difference equations. Specifically, the states of such systems have logical or symbolic rather than numerical values, and advance at discrete, unpredictable, irregular intervals, where the notion of time is replaced by event sequences. The occurrence of physical discrete events originates a dynamic trajectory that is piecewise constant and event driven (Ramadge and Wonham 1989).

Received 25 January 1994. Revised 7 November 1994.

† Argonne National Laboratory, P.O. Box 2528, Idaho Falls, ID 83403-2528, U.S.A. Tel: +(1) 208 533 7769; Fax: +(1) 208 533 7471; e-mail: garcia@anl.gov.

‡ Mechanical Engineering Department, Pennsylvania State University, University Park, PA 16802, U.S.A.

The events describe the system behaviour and represent control decisions or changes in the physical state of the process. Dynamic discrete systems described by events are called discrete event systems (DES). A comparison between the dynamics of continuous variable and discrete event systems can be found in Cao (1989). This paper focuses on the control of a class of DES.

In many control applications, the supervisory system is often required to be adaptive and flexible enough to accommodate time-varying supervisory specifications or persistently changing environments. Discrete event implementations with time-varying specifications or unknown disturbance are here loosely defined as time-varying DES. For offline design, a supervisor is required to achieve a closed-loop behaviour based on given specifications or an assumed system response. Adaptive behaviour can be incorporated by using learning algorithms (Narendra and Thathachar 1989) to evaluate and define these modifications. Adaptability is also desired in those cases where the dynamics of a given plant are partially known. In general, to deal with model uncertainties, two approaches can be mentioned (Lin 1992), namely, robust supervision and adaptive supervision. In robust supervision, a given supervisor is designed so that it can perform adequately in the event of identified uncertainties. However, uncertainties are not resolved. On the other hand, an adaptive supervisory system resolves uncertainties by identifying the plant conditions and updating the control algorithms accordingly. Among the growing literature on DES, reported work in uncertain and/or time-varying systems has been rather limited. Recently, the concepts of robust and adaptive supervision have been introduced (Lin 1992) under the Ramadge–Wonham framework (Ramadge and Wonham 1987, 1989, Wonham and Ramadge 1987) to deal with time-varying systems. However, this technique may not be computationally feasible in some practical cases, such as large complex processes or under partial observation. In Chung *et al.* (1993), a supervisory control scheme based on limited look-ahead control is described. In this online scheme, the next control action is determined based on the projection of the process behaviour. Procedures to perform this calculation are given by Chung *et al.* (1992, 1993). However, this scheme may not be feasible in some complex, real-time applications, especially, under partial observation. To deal with these problems partially, a definition of stability in the sense of Lyapunov is introduced for logical DES in Passino *et al.* (1991). Although this approach offers low computational complexity, the major difficulty lies in finding the Lyapunov functions (similar to the classical control theory). In Passino and Michel (1992), the notions of uniform boundedness, stability and finite time stability are extended for the case of DES defined on a metric space.

For supervision of complex dynamic processes, the approach taken in this paper is based on the principle of divide-and-conquer. Specifically, the controlled DES along with its operating conditions is partitioned into subprocesses and operating regimes, and a supervisor is devised for each pair of subprocesses and their operating condition. In general, more than one supervisor may be synthesized for a given subprocess (Garcia 1993, Garcia and Ray 1992). However, because a process is in exactly one operating condition at a given instant of time, one and only one supervisor for each subprocess is operational at that instant. As time evolves, based on the current operating condition, a supervisor is identified among the available ones to act directly on the plant. Changes in supervisors may also result from time-varying supervisory specifications. The class of discrete event systems that can reconfigure its supervisory algorithms, called Reconfigurable Discrete Event Systems (RDES), is investigated in this paper. In the RDES architecture, a bank of supervisors is defined for each controlled DES and supervisors are designed for each identified operational condition. When a decision maker at a higher level in the hierarchy decides to change the supervising algorithm, switching to a new supervisor takes place. To this effect, a co-ordinator is defined as a decision-making entity having two main responsibilities. First, it decides the set of active supervisors based on the current system performance or specifications. Secondly, it exercises control over the enabled supervisors in real time. In the extended context of hybrid systems (Gollu and Varaiya 1989), reconfiguration at both the control and supervisory levels results in reconfigurable hierarchical hybrid supervisory control systems, as introduced by (Garcia 1993, Garcia and Edwards 1993a, Garcia and Ray 1992, Garcia *et al.* 1995).

Reconfiguration requires diagnostic and decision-making algorithms. Diagnostic or performance evaluation routines such as those used by Garcia and Edwards (1993b), Garcia *et al.* (1991) may be utilized to identify current operating conditions. Subsequently, decision-making algorithms (e.g. Lapin 1985) select the set of operational supervisors for a given time period. Examples are given by Garcia (1993), Garcia and Edwards (1993a, 1993b), Garcia and Ray (1992), Garcia *et al.* (1990, 1991a, 1991b, 1995). This paper focuses on the issues for the underlying RDES architecture created by the proposed reconfigurable supervision scheme. To this end, a review of basic ideas of supervisory control in the state-based framework is provided with modelling aspects introduced considering process control applications. They are then applied to furnish analytical guidelines related to the implementation of RDES. However, limited examples are given here. It is then assumed that the particular application requiring a RDES theory has been identified and that the techniques for plant identification, asserting supervisor performance and supervisor selection have already been devised for the given application. To partially provide documentation for the feasibility of the ideas presented here, the following section and Garcia *et al.* (1995) contain instances for using the RDES paradigm when designing reconfigurable supervisory systems. However, it is expected that the paper can also be useful to practitioners in other potential application domains.

The organization of the paper is as follows. After presenting an application of the RDES concept, the framework used to describe DES is given in § 3 and the statement of the problem and its solution is given in § 4. A re-configurable supervisory architecture is then described in § 5 while the problems resulting from this supervisory adaptability are postulated in § 6. In § 7, the supervisory tasks required in a reconfigurable configuration are presented, and their solutions are derived in § 8. In § 9, topics on convergence among predicates are discussed. Section 10 suggests a probabilistic setting for specifying supervisory tasks. Finally, § 11 summarizes and concludes the paper.

2. Engineering applications of reconfigurable discrete event systems

The concept of reconfigurable discrete event systems (RDES) is applicable to diverse disciplines including command and control (C^2) of battlefield management, computer-integrated manufacturing (CIM), and process control. Specifically, RDES is well suited to describing and synthesizing supervisory algorithms for decision and control of complex dynamic processes such as power plants. In such cases, the decision and control system must have adaptive behaviour to ensure desired plant operations under uncertain or unfamiliar situations. Therefore, different control policies should be available to manoeuvre the plant under different operating conditions, and the

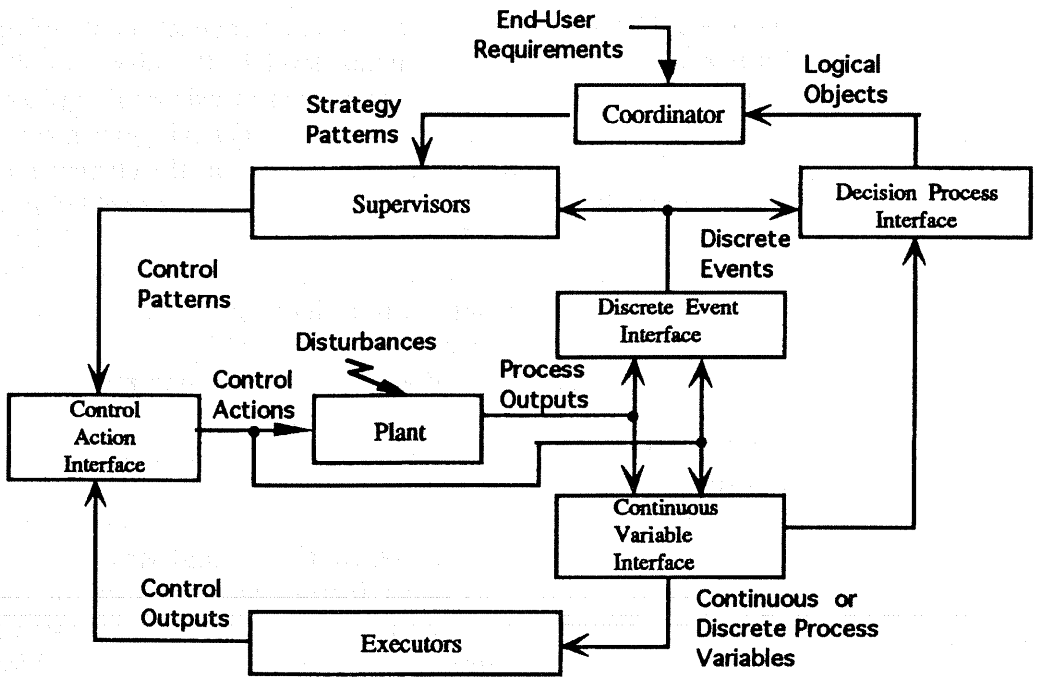

Figure 1. A reconfigurable supervisory control system.

decision and control system should be capable of supervising the operation of these control modules to achieve the mission objectives. In this respect, Garcia (1993), Garcia *et al.* (1995) have proposed a supervisory control system (SCS) architecture and evaluated the efficacy of this control concept via an in-plant test of a nuclear reactor.

The proposed SCS, shown in Fig. 1, is hierarchically structured into three levels, namely, the execution, supervision, and coordination levels. The execution level generates direct control actions over the process (or subprocesses). The entities responsible for implementing the control policies are named *executors*. In general, an executor is capable of exercising more than one control algorithm. These executors are governed by *supervisors* at the supervision level. The supervisors assign *control patterns* to the executors to select the control policies used on the plant. At the highest layer of the hierarchical structure, the coordination level overlooks the operations of the supervisors. An entity called the *coordinator* is defined to select appropriate supervisors dynamically and assign *strategy patterns* to them to ensure that the processes behaves in a desirable manner.

Notice then that the described SCS introduces reconfiguration at both execution and the supervision levels. A reconfigurable strategy is a dynamic scheme that identifies the current operating status of the controlled environment, evaluates control performances by dynamically monitoring process behaviour, and then selects the appropriate system configuration. In particular, reconfiguration at the execution is used because there may not exist a single control algorithm that can satisfy the performance criteria for the complete spectrum of process operating conditions. Reconfiguration at the supervisory level is used because a single 'overly complex' supervisor might be required to accommodate the overall supervising process requirements that would unnecessarily complicate the formulation, verification, maintenance, and upgrading of the supervisory scheme. Thus, the given control system is constituted from a mix of reconfigurable continuous/discrete time and

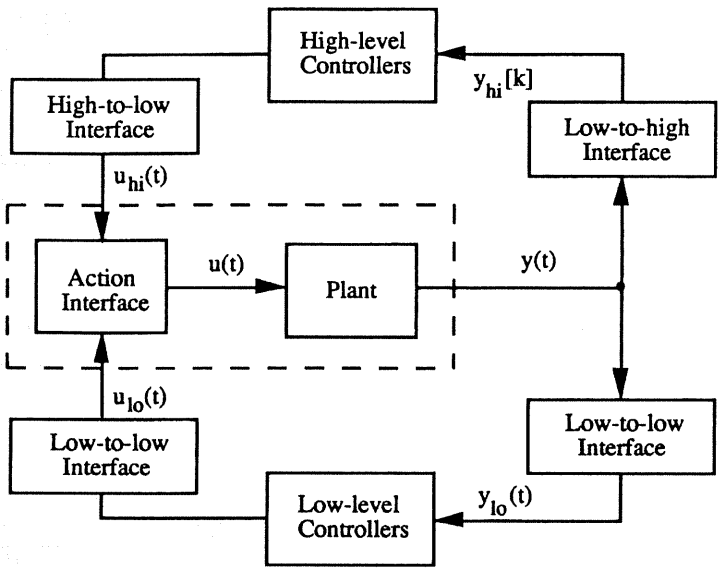

Figure 2. Resulting reconfigurable hybrid control system.

discrete event processes and is structured in two levels as shown in Fig. 2. The low-level controllers represent the control algorithms that reside in the executors and the high-level controllers represent the decision-making algorithms that implement the co-ordinator and supervisors. While low-level controllers compute control actions based on the current process dynamics, the high-level decisions are based on current observations in the supervised world. To achieve proper system response, control systems analysis and synthesis techniques must be provided to ensure appropriate operation of each level under the figure of reconfiguration. For the low-level case, control synthesis tools have been extensively reported. For the high-level case, the paradigm of reconfigurable DES is initially addressed in this paper, as presented below, to address reconfigurable supervisory implementations.

3. Framework for DES

3.1. *The discrete event model*

Based on Li and Wonham (1988 a), the following discrete event model is introduced by Garcia (1993) to represent the DES

$$M := (\Gamma, \Psi) \tag{1}$$

where M is called the discrete-event mechanism and Γ and Ψ are called the *static* and *dynamic* components of M, respectively. Specifically, Γ is defined over the following 3-tuple

$$\Gamma := (\mathscr{V}, \Sigma, \mathscr{P}) \tag{2}$$

where $\mathscr{V}$, Σ, and $\mathscr{P}$ are finite sets. Specifically, $\mathscr{V}$ is the non-empty set of *internal state variables* of M defined over an index set $\hbar = \{1, 2, \ldots, n\}$. These variables are used to represent internal system dynamics. Therefore, they can be classified as latent variables (Sreenivas & Krogh 1991). The set Σ is the event space or *input alphabet* representing the set of events defined for the process M. It is customary to observe the rule that no two events in Σ are mutually exclusive. Finally, the set $\mathscr{P}$ is the collection of *state predicates* defined on the state space $\mathscr{X}$. A predicate $P \in \mathscr{P}$ is a Boolean map

$P: \mathscr{X} \to \{0, 1\}$ that holds on $x \in \mathscr{X}$ if $P(x) = 1$. With every $P \in \mathscr{P}$, a state region $\mathscr{X}_P$ can be associated such that, for all $x \in \mathscr{X}_P$, P takes the value one. Consequently, the names of predicates and state regions will be used interchangeably on what follows next. On the other hand, each internal variable $x_i \in \mathscr{V}$, also called a state variable, has an associated range called range (x_i) which can be finite or infinite. Then, the range set of $\mathscr{V}$ is the state-space $\mathscr{X}_\mathscr{V}$ given by the product of the ranges of the variables in $\mathscr{V}$. That is

$$\mathscr{X}_\mathscr{V} := \bigtimes_{i \in \hbar} \text{range}(x_i) \tag{3}$$

where $\times$ represents the cartesian product. (For brevity of notation, the dependence of the state-space $\mathscr{X}_\mathscr{V}$ on $\mathscr{V}$ is dropped in the following.)

On the other hand, the dynamic component Ψ of a DES model is defined over the following 4-tuple

$$\Psi := (I, d, f, O) \tag{4}$$

where $I(\cdot)$ is called the *input* function, $O(\cdot)$ is the *output* function, $d(\cdot)$ is the *possible events* function and $f(\cdot)$ is a partial function called the *state transition function*. The input and output functions $I(\cdot)$ and $O(\cdot)$ are, in general, mappings of the form

$$I: \bigtimes_i \mathscr{X}_i \times 2^\Sigma \to \mathscr{X} \times \Sigma$$

$$O: \mathscr{X} \times \Sigma \to \mathscr{X}^O \times 2^\Sigma$$

where $\mathscr{X}_i$ is the state-space of the ith mechanism attached to M and $\mathscr{X}^O$ is the output state-space of M. The possible events function $d: \mathscr{X} \to 2^\Sigma$ is a set-valued function that specifies the set of possible events defined at each state. Formally

$$d(x) := \{\sigma \in \Sigma: D.\sigma.x = 1\} \tag{5}$$

where the mapping $D.\sigma: \mathscr{X} \to \{0, 1\}$ is a Boolean-valued expression in the variables of $\mathscr{V}$ representing the enabling condition for the event σ; that is

$$D.\sigma.x := \begin{cases} 1, & \text{if } \sigma \in d(x) \\ 0, & \text{otherwise} \end{cases}$$

The state transition function $f: \mathscr{X} \times \Sigma \to \mathscr{X}$ defines the dynamics of a given DES. It indicates how changes in states occur in a given DES due to incoming events. The partial function $f(x, \sigma)$ can be extended to a subset of Σ^* as

$$f(x, \varepsilon) := x, \quad \text{or} \quad f_\varepsilon(x) = x, \quad \forall x \in \mathscr{X} \tag{6}$$

where ε denotes the empty sequences in Σ^*. The implication of (6) is that every point is a fixed point of the mapping f_ε; and

$$f(x, s^\wedge\sigma) := \begin{cases} \text{undefined} & \text{if either } f(x, s) \text{ or } f(f(x, s), \sigma) \text{ is undefined} \\ f(f(x, s), \sigma) & \text{otherwise} \end{cases} \tag{7}$$

where s is a given trace (sometimes also called a word), and $r^\wedge t$ is the concatenation of r and t with $r, t \in \Sigma^*$. The dynamics of M are then modelled by $f(\cdot)$ as follows:

$$x[k+1] = \mathbf{f}(x[k], \sigma[k+1]); \quad \text{with } P_0.x[0] = 1 \tag{8}$$

where $x[k] \in \mathscr{X}$ is the state after the kth event, $\sigma[k]$ is the kth event, and $x[0]$ is an initial state satisfying a given initial predicate P_0. A mechanism M is deterministic

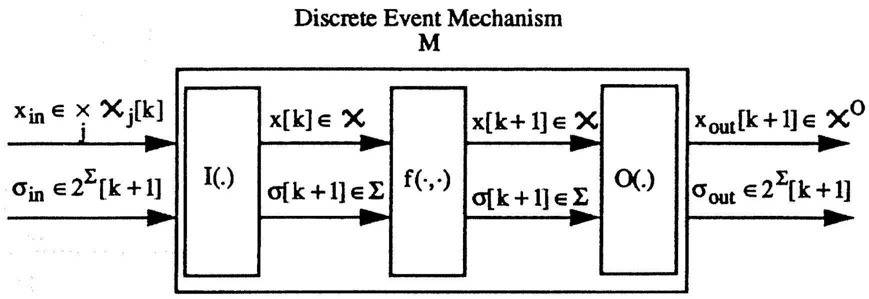

Figure 3. Input/output signal flow of a discrete event mechanism M.

when $\mathbf{f}(x[k], \sigma[k+1])$ is a singleton for every $x[k]$ and $\sigma[k+1]$. Specifically, M can be interpreted as a device that starts within an initial state region $\mathscr{X}_0$ (i.e. $x[0] \in \mathscr{X}_0$) and executes state transitions as a response (or cause) of a sequence of events and state conditions. Notice that $\sigma \in d(x[k])$ implies that $f(x[k], \sigma)$ is defined. Therefore, the next event $\sigma[k+1]$ in (8) must satisfy $\sigma[k+1] \in d(x[k])$ for all k. It is customary to observe the rule that no two transitions fire simultaneously. Following Sreenivas and Krogh (1991), Fig. 3 shows a representative input/output signal flow of a discrete mechanism M.

3.2. *Control approach*

To control DES, certain events in the system are enabled or disabled by the choice of control inputs thus governing, whenever possible, transitions among states. Here, control specifications are given in terms of predicates on the set of states. The design problem is then to formulate a control agent, hereafter called the supervisor, that assigns control patterns (to be defined below) at each environment state so that a specified predicate can be satisfied. In general, an event may or may not be under the control of a supervising agent. Based on their controllabilty, events can be classified into *controllable* and *uncontrollable* events. While controllable events can be disabled or prevented from occurring whenever desired, uncontrollable events are those whose occurrence cannot be governed by a supervisor. Thus, the set Σ is partitioned into two disjoint subsets, Σ_c and Σ_u. The subset Σ_c represents the set of controllable events, Σ_u represents the set of uncontrollable events with $\Sigma_u = \Sigma - \Sigma_c$. The control law for a discrete event process is then realized by a control pattern for M.

Definition 3.1: A *control pattern* for a mechanism M is defined as a Boolean function $U.\sigma: \mathscr{X} \to \{0, 1\}$ that specifies if a controllable event σ is allowed to occur at a given state x. Specifically, an event σ is enabled by $U.\sigma$ at x if $U.\sigma.x = 1$; it is disabled, otherwise. □

Definition 3.2: The *control input* $u[k] \in 2^{\Sigma_c}$ is the set of controllable events that are allowed by the supervisor to occur (i.e. not disabled) at the instant k on a controlled DES. This can be expressed as follows:

$$u[k] := \{\sigma \in \Sigma_c : \sigma \in d(x[k]), U.\sigma.x = 1\}$$ □

As indicated previously, the supervisor has authority only over controllable events. In addition to these enabled events, other events that are out of the authority of the supervisor may occur at a given plant state $x[k]$. These events are called disturbing events or disturbances.

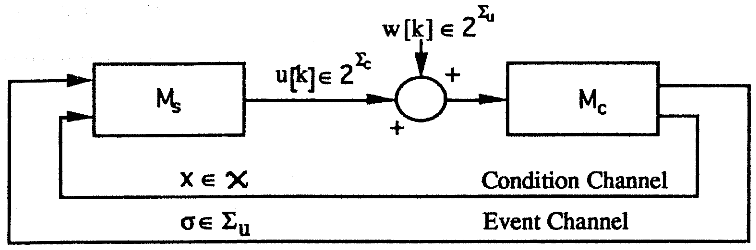

Figure 4. Discrete event model for closed-loop supervision.

Definition 3.3: The *disturbances* acting on a DES, denoted by $w[k]$, are the set of uncontrollable events that may occur at a given state $x[k]$ defined as follows

$$w[k] := \{\sigma \in \Sigma_u : \sigma \in d(x[k])\} \qquad \square$$

Notice that $w[k] \in 2^{\Sigma_u}$ for all k. With the introduction of uncontrollable events, the set of current possible events for the present state $x[k]$ is dynamic and equal to the union of disturbances and the set of control inputs. That is

$$\sigma[k+1] \in u[k] \cup w[k] \qquad (9)$$

The *controlled discrete-event process* M_c, which results from the original machine M under the context of Σ_c, emphasizes the event constraints and control mechanism explicitly indicated by (9). M_c can be interpreted as a version of M that admits external control (Ramadge and Wonham 1989). The resulting control system, that is, the controlled DES process M_c with a supervisor M_s generating $u[k]$, is called the *supervised* system. We will restrict our attention to the case where the plant is driven by events and generates outputs that the supervisor monitors. Thus, the supervisor design problem can now be defined as the problem of finding a supervisor that disables certain events at the appropriate time so that the controlled discrete event process M_c behaves according to certain predicate constraints. Figure 4 illustrates a closed loop arrangement, denoted by (M_c, M_s), conformed by M_c and M_s where two types of channels can be identified: *condition channels*, which transport piecewise constant signals representing state conditions; and *event channels*, which transport discrete (delta) signals representing system events (Sreenivas and Krogh 1987). The plant M_c influences the state transitions of M_s by means of the observed plant state and, possibly, incoming events, while M_c is driven by a sequence $\{w[k]\}$ of disturbances and by a sequence $\{u[k]\}$ of control inputs determined by the consecutive states $x[k]$ of M_s. This class of supervision will be called *event/state* feedback control. Under the illustrated event/state feedback control, the input and output mappings $I(\cdot)$ and $O(\cdot)$ defined in §3.1 can be expressed for each case as follows.

$$\text{Plant:} \quad I^P: 2^{\Sigma} \to \Sigma \qquad O^P: \mathcal{X}^P \times \Sigma \to \mathcal{X}^o \times \Sigma_u$$

$$\text{Supervisor:} \quad I^S: \mathop{\times}_i \mathcal{X}_i \times \Sigma \to \mathcal{X}^S \times \Sigma \qquad O^S: \mathcal{X}^S \to 2^{\Sigma_c}$$

where $\mathcal{X}^P$ and $\mathcal{X}^S$ represent the state-space of the plant and the supervisor, respectively; $\mathcal{X}_i$ represent the state-space of the ith mechanism M_i attached to a given supervisor; and Σ_c and Σ_u are the controllable and uncontrollable subsets of Σ. While the space $\mathcal{X}^o$ is the output space of the plant and usually equal to $\mathcal{X}^P$, $\mathcal{X}^S$ is formed in general as

$$\mathcal{X}^S = \mathop{\times}_i \mathcal{X}_i \times \mathcal{X}^S_{\text{int}}$$

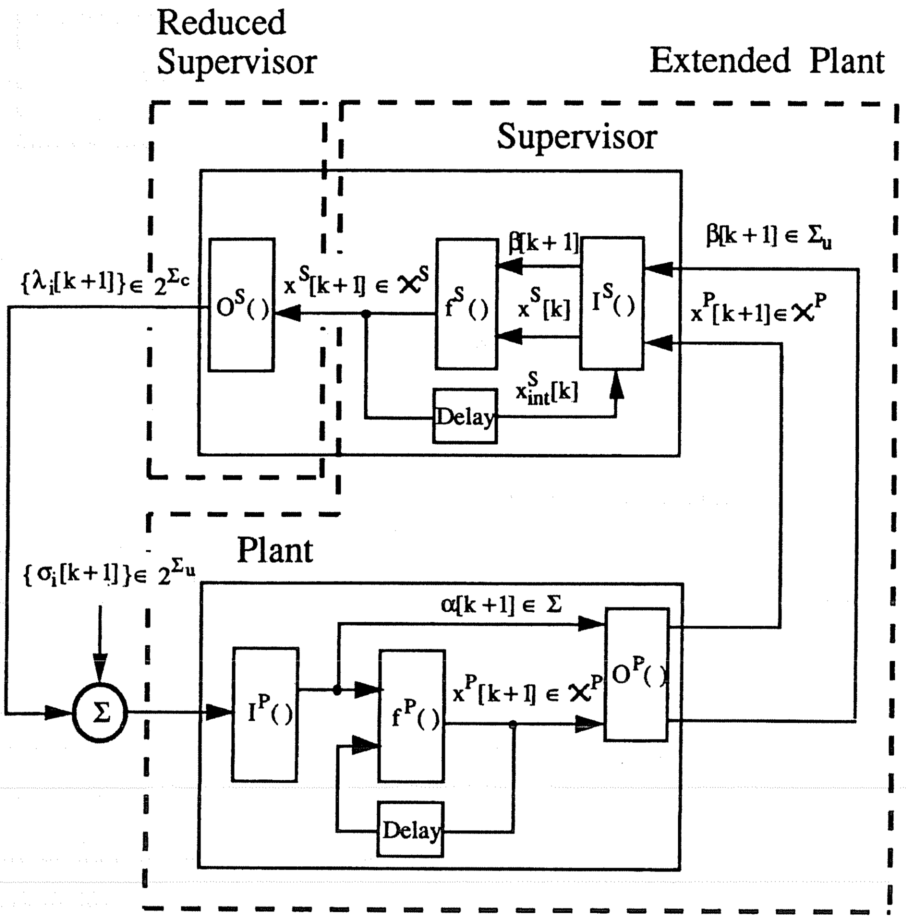
Reduced Supervisor
Extended Plant
Supervisor
$\beta[k+1]$
$\beta[k+1] \in \Sigma_u$
$\{\lambda_i[k+1]\} \in 2^{\Sigma_c}$
$O^S()$
$x^S[k+1] \in \mathcal{X}^S$
$f^S()$
$x^S[k]$
$I^S()$
$x^P[k+1] \in \mathcal{X}^P$
Delay
$x^S_{int}[k]$
Plant
$\{\sigma_i[k+1]\} \in 2^{\Sigma_u}$
$\alpha[k+1] \in \Sigma$
Σ
$I^P()$
$f^P()$
$x^P[k+1] \in \mathcal{X}^P$
$O^P()$
Delay

Figure 5. Plant/supervisor pair in an event/state feedback loop.

where $\mathcal{X}^S_{int}$ is defined as the *internal state-space* of the supervisor. Specifically, $\mathcal{X}^S_{int}$ is introduced to allow a given supervisor to record process history information. This auxiliary information may be required when enforcing supervisory constraints derived from dynamic specifications. Supervisors observing an internal space are called *dynamic* supervisors; otherwise, they are called *static* or 'memoryless' supervisors (Kumar *et al.* 1993, Li 1991). In view of these mappings, Fig. 4 is expanded as in Fig. 5. Notice from Fig. 5 that given an 'event/state' feedback configuration involving an initially (dynamic) supervisor, an equivalent 'state' feedback configuration involving a (static) supervisor can, in general, be found by defining a new 'extended' plant as well as a 'reduced' supervisor as indicated in Fig. 5. This observation will be used later in the derivations.

4. Control of time-varying DES

The objective of a supervisor is to modify the open loop response of a given DES M_c to track the desired response as close as possible. Let $M_c[k]$ denote the discrete event model of the time-varying discrete event mechanism M_c to be controlled, where the explicit dependence on 'time' is indicated by the variable k. Without loss of generality, only plant/supervisor pairs arranged in state feedback configurations (i.e. involving static supervisors) are considered here. If this is not the case, equivalent

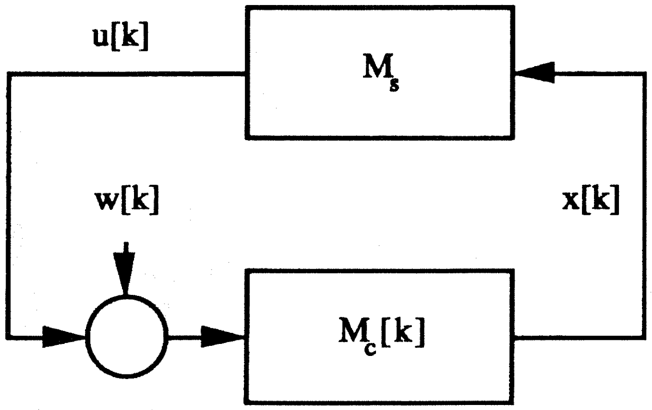

Figure 6. Single robust supervisor approach.

relations can be first formulated, for example, as the plant-extension/supervision-reduction procedure mentioned for Fig. 5. Furthermore, it is assumed that $M_c[k]$ belongs to a finite set of DES models M_{c_i} at any time, as follows

$$M_c[k] \in \{M_{c_i}\}_{i=1}^{m} \tag{10}$$

where m denotes the number of possible operating conditions. To control $M_c[k]$, two supervisory approaches can be used.

4.1. *Single robust supervisor approach*

Conceptually, a single *robust* supervisor M_s can be designed to account for all possible variations of $M_c[k]$ as seen in Fig. 6. In this case, $M_c[k]$ can be viewed as

$$M_c[k] = \hat{M}_c + \Delta M_c[k] \tag{11}$$

where $\hat{M}_c$ is the nominal time-invariant model of the controlled DES and $\Delta M_c[k]$ represents the time-varying modelling uncertainties. A supervisor M_s is thus synthesized to handle the plant. However, this approach suffers from several problems. First, the variability of $\Delta M_c[k]$ could be large enough that a single supervisor may not exist for all cases. Even if a supervisor exists, it may become too complex to be analysed and synthesized. The supervisor thus developed may also be overly conservative and not satisfy the given performance specifications. Such a design is hard to justify if large uncertainties occur rarely. On the other hand, if these rather rare cases are not considered, the control system design is vulnerable to operational risk over the service life of the plant. Thus, system performance may be signficantly degraded to ensure robustness of the supervisory scheme.

4.2. *Reconfigurable supervisor approach*

An alternative approach to control DES with large operational changes is based on the concept of reconfiguration. In this case, instead of supervising the controlled DES with an unique supervisor M_s, the control efforts are exercised by employing a time-varying supervisor $M_s[k]$ as indicated in Fig. 7. Here, a finite set of possible supervisors is defined as

$$M_s[k] \in \{M_{s_i}\}_{i=1}^{n} \tag{12}$$

where n indicates the number of designed supervisors. In general, $n \leqslant m$. However, to

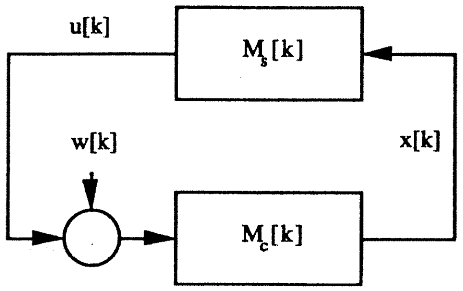

Figure 7. Time-varying supervisor approach.

simplify the description, it is assumed that $n = m$ and that control specifications are given on the basis of predicates P_i for each plant operational condition M_{c_i} with $i = 1, \ldots, n$. Therefore, the problem is to design a set $M_s[k]$ of supervisors such that

$$(\forall M_{c_i}) \exists M_{s_j} \quad \text{subject to} \quad R_e(M_{c_i}/M_{s_j}, P_i) \leqslant P_i \tag{13}$$

where M_{c_i}/M_{s_j} is the feedback composition of M_{c_i} and M_{s_i}, $R_e(M, P)$ is the reachable predicate from an initial predicate P under the discrete mechanism M, as given in the Appendix, and $\leqslant$ is the partial ordering defined as $P_1 \leqslant P_2$ if $P_1 \wedge P_2 = P_1$. Thus, supervisors are designed to accommodate only a restricted subset of all possible plant variations. Recall that a set of supervisors may also be required to enforce online changing supervisory specifications present in reconfigurable supervisory systems. This approach simplifies the design of supervisors as the tasks of analysis, synthesis, verification, upgrade and maintenance become easier. In addition, from the practical point of view, a reconfigurable system does not enforce a complete modification of an existing configuration, but instead it extends the current system to accommodate additional requirements.

Remark 4.1: Equation (13) implicitly assumes that the plant/supervisor pair starts operating in a space-state region where the corresponding predicate is satisfied. However, this is not essential. □

5. Reconfigurable supervisory architecture

This paper particularly addresses hierarchical DESs that can reconfigure their supervisors based on operational system changes. This class of DES will be called reconfigurable discrete event systems (RDES). These and other DES arrangements are discussed further by Garcia (1993) while a reconfigurable hybrid system for process control that requires RDES techniques is given by Garcia *et al.* (1995). In a RDES architecture, as illustrated in Fig. 8, a bank of supervisors is defined for each controlled DES and supervisors are designed for each identified operational condition. Only one supervisor in each bank is acting at any time. When a higher decision maker in the hierarchy (in this case, the coordinator) decides to change the supervising algorithm, a switch to a new supervisor takes place. This supervisory strategy introduces a transient period, which is expected to die out, based on the designing criteria used to implement any supervisor. Eventually, a set of target states is reached from where the new desired behaviour can be achieved.

Figure 8 introduces a new component in the RDES system; namely, the *control channel.* Via the control channel Con$_i$, the ith supervisor applies control to the

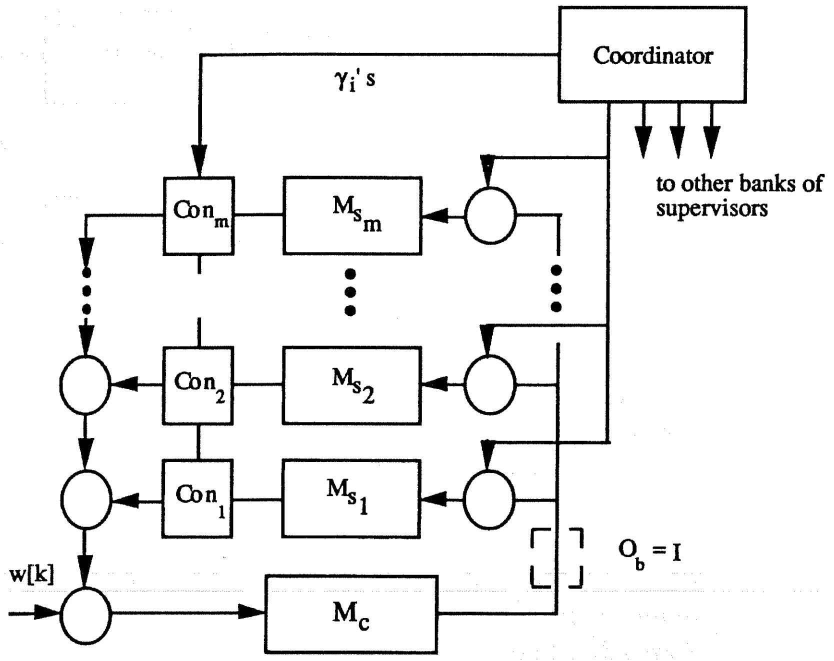

Figure 8. Simplified diagram of a reconfigurable discrete event system.

controlled DES M_c. The ith control channel is a mapping $\mathrm{Con}_i: 2^{\Sigma_c} \times \{0, 1\} \rightarrow 2^{\Sigma_c} \cup \{\varepsilon\}$ implemented as the following conditional identity function:

$$\mathrm{Con}_i(u, \gamma_i) := \begin{cases} u & \text{if } \gamma_i = 1 \\ \varepsilon & \text{otherwise} \end{cases} \tag{14}$$

where $\gamma_i[k] \in \{0, 1\}$ is a control signal generated by the coordinator to govern the Con_i output. The scalar γ_i acts as a *supervisor enabling* signal. Specifically, the output of a given supervisor M_{s_i} is communicated to the controlled DES M_c if its respective control channel has been enabled (i.e. $\gamma_i = 1$) by the co-ordinator. This, in turn, is equivalent to enabling or disabling a given supervisor's operation.

Remark 5.1: Reconfiguration masks control outputs. Therefore, supervisors' actions are conditioned to the coordinator's decisions. On the other hand, to reduce the time period of any future transient, it is desirable continually to provide each supervisor with information about the current state of the controlled plant. To this end, it is enforced that the coordinator has no influence on the observability aspects of any supervisor, and plant state changes (and event occurrences) are 'continually' transmitted as inputs to each supervisor. Because no supervisor is 'blind' at any time, transfers from one supervising algorithm to another are performed more effectively. If that were not the case, a just activated supervisor might require an extensive period before gaining enough event history to reach a good system state estimation. □

6. Problems resulting from reconfiguring supervisors

It is often assumed in the DES literature that the initial state of a given plant/supervisor pair is assumed to be fixed, *a priori* known and one of the 'legal' states. Starting from a pre-defined initial region, the objective of a supervisor is then to confine the process behaviour within specified bounds. This corresponds to a regulatory problem. However, under reconfigurable supervision, the initial state-space region may not be the legal region for the current supervising condition. Switching from a given supervisor to another introduces new important aspects, particular to RDES, as discussed next.

Let M_c denote the plant model and M_{s_i} the supervisor that has been used so far. If M_{s_i} has uniquely been acting over M_c for a sufficiently large period, it is safe to say that an operational steady-state has been reached. Under proper supervision, any state trajectories generated by the close loop system, conformed by M_c and M_{s_i}, have been confined to the desired state region. That is,

$$(\forall k > t_0 \wedge k < t_f)\, M_s[k] = M_{s_i} \wedge P_i . x[k] = 1 \tag{15}$$

where t_0 and t_f denote given instants and the predicate P_i defines the control specification for the M_c/M_{s_i} pair. Assume that at t_f, the current supervisor M_{s_i} is disabled and a new (previously unattached) supervisor M_{s_j} starts to act directly on the plant M_c. Let P_j denote the 'target' predicate for the new plant/supervisor configuration M_c/M_{s_j}. At time t_f, while $P_i.x[t_f] = 1$, it is possible that $P_j.x[t_f] = 0$. Because the current state $x[t_f]$ may not satisfy the new control specification P_j, it must be ensured that the next desired predicate can be satisfied after a finite number of state transitions from the current valid predicate. Therefore, convergence from P_i to P_j must be guaranteed for proper operations of the system. This forces a transition period of the supervising response. During this period, the new (M_c, M_{s_j}) pair moves to certain conditions from which the new desired supervising behaviour P_j can be guaranteed. A transitory and a steady-state period can be identified and supervisory algorithms must be then designed to ensure that the transient and steady response have the desired form. The following sections address these problems.

7. Control tasks on reconfigurable discrete event systems

To limit the unsupervised behaviour of the plant M_c, a supervisor M_s is employed in a feedback configuration with M_c as indicated in §3. This control is achieved by the function $u(\cdot)$ generated by M_s. Following the control requirements indicated in §6, the control tasks required in a reconfigurable architecture are formulated as the fulfilment of the next three constraints.

7.1. *Task 1—Control invariance*

This is achieved by a supervisor that ensures, by control actions, that a given predicate on the state space $\mathscr{X}$ of M_c remains invariantly true whenever it is initially satisfied. Thus, given that a pair (M_c, M_{s_i}) has reached a steady-state condition, it is required that the specified predicate P_i be invariantly satisfied whenever it is initially satisfied. That is

$$(\forall x \in \mathscr{X}_{P_i})(\forall \sigma \in u[k] \cup w[k])\, \mathrm{sp}.f_\sigma . P_i \leqslant P_i \tag{16}$$

or equivalently

$$(\forall x \in \mathscr{X}_{P_i})(\forall \sigma \in u[k] \cup w[k])\, P_i \leqslant \mathrm{wlp}.f_\sigma . P_i \tag{17}$$

where $u[k]$ is the control input to M_c generated by M_{s_i}, $w[k]$ is the externally induced disturbance at the instant k, and sp and wlp are the strongest postcondition and the weakest liberal precondition predicate transformers, respectively, as presented in the Appendix. This task usually occurs in regulatory problems. The idea is to maintain the state of the system inside a certain region despite disturbances. A set-point, specified in terms of conjunction and/or disjunction of predicates, is then 'closely' followed. The proximity or satisfaction to a given specification is generally measured by defining an index on the state space.

7.2. *Task* 2—*Region avoidance*

This is achieved by a supervisor that ensures, by control actions, that a given region characterized by a certain predicate will never be reached. In reference to RDES, at some point, the coordinator switches from a supervisor M_{s_i} to another M_{s_j} and a new predicate P_j is desired to be satisfied. For the general case of $P_j \neq P_i$, the supervisory system must then try to reach the target region P_j from this initial condition P_i. To this end, a series of state trajectories leading to P_j can be generated. However, in this process, it must be ensured that the system does not enter into 'bad' regions where it may fail to converge. Let $P_B \in \mathscr{P}$ denote a bad predicate that defines a forbidden state region. Then, a correct supervisor M_s must guarantee that, for any sequence of next possible transition, the following equation holds

$$(\forall x \in \mathscr{X}_{\neg P_i})(\forall \sigma \in u[k] \cup w[k]) \, \mathrm{sp}.f_\sigma . P_x \leqslant \neg P_B \tag{18}$$

Equation (18) ensures that all subsequent state trajectories leaving P_i stay out of P_B.

7.3. *Task* 3—*Convergence*

This is achieved by a supervisor that ensures, by control actions, that a given region characterized by a certain predicate can be reached from a specified starting space. Thus, the convergence of the supervisory system to the target predicate must be ensured. That is, starting from a state satisfying P_i, the system should converge to a state in P_j after, at most, countably many transitions. This can be seen as an asymptotic convergence. However, it is often desired to achieve convergence in a finite number of transitions. In that case, there must exist a positive integer value 'q' such that

$$\exists q \in \mathscr{Z} \quad \text{subject to} \quad P_j \wedge R_e(M, P_i, q) \neq P_\varnothing \tag{19}$$

where $R_e(M, P, q)$ is the reachable predicate that can be reached from a given predicate P in q event transitions. This capability is particularly important in many situations such as error recovery and system reconfiguration.

From the above discussion, it can be concluded that the concept of reconfiguration relies on the possibility of driving a process under supervision from an arbitrary initial region to a specified target subspace of the state-space universe following permitted state paths, and then keeping it there as required.

8. Solutions to the control tasks for RDES

The closed loop system consisting of a controlled DES or plant M_c and a supervisor M_s will be denoted by M_c/M_s, hereafter called the system. If M_s is characterized by a feedback control law u, then the system can be written as $M_c\backslash u$ to emphasize the control action, u, performed on the plant M_c. The task of supervisory control synthesis is facilitated if the state feedback approach is adopted and the control specifications are given by predicates (Ushio 1989). To this end, solutions to the control tasks based on (static) state feedback control, as indicated in §3, are presented next. For event/state feedback configurations involving dynamic supervisors, techniques such as that discussed in Fig. 5 can be used to apply the results presented here. A formal extension to event/state supervision will be given in a sequel to this paper.

8.1. *Task I—Control invariance*

Let a given predicate P be initially satisfied. It is required that, after any event firing sequence, the process continues to satisfy P under the current closed-loop system M_c/M_s. Formally

$$(\forall \sigma \in u[k] \cup w[k])\, P \leqslant \text{wlp}.f_\sigma . P \tag{20a}$$

or equivalently,

$$(\forall \sigma \in u[k] \cup w[k])\, \text{sp}.f_\sigma . P \leqslant P \tag{20b}$$

Proposition 8.1: *The task described by* (20) *can only be satisfied if the predicate P is Σ_u-invariant* (*Ramadge and Wonham* 1987).

Proof: Intuitively, (20) indicates that, under any sequence of events, the state of the system must be invariantly maintained within the region defined by the predicate P. This condition must be satisfied for any possible event sequences. If there were no disturbances, (20) can be achieved by solely imposing restrictions on the supervisory policies. However, in the case of $\Sigma_u \neq \varnothing$, predicate invariance under disturbances must also be enforced to guarantee (20). This additional restriction, denoted by Σ_u-invariant, is a property of the plant and not of the supervisory scheme employed. The rationale is that uncontrollable events admit no control. A formal proof of this proposition can be found in Ramadge and Wonham (1987). □

Remark 8.1: It may be quite possible that certain applications call for satisfying more stringent requirements during Task I. For example, it may be required that the state trajectories starting from a region defined by a given predicate P_1 not only remain confined to the regioin defined by another predicate P_2 but also that all the states where P_2 holds are visited. In this case, the supervisory control problem can be solved if and only if P_2 is controllable from P_1. That is, controllability is a necessary and sufficient condition for the existence of a supervisor that ensures proper execution of Task I. Definitions for controllability and alike can be found in Kumar (1993), Li (1991), Li and Wonham (1988 a), Ramadge and Wonham (1987), Ushio (1989) while a method for computing the weakest controllable predicate is given by Kumar *et al.* (1993). □

8.1.1. *Algorithms to synthesize supervisors to achieve Task I*:

To synthesize supervisors that guarantee that Task I is achieved, the following algorithms, which are equivalent to each other, are proposed.

Synthesis algorithm 8.1.1

Given that the desirable predicate P is Σ_u-invariant and M_c is in a state x where P is satisfied, a controllable event σ is enabled at x if and only if its firing implies that the control-invariant condition given by

$$(\forall \sigma \in u[k])\, P \leqslant \text{wlp}.f_\sigma . P \quad k = 0, 1, 2, \ldots$$

or

$$(\forall \sigma \in u[k])\, \text{sp}.f_\sigma . P \leqslant P \quad k = 0, 1, 2, \ldots$$

is preserved. That is

$$(\sigma \in \Sigma_c)\, P \leqslant (U.\sigma \Rightarrow \text{wlp}.f_\sigma . P)$$

or equivalently

$$(\sigma \in \Sigma_c)\, U.\sigma \Rightarrow (\text{sp}.f_\sigma . P \leqslant P)$$

In this case, it is said that $U.\sigma$ is an *σ-friend* (Ramadge and Wonham 1987) of P.

Synthesis algorithm 8.1.2 (derived from Lemma 2.2 of Li and Wonham 1988 a)

Given that M_c is started in any state satisfying a given predicate P_1, and a weaker predicate $P_2 \geqslant P_1$ is control-invariant, the following condition on predicate reachability

$$R_e(M_c \backslash u, P_1) \leqslant P_2$$

holds if the corresponding feedback predicate $U.\sigma$ for the (static) feedback u is chosen as follows

$$(\forall \sigma \in \Sigma_c)\, U.\sigma := \text{wlp}.f_\sigma.P_2$$

where $R_e(M \backslash u, P)$ denotes the predicate reachable from P under the mechanism M having, as control input, u.

Synthesis algorithm 8.1.3

Given that P is valid at the current state x, a controllable event σ is enabled if and only if, upon firing σ, P continues to be valid under any possible subsequent sequence of uncontrollable events. That is,

$$(\forall \sigma \in \Sigma_c)\, \sigma \in u(x) \quad \text{iff } R_e({}^uM_c, \text{sp}.f_\sigma.P_x) \leqslant P$$

where P_x is the predicate valid exactly at x (i.e. $P_x: (x = x)$) and ${}^uM$ is the submechanism of M generated by disabling all controllable events.

Computationally efficient implementations of algorithm 8.1.3 can be found in, for example, Li (1991) and Garcia (1993), where the specified supervisory problem (i.e. guaranteeing Task I) is re-formulated as integer linear programming problem (Kaufmann and Arnaud Henry-Labordere 1977).

8.2. *Task II—Region avoidance*

The objective of this task is to avoid a certain region in the state-space. This is particularly important when traversing from a given state region to another, a situation that may occur whenever the initial state x_0 of the DES is not one of the legal states of the current plant/supervisor closed-loop system.

Let P_T denote the (target) predicate that is valid in the legal states for the current plant/supervisor closed-loop system; and let P_B denote the predicate that defines the 'bad' or prohibited region in the state-space. Starting from the initial state x_0, the requirement is to arrive at a state y where P_T is valid without validating P_B at any time. Avoidance of P_B must be satisfied not only at the given current state of the DES but also at any state that may uncontrollably lead to P_B. That is, the predicate to be avoided is

$$P_B \vee R_e^{-1}({}^uM_c, P_B) \tag{21}$$

with $R_e^{-1}(M, P)$ denoting the attractable predicate from where P can be reached under the mechanism M. Since $P_B \leqslant R_e^{-1}({}^uM_c, P_B)$ always holds, (21) can be uniquely characterized by

$$R_e^{-1}({}^uM_c, P_B) \tag{22}$$

Thus, to avoid P_B, a given $\sigma \in \Sigma_c$ is enabled if, after its firing, the trajectory does not enter a region where (22) is valid; i.e.

$$(\sigma \in \Sigma_c)\, \sigma \in u(x) \quad \text{if } R_e^{-1}({}^uM_c, P_B).f_\sigma(x) = 0 \tag{23}$$

Notice, though, that (23) should not uniquely define the condition needed for an event σ to belong to $u(x)$ as seen below.

Assume that the evolving state trajectory has entered the region defined by (22) due to a given uncontrollable reason. Then, the only way the closed loop system can move out of this region, which could potentially lead to P_B, is to enable a certain controllable event at x; i.e.

$$(\sigma \in \Sigma_c)\, \sigma \in u(x) \quad \text{if } R_e^{-1}({}^uM_c, P_B).x = 1 \tag{24}$$

Thus, conditions (23) and (24) define the instances when a given controllable event may be enabled at any given state, as indicated in the following (synthesis) algorithms.

8.2.1. *Algorithms to synthesize supervisors to achieve Task II.* To synthesize supervisors that will guarantee that Task II can be achieved, the following equivalent algorithms are proposed.

Synthesis algorithm 8.2.1

The following implication should be always satisfied

$$(\forall \sigma \in \Sigma_c)\, U.\sigma.x \Rightarrow R_e^{-1}({}^uM_c, P_B).x \vee \neg R_e^{-1}({}^uM_c, P_B).f_\sigma(x)$$

The above implication can be substituted by an equality if (23) and (24) are the only criteria used to enable controllable events.

Synthesis algorithm 8.2.2

Let the system be at the current state x. Then, a controllable event σ is enabled if and only if, upon firing σ, the permissible region $\neg P_B$ continues to remain valid under any possible subsequent sequence of uncontrollable events or if at the current state the trajectory can uncontrollably reach the undesirable region P_B. That is

$$(\forall \sigma \in \Sigma_c)\, \sigma \in u(x) \quad \text{iff } (R_e({}^uM_c, \text{sp}.f_\sigma.P_x) \leqslant \neg P_B) \vee (P_x \leqslant R_e^{-1}({}^uM_c, P_B))$$

A computationally efficient implementation of algorithm 8.2.2 can be found in Garcia (1993) where the specified supervisory problem (i.e. guaranteeing Task II) is formulated as an integer linear programming problem.

8.3. *Task III—Convergence*

The task is to reach a target state region from the given initial conditions. Let P_1 and P_2 denote the predicates defining the starting and target regions, respectively. The synthesized supervisor must be capable of driving the state of M_c from P_1 to P_2 without violating the mentioned system constraints. In what follows, it is assumed that the reader is familiar with the notion of controllability of predicates and the like, as presented in, for example, Kumar *et al.* (1993), Li (1991), Li and Wonham (1988a).

Definition 8.1: Let P_1 and P_2 denote the initial and target predicates, respectively. Let $\mathcal{K}$ denote a set of non-negative integers such that

$$\mathcal{K} := \{j \in Z^+ : R_e(M_c/M_s, P_1, j) \wedge P_2 \neq P_\varnothing\}$$

where Z^+ denotes the set of all non-negative integers; $R_e(M, P, q)$ is defined as in (19) and $P_\varnothing$ denotes the predicate defined such that $(\forall x \in \mathcal{X}) P_\varnothing.x = 0$. If $\mathcal{K} \neq \varnothing$ for a given plant/supervisor configuration, then the (*minimum*) *predicate for convergence* from P_1 to P_2, denoted as $P_c(P_1, P_2)$, is defined as follows:

$$P_c(P_1, P_2) = R_e(M_c/M_s, P_1, \imath) \quad \text{where } \imath = \min_{j \in K}(j) \tag{25}$$

□

Proposition 8.2: *Let $\wp(P_1, P_2)$ denote the set of state paths $\mathscr{X}^*$ defined as follows*

$$\wp(P_1, P_2) := \{\Pi \in \mathscr{X}^*: P_1 . \Pi(1) = P_2 . \Pi(q) = 1, (\forall i \in \{1, \ldots, q\})\, P_c(P_1, P_2) . \Pi(i) = 1 \quad \text{with } q = |\Pi|\}$$

where Π denotes a given state path on $\mathscr{X}$, $\Pi(i)$ denotes the ith state on the state sequence imposed by Π, and $|\Pi|$ is the length of Π. Assume that $\mathscr{K} \neq \varnothing$. Then

$$\wp(P_1, P_2) \neq \varnothing$$

Proof: From Definition 8.1, $\mathscr{K} \neq \varnothing$ implies that there must exist a state x in $\mathscr{X}$ reachable from a state in P_1 that validates P_2. Therefore, there must exist a state trajectory leaving from a state in P_1 that ends in x, which completes the proof. □

Theorem 8.1: *Let Π denote a given state path on $\mathscr{X}$ and $\Pi(i)$ the ith state on the state sequence imposed by Π. Let $\wp(P_1, P_2)$ be defined as given in Proposition* 8.2. *Assume that $\mathscr{K} \neq \varnothing$ and $P_c(P_1, P_2)$ is controllable from P_1. Then*

$$(\forall x \in P_1) \exists \prod \in \wp(P_1, P_2) \quad \textit{subject to} \quad x = \prod(1)$$

Proof: From the definition of controllability (Kumar *et al.* 1993, Li 1991), $P_c(P_1, P_2)$ is the minimal predicate, weaker than P_1, which is valid on states reachable from the region defined by P_1 and valid on at least one state where P_2 is also valid. From the assumption $\mathscr{K} \neq \varnothing$, $P_c(P_1, P_2)$ must be valid in some states on the state space. In addition, from Proposition 8.2, there must exist a state trajectory (contained in $P_c(P_1, P_2)$) leading to a state in P_2 from a state satisfying P_1. However, from Proposition 8.2 and the assumption of $P_c(P_1, P_2)$ being controllable, any state in $P_c(P_1, P_2)$ can be reached. This implies that there must exist a state trajectory leading to a state in P_2 from each state satisfying P_1. This completes the proof. □

Remark 8.2: From Theorem 8.1, it can be seen that there exist state paths leading from any state satisfying P_1 to P_2 contained in $P_c(P_1, P_2)$, given that any state in $P_c(P_1, P_2)$ is reachable. □

Theorem 8.1 indicates that there exists a state feedback supervisor that satisfies (19) if and only if P_2 is controllable from P_1.

8.3.1. *Algorithms to synthesize supervisors to achieve Task III.* To synthesize supervisors that will guarantee that Task III can be achieved, the following equivalent algorithms are given. First, in light of Theorem 8.1 and Theorem 2.1 in Li and Wonham (1988 a), the synthesis of a supervisor that ensures convergence is presented below.

Synthesis algorithm 8.3.1

(i) Let $P_c(P_1, P_2) \neq P_\varnothing$.

(ii) If $R_c(P_1, P_2)$ is not controllable from P_1, find the supremal sub-predicate of $R_c(P_1, P_2)$, i.e. $\sup C(P_1, R_c(P_1, P_2))$.

(iii) Assume then that

$$\sup C(P_1, P_c(P_1, P_2)) \wedge P_2 \neq \varnothing \tag{26}$$

If (26) is not satisfied for the current $R_c(P_1, P_2)$, increase the value of i in (25) until (26) is valid.

(iv) Then, any controllable event σ may be enabled at a state x only if its firing leads to a state from where it is known that P_2 can be reached; that is,

$$(\forall \sigma \in \Sigma_c)\, \sigma \in u(x) \quad \text{if } \mathrm{sp}.f_\sigma.P_x \leqslant \sup C(P_1, P_c(P_1, P_2))$$

Next, based on Theorem 8.1 and Theorem 3.1 in Li and Wonham (1988 b), an alternative algorithm for supervisor synthesis is given below.

Synthesis algorithm 8.3.2

(i) Assume the conditions (i)–(iii) taken by Algorithm 8.3.1.

(ii) Then, any controllable event σ may be enabled at a given state x if and only if, after the firing of σ, the predicate $\sup C(P_1, R_c(P_1, P_2))$ is satisfied under any possible subsequent sequence of uncontrollable events. That is

$$(\forall \sigma \in \Sigma_c)\, \sigma \in u(x) \quad \text{iff } R_e({}^uM_c, \mathrm{sp}.f_\sigma.P_x) \leqslant \sup C(P_1, P_c(P_1, P_2))$$

Synthesis algorithm 8.3.3

On certain occasions, it may be difficult or computationally intractable to verify the conditions imposed by algorithms 8.3.1 or 8.3.2 for offline synthesis of a supervisor. An algorithm may also be formulated to define online the supervisory control policies as the system evolves. Specifically, such an algorithm would identify the current system state and then only enable those controllable events that can ensure convergence from the given state to the target state region P_2. To ensure that the closed-loop system M_c/M_s eventually reaches P_2, from another region P_1, a constraint on state trajectories starting at P_1 and ending in P_2 must be imposed. Failure to impose (minimal) uncontrollable constraints may cause the system to diverge from reaching the target region P_2. For example, it may be required that P_1 be evolved by the state region from where there exists an uncontrollable event sequence that will eventually lead to P_2; that is

$$P_1 \leqslant R_e^{-1}({}^uM_c, P_2) \tag{27}$$

Thus, a controllable event σ may be enabled iff, after its firing, the resulting state still remains in the state region defined by the right-hand term of (27); i.e.

$$(\forall \sigma \in \Sigma_\sigma)\, \mathrm{sp}.f_\sigma.P_x \leqslant (U.\sigma \Rightarrow R_e^{-1}({}^uM_c, P_2)) \tag{28}$$

If (28) is enforced by an online converging algorithm whenever controllable events are considered, it would be guaranteed that no uncontrollable sequence of events would lead the closed loop system out from convergence. In fact, if no controllable event is enabled, (28) indicates that the system will eventually converge to the desired state region. However, because uncontrollable events are out of any control from the given supervisor, (28) ensures convergence in no 'time' span. Thus, the system may loop around P_2 from intolerable periods of time. To improve convergence response, other criteria can be used to select controllable events such as, for instance, reducing the state distance between target region and current states. To preserve consistency in this paper's presentation, the reader is referred to Garcia (1993) where an online algorithm devised to achieve predicate convergence is discussed.

Next, the general problem of predicate convergence is presented.

9. Predicate convergence defined on regions of attraction

9.1. *Introduction*

The possibility of driving the controlled DES from arbitrary initial states to a specified target region is of interest in a more general framework. To this end, stabilization properties and asymptotic behaviour of a given DES needs to be defined and investigated. To the best of the author's knowledge, the first study involving concepts of stabilization of DES was introduced by Brave and Heymann (1989), followed by a redefinition of classical concepts of dynamics as invariant sets and attractors for DES modelled by finite state-machines (Brave and Heymann 1990). This section extends some of the results of Brave and Heymann (1990) for the case of the infinite state DES model of the proposed framework presented in §3. To this effect, some characteristics of open-loop processes are discussed next.

9.2. *Region of strong attraction*

Definition 9.1: Let $P_2 \leqslant P_1$ where P_1 and P_2 are two predicates on $\mathscr{X}$. Then P_2 is a *concentric strong attractor* for P_1 under the discrete event mechanism M, denoted as $P_2 \overset{M}{\Leftarrow} P_1$, if the following conditions are met:

(i) $(\forall \sigma \in \Sigma)\, P_2 \leqslant \text{wlp}.f_\sigma.P_2 \ (\text{rel}\, P_2)$

(ii) $R_e(M, P_1) \leqslant R_e^{-1}(M, P_2)$

(iii) $(\forall x \in \mathscr{X} \text{ with } R_e(M, P_1).x = \neg P_2.x = 1)\, f(x, s) = x \Rightarrow s = \varepsilon$

where ε denotes the empty event sequence. □

Strong attraction ensures that the system eventually converges to a specified target region if it is initialized in a state belonging to its region of attraction. Condition (i) indicates that any state in the region defined by P_2 will stay inside it under any event firing. Therefore, Σ-invariance is a necessary condition for a given predicate P to be a concentric strong attractor. Condition (ii) indicates that any state reachable from P_1 is attractable to P_2. Therefore, any state trajectory leaving P_1 can be then conducted to end in P_2. Finally, condition (iii) indicates that for any state reachable from P_1 but not in P_2, there exists no sequence of events other than the empty string that causes no state changes. Notice that for any predicate P, the smallest region of attraction of P is P itself. However, it will be of interest to find the largest predicate for which P is a strong attractor. To this end, the following theorem is given.

Theorem 9.1: *Let $SA(M, P)$ denote the class of predicates for which P is a concentric strong attractor; i.e.*

$$SA(M, P) := \{Q \in \mathscr{P} : P \leqslant Q, P \overset{M}{\Leftarrow} Q\}$$

Then, $SA(M, P)$ has a maximal element.

Proof: It suffices to show that $SA(M, P)$ is non-empty and closed under disjunction. First, notice that for any predicate P, the smallest region of strong attraction of P is P itself. Therefore, $SA(M, P)$ is non-empty. To prove that $SA(M, P)$ is closed under disjunction of predicates, it must be shown that

$$(\forall Q_1, Q_2 \in SA(M, P))(Q_1 \vee Q_2) \in SA(M, P)$$

To this end, $(Q_1 \vee Q_2)$ must satisfy conditions (i)–(iii) of Definition 9.1. For any two predicates Q_1 and Q_2, Proposition A.2 in the Appendix indicates that

$$\text{wlp}.f_\sigma.Q_1 \vee \text{wlp}.f_\sigma.Q_2 \leqslant \text{wlp}.f_\sigma.(Q_1 \vee Q_2) \tag{29}$$

Because $Q_i \in SA(M, P)$ for $i = 1, 2$, then $Q_i \leqslant \text{wlp}.f_\sigma.Q_i$ and (29) becomes

$$Q_1 \vee Q_2 \leqslant \text{wlp}.f_\sigma.(Q_1 \vee Q_2)$$

and condition (i) is satisfied. To verify (ii), notice that by the definition of $R_e(\cdot, \cdot)$ given in the Appendix, the following equality holds

$$R_e(M, Q_1 \vee Q_2) = R_e(M, Q_1) \vee R_e(M, Q_2) \tag{30}$$

Because $Q_i \in SA(M, P)$ for $i = 1, 2$, then $R_e(M, Q_i) \leqslant R_e^{-1}(M, P)$ and, together with (30), it follows that $R_e(M, Q_1 \vee Q_2) \leqslant R_e^{-1}(M, P)$ and condition (ii) satisfied. To prove that (iii) is satisfied, assume to the contrary that

$$\exists s \in \Sigma^*, \quad x \in \mathscr{X} \quad \text{subject to } (\neg P \wedge R_e(M, Q_1 \vee Q_2)).x = 1, \quad f(x, s) = x$$

Let Π_{12} denote an instance of such a cycling state trajectory; i.e. $\Pi_{12}(1) = \Pi_{12}(|\Pi_{12}|)$. Without loss of generality, assume that Π_{12} starts in the region reachable from Q_1 and outside of P; i.e.

$$(\neg P \wedge R_e(M, Q_1)).\Pi_{12}(1) = 1 \tag{31}$$

Equation (31) does not invalidate our assumption because, as seen from (30), $R_e(M, Q_1) \leqslant R_e(M, Q_1 \vee Q_2)$. Based on $Q_i \in SA(M, P)$ for $i = 1, 2$, then there exists no cycle, neither in the region defined by $\neg P \wedge R_e(M, Q_1)$ nor in $\neg P \wedge R_e(M, Q_2)$. This and (31) imply that $\exists j \in \{1, \ldots, |\Pi_{12}|\}$ such that

$$(\neg P \wedge R_e(M, Q_1) \wedge \neg R_e(M, Q_2)).\Pi_{12}(j) = 1 \tag{32}$$

and

$$(\neg P \wedge R_e(M, Q_2) \wedge \neg R_e(M, Q_1)).\Pi_{12}(j+1) = 1 \tag{33}$$

Because $\Pi_{12}(j+1)$ follows $\Pi_{12}(j)$, there must exist an event σ such that

$$\Pi_{12}(j+1) = f_\sigma(\Pi_{12}(j)) \tag{34}$$

Equation (34) implies that $R_e(M, Q_1).\Pi_{12}(j+1) = 1$ which contradicts (33). Therefore, condition (iii) is also satisfied. □

Corollary to Theorem 9.1: *Let $SA(M, P)$ denote the class of all predicates for which P is a concentric strong attractor. Then, a maximal element of $SA(M, P)$ exists, denoted by $SA_{\max}(M, P)$ such that*

$$\neg \exists Q \in SA(M, P) \quad \text{subject to } SA_{\max}(M, P) < Q \tag{35}$$

Proof: Let Q_1 and Q_2 denote any two predicates such that $Q_1, Q_2 \in SA(M, P)$. Theorem 9.1 shows that $SA(M, P)$ is closed under disjunction of predicates. Therefore, $(Q_1 \vee Q_2) \in SA(M, P)$. Notice that for $i = 1, 2$ $Q_i \leqslant (Q_1 \vee Q_2)$. Then, $(Q_1 \vee Q_2)$ is the maximal element for the set $\{Q_1, Q_2\} \subseteq SA(M, P)$. If there exists another element $Q_3 \in SA(M, P)$ different from Q_1 and Q_2, the same reasoning can be applied to obtain $(Q_1 + Q_2 + Q_3)$, which is the maximal element for $\{Q_1, Q_2, Q_3\}$. The procedure above is repeated until all the elements of $SA(M, P)$ are considered to end with a maximal element. □

Definition 9.2: The maximal element satisfying (35) is called the *ball of strong predicate attraction* of P under M. □

If P is not Σ-invariant, then $SA_{\max}(M, P) = \varnothing$. In such a case, the maximal Σ-invariant subpredicate of P is found to replace P in the above equations. Convergence from a given region P_1 to another, P_2 can now be ensured as indicated by the next proposition.

Proposition 9.1: *A target region defined by a given predicate P_2 can be reached from another region P_1 if P_1 is contained within the ball of strong predicate attraction of P_2; that is*

$$P_1 \leqslant SA_{\max}(M, P_2) \tag{36}$$

Proof: From Theorem 9.1 and Definition 9.2, it is implied that, for any predicate $Q \in SA(M, P)$, P is a strong attractor for Q. In addition, (36) indicates that $P_1 \in SA(M, P_2)$. This and Theorem 9.1 result in P_2 being a strong attractor for P_1, asserting that P_2 can be reached from P_1. □

The implication of Proposition 9.1 is that convergence to a given new state region P_2 from another region P_1 is guaranteed after reconfiguration. Notice that definitions 9.1 and 9.2 characterize discrete-event mechanisms without external control. This may result in (36) being rather restrictive for system design. A weaker form of attraction obtained under supervision is given next.

9.3. *Region of weak attraction*

Definition 9.3: Let $P_2 \leqslant P_1$ where P_1 and P_2 are two predicates on $\mathscr{X}$. Then, P_2 is a *concentric weak attractor* for P_1 with respect to the plant M_c, denoted as $P_2 \xleftarrow{M_c} P_1$, if there exists a supervisor M_s such that, for the closed-loop system M_c/M_s, P_2 is a concentric strong attractor of P_1; i.e. $P_2 \overset{M_c/M_s}{\Longleftarrow} P_1$ □

Remark 9.1: It follows from Definition 9.3 that a predicate can become a strong attractor for another predicate if suitable control inputs are generated under supervision. Therefore, strong attraction implies weak attraction. □

Proposition 9.2: *Let $P_2 \leqslant P_1$ where P_1 and P_2 are two predicates on $\mathscr{X}$ such that P_2 is Σ_u-invariant. Then, P_2 can be a concentric weak attractor for P_1 if there exists a supervisor M_s such that the following conditions hold*

(i) $R_e(M_c/M_s, P_1) \leqslant R_e^{-1}(M_c/M_s, P_2)$

(ii) $(\forall x \in \mathscr{X}$ *with* $(\neg P_2 \wedge R_e(M_c/M_s, P_1)).x = 1) f(x, s) = x \Rightarrow s = \varepsilon$

where ε is the empty event sequence.

Proof: Condition (i) ensures that any state reachable from P_1 is attractable to P_2. Therefore, any state trajectory leaving P_1 can then be dragged to P_2. Because it has been assumed that P_2 is Σ_u-invariant, trajectories entering P_2 will remain there under any sequence of uncontrollable events. Condition (ii) guarantees that for any state reachable from P_1 but not in P_2, there exists no sequence of events other than the empty string that causes no state changes. This ensures that the system can eventually converge to P_2. □

Corollary to Proposition 9.2: *If all events are controllable, then the necessary and sufficient condition needed to ensure convergence is given by*

$$P_1 \leqslant R_e^{-1}(M_c/M_s, P_2) \tag{37}$$

Proof: Equation (37) indicates that for any state in P_1, there exists a state trajectory leading to P_2. If $\Sigma_c = \Sigma$, all such trajectories are driven by sequences of controllable events. Therefore, it is guaranteed that a supervisor can exert complete control on the plant to drive the system directly from P_1 to P_2. In addition, notice that if $\Sigma_c = \Sigma$, a proper supervision can avoid cycles in the state space. □

Remark 9.2: Proposition 9.2 establishes the necessary conditions for a predicate P_2 to become a strong attractor for another P_1 under supervision. Notice that, unlike strong attraction, Σ_u-invariant is a necessary condition for weak attraction. □

Similar to the case of strong attraction, it is of interest to find the largest predicate for which a given predicate P is a weak attractor. To this end, the following theorem is given.

Theorem 9.2: *Let $WA(M_c, P)$ denote the class of all predicates for which P is a weak attractor; i.e.*

$$WA(M_c, P) := \{Q \in : P \leqslant Q, P \xleftarrow{M_c} Q\}$$

Then, $WA(M_c, P)$ has a maximal element.

Proof: It suffices to show that $WA(M_c, P)$ is non-empty and closed under predicate disjunction. First, notice that for any predicate P, the smallest region of weak attraction of P is P itself. Therefore, $WA(M_c, P)$ is non-empty. To prove that $WA(M_c, P)$ is closed under disjunction of predicates, it must be shown that

$$(\forall Q_1, Q_2 \in WA(M_c, P))(Q_1 \vee Q_2) \in WA(M_c, P) \tag{38}$$

To this end, $(Q_1 \vee Q_2)$ must satisfy conditions (i) and (ii). To verify (i), notice that if $Q_1, Q_2 \in WA(M_c, P)$ then there must exist supervisors M_{s_1} and M_{s_2} such that the following inequality holds

$$R_e(M_c/M_{s_i}, Q_i) \leqslant R_e^{-1}(M_c/M_{s_i}, P) \tag{39}$$

for $i = 1, 2$. Let $u_{12}(\cdot)$ denote the control input policy enforced by an extended supervisor $M_{s_{12}}$ defined for $(Q_1 \vee Q_2)$ with

$$u_{12}(x) = u_1(x) \cup u_2(x) \tag{40}$$

Equation (40) implies that

$$R_e(M_c/M_{s_{12}}, Q_1 \vee Q_2) = R_e(M_c/M_{s_1}, Q_1) \vee R_e(M_c/M_{s_2}, Q_2) \tag{41}$$

$$R_e^{-1}(M_c/M_{s_{12}}, P_1 \vee P_2) = R_e^{-1}(M_c/M_{s_1}, P_1) \vee R_e^{-1}(M_c/M_{s_2}, P_2) \tag{42}$$

Thus, from (39)–(42), it follows that

$$R_e(M_c/M_{s_{12}}, Q_1 \vee Q_2) \leqslant R_e^{-1}(M_c/M_{s_{12}}, P)$$

and condition (i) is satisfied. To verify (ii), assume first that $\neg P \wedge R_e(M_c/M_{s_{12}}, Q_1 \vee Q_2)$ is originally cyclic and find the conditions on supervisors' synthesis that guarantee the terms given by (i) and (ii). Thus

$$\exists s \in \Sigma^*, x \in \mathscr{X} \quad \text{subject to } (\neg P \wedge R_e(M_c/M_{s_{12}}, Q_1 \vee Q_2)).x = 1, f(x,s) = x$$

Let Π_{12} denote an instance of such a cycle with $\Pi_{12}(i) = f(\Pi_{12}(i-1), \sigma_i)$, $\Pi_{12}(1) = \Pi_{12}(|\Pi_{12}|)$ and $|\Pi_{12}|$ denoting the length of Π_{12}. Without loss of generality, assume that Π_{12} starts in the region reachable from Q_1 under the supervision of M_{s_1} and outside of P; i.e.

$$(\neg P \wedge R_e(M_c/M_{s_1}, Q_1)).\Pi_{12}(1) = 1 \tag{43}$$

Because $Q_i \in WA(M_c, P)$ for $i = 1, 2$, then there exists no cycle either in the region defined by $\neg P \wedge R_e(M_c/M_{s_1}, Q_1)$ nor in $\neg P \wedge R_e(M_c/M_{s_1}, Q_2)$. This and (43) imply that $\exists i \in \{1, \ldots, |\Pi_{12}|\}$ such that

$$(\neg P \wedge R_e(M_c/M_{s_1}, Q_1) \wedge \neg R_e(M_c/M_{s_2}, Q_2)).\Pi_{12}(i) = 1 \tag{44}$$

and

$$(\neg P \wedge R_e(M_c/M_{s_2}, Q_2) \wedge \neg R_e(M_c/M_{s_1}, Q_1)).\Pi_{12}(i+1) = 1 \tag{45}$$

Because $\Pi_{12}(i+1)$ follows $\Pi_{12}(i)$, there must exist an event σ_{i+1} such that

$$\Pi_{12}(i+1) = f(\Pi_{12}(i), \sigma_{i+1}) \tag{46}$$

Equations (44)–(46) result in two conclusions. First, σ_{i+1} must be a controllable event. If σ_{i+1} were an uncontrollable event, $R_e(M_c/M_{s_1}, Q_1).\Pi_{12}(i+1) = 1$, which would contradict (45). Secondly, to satisfy (44) and (45), the following condition must hold

$$\sigma_i \in u_1(\Pi_{12}(i-1)) \quad \text{and } \sigma_{i+1} \in u_2(\Pi_{12}(i)) - u_1(\Pi_{12}(i)) \tag{47}$$

Because (47) can be possible, $\neg P \wedge R_e(M_c/M_{s_{12}}, Q_1 \vee Q_2)$ is not necessarily acyclic and condition (ii) is not surely satisfied. To meet (ii), the cycle Π_{12} must be eliminated, which can be achieved by removing σ_{i+1} from $u_{12}(\cdot)$. That is, if the control law is redefined as

$$\hat{u}_{12}(x) = \begin{cases} (u_1(x) \cup u_2(x)) - \sigma_{i+1} & \text{if } x = \Pi_{12}(i) \\ u_1(x) \cup u_2(x) & \text{otherwise} \end{cases} \tag{48}$$

for a new supervisor $\hat{M}_{s_{12}}$, then Π_{12} does not exist in $\neg P \wedge R_e(M_c/\hat{M}_{s_{12}}, Q_1 \vee Q_2)$. Assume Π_{12} was the only (original) cycle in $\neg P \wedge R_e(M_c/\hat{M}_{s_{12}}, Q_1 \vee Q_2)$. Thus, $\hat{u}_{12}(\cdot)$ as given by (48) will ensure fulfilment of (ii). Let us verify that this modification of the control law from $u_{12}(\cdot)$ to $\hat{u}_{12}(\cdot)$ has not affected the validity of condition (i). To this end, notice that from (46) the elimination of σ_{i+1} might affect only the connectivity of $\Pi_{12}(i)$ and any other state connected to P through $\Pi_{12}(i)$. However, from (39) and (47), there must exist a state trajectory from $\Pi_{12}(i)$ to P that does not travel σ_{i+1}. Therefore,

$$R_e(M_c/\hat{M}_{s_{12}}, Q_1 \vee Q_2) \leqslant R_e^{-1}(M_c/\hat{M}_{s_{12}}, P)$$

and condition (i) is satisfied. Equation (38) is then proved under $\hat{u}_{12}(\cdot)$. If there exists other cycles in $\neg P \wedge R_e(M_c/M_{s_{12}}, Q_1 \vee Q_2)$ besides Π_{12}, this procedure is repeated for each of them and, following a similar argument, the resulting supervisor will ensure fulfilment of conditions (i) and (ii) hence of (38) under $\hat{u}_{12}(\cdot)$. □

Corollary to Theorem 9.2: *Let $WA(M_c, P)$ denote the class of all predicates for which P is a concentric weak attractor. Then, a maximal element of $WA(M_c, P)$ exists, denoted by $WA_{\max}(M_c, P)$, such that*

$$\neg\exists Q \in WA(M_c, P) \quad \text{subject to } WA_{\max}(M_c, P) < Q \tag{49}$$

Proof: Let Q_1 and Q_2 denote any two feedbacks such that $Q_1, Q_2 \in WA(M_c, P)$. Theorem 9.2 shows that $WA(M_c, P)$ is closed under disjunction of predicates. Therefore, $(Q_1 \vee Q_2) \in WA(M_c, P)$. Notice that for $i = 1, 2, Q_i \leqslant (Q_1 \vee Q_2)$. Then, $(Q_1 \vee Q_2)$ is the maximal element for the set $\{Q_1, Q_2\} \subseteq WA(M_c, P)$. If there exists another element $Q_3 \in WA(M_c, P)$ different from Q_1 and Q_2, apply the same reasoning to get $(Q_1 \vee Q_2 \vee Q_3)$ which is the maximal element for $\{Q_1, Q_2, Q_3\}$. The procedure above is repeated until all elements of $WA(M_c, P)$ are considered to end with a unique maximal element. □

Definition 9.4: The maximal element satisfying (49) is called the *ball of weak predicate attraction* of P with respect to M_c. □

If P is not Σ_u-invariant, then $WA_{\max}(M_c, P) = \emptyset$. In this case, the maximal Σ_u-invariant sub-predicate of P is found to replace P in the above equations. Convergence from a given region P_1 to another P_2 can now be ensured under supervision as indicated next.

Proposition 9.3: *A target region defined by a given predicate P_2 can be reached from another region P_1, under suitable supervision, if the latter is evolved by the ball of weak attraction of the former; that is*

$$P_1 \leqslant WA_{\max}(M_c, P_2) \tag{50}$$

Proof: From Theorem 9.2 and Definition 9.4, it is implied that, for any predicate $Q \in WA(M_c, P)$, P is a weak attractor for Q. In addition, (50) indicates that $P_1 \in SA(M_c, P_2)$. This and Theorem 9.2 result in P_2 being a weak attractor for P_1 asserting that P_2 can be reached from P_1 under proper supervision. □

Thus, if, when the predicates P_1 and P_2 are being defined, (50) is enforced to be valid, convergence to the new state region defined by P_2 from another defined by P_1 can be achieved after reconfiguration, assuming that a suitable set of control inputs is commanded on the controlled process. This argument is summarized in the following synthesis algorithm.

Synthesis algorithm 9.3.1

Let P_1 and P_2 denote the predicates specified to be enforced whenever the closed-loop system is characterized by the plant/supervisor pairs M_c/M_{s_1} and M_c/M_{s_2}, respectively. If the coordinator decides a reconfiguration from supervisor M_{s_1} to M_{s_2}, then the plant is needed to be moved from P_1 to the region defined by P_2. Therefore, convergence to P_2 from P_1 must be guaranteed. To ensure this convergence, an algorithm can be devised that enforces (50) to be valid. Specifically, if the given predicates P_1 and P_2 satisfy (50), Proposition 9.3 ensures that a supervisor exists that can drag any state in P_1 to a state in P_2.

10. A concept of stochastic modelling approach for specifying the supervisory tasks

To describe a given process, its discrete event representation M can explicitly incorporate modelling features reflecting the stochastic nature of the underlying system. With respect to the supervisory tasks, it may be beneficial to model explicitly the stochastic characteristics of event occurrences. For the discrete event model discussed so far, an event can be characterized based on the possibility of its occurrence using the enabling function $d(x)$ or, similarly, using the set-valued functions $u(x)$ or $w(x)$ defined in §3. For example, it is known with certainty that an uncontrollable event σ will not occur at x if $\sigma \notin w(x)$. However, for any $\sigma \in w(x)$, it cannot be guaranteed that σ will actually fire. The current definition of $w(\cdot)$ does not provide for a continuum measure of confidence for event occurrence.

It is important for many practical implementations to distinguish among events that may occur at a given state x from those that will certainly not occur at x. This might be useful when relaxing some of the controllability constraints imposed by the mentioned supervisory tasks. For example, Task I (control invariance) requires that the specified predicate be Σ_u-invariant. This condition may be difficult to meet in general. However, the designer may be content if it is satisfied within a certain degree of confidence. To incorporate the above ideas, the discrete event model presented in §3 is extended as follows.

Let Prob$(\cdot, \cdot)$ denote a mapping from the set of events and states to the real line segment between 0 and 1; i.e. Prob: $\Sigma \times \mathscr{X} \rightarrow [0, 1]$. For a given event σ, Prob(σ, x) indicates the probability that σ may occur at the current state x. Thus, the definition of $d(x)$ given in (5) can be rewritten as follows

$$d(x) := \{\sigma \in \Sigma \colon \mathrm{Prob}(\sigma, x) > 0\} \tag{51}$$

Notice that from (51), $(\forall \sigma \notin d(x))\, \mathrm{Prob}(\sigma, x) = 0$ and $\mathrm{Prob}(\sigma, x) = 1$ for a controllable event σ forced to occur at x by the supervisor.

Definition 10.1: Let P denote a given predicate, x a given state satisfying P, and p a real number between 0 and 1. P is said to be *Σ_u-invariant at x with probability p* if and only if

$$(\forall \sigma \in w(x) \colon \mathrm{sp}.f_\sigma.P > P)\, \mathrm{Prob}(\sigma, x) \leqslant (1-p) \qquad \square$$

Definition 10.2: A predicate P is said to be *Σ_u-invariant with probability p* if, for any state x satisfying P, P is Σ_u-invariant at x with probability p. $\square$

Definition 10.1 says that for all uncontrollable events that might occur at a given state x and push the state of the system out of P, their probability of occurrence is less than or equal to $(1-p)$. For $p = 1$, Definition 10.2 reduces to the standard definition of Σ_u-invariance (Ramadge and Wonham 1987).

The function Prob$(\cdot, \cdot)$ can then be used when specifying the supervisory tasks. For example, if the statistics of event occurrences is known, the Σ_u-invariance condition imposed by Task I can be relaxed by requiring that the given predicate P be Σ_u-invariant with probability p. In general, this condition can be more easily met and still be acceptable for certain supervisor designs. Extensions of these ideas to the other supervisory tasks can also be formulated. However, this is a subject of future research and is not addressed in this paper.

11. Summary and conclusions

Reconfiguration refers to the capability of changing system configuration based on operational conditions. It is applied to control processes with time-varying specifications or with large operational changes. In reconfigurable supervisory systems, the sets of supervisory specifications or operating discrete event processes, could be time-dependent. This paper proposes a reconfigurable supervisory approach to control plants subjected to unknown disturbances, including time-varying discrete event systems (DES). In a reconfigurable approach, the controlled DES process along with its operating conditions are partitioned into sets of subprocesses and operating regimes, and a supervisor is devised for each pair of subprocess and subprocess operating conditions. The current operating condition of the plant is identified, and a supervisor is selected by a high level decision-maker (e.g. a coordinator) among the available ones to act directly on the plant. Issues concerning the class of discrete event system resulting from reconfigurable supervision, named reconfigurable DES (RDES), are addressed in this paper. In particular, the paper addresses basic ideas of supervisory control in the state-space framework and discusses analytical guidelines related to RDES implementations. These implementations are particularly important in reconfigurable hierarchical hybrid control systems (Garcia 1993, Garcia and Edwards 1993a, Garcia *et al.* 1995).

A structured discrete event framework is proposed and discussed in this paper. The proposed model possesses a discrete finite-dimensional space generated by a finite set of state variables with control specifications given in terms of predicates on the set of states and system dynamics expressed as predicate transformations. To modify the open loop response of a given DES, event/state feedback control is employed where supervisors act on observed event or state signals received from the plant through event and condition channels. In addition, the controllable actions are logically separated from the uncontrollable disturbances. This results in a closed-loop configuration that is consistent with the classical control theory for continuously varying processes and facilitates analysis and synthesis of algorithms with a special emphasis on process control applications.

Besides the simple feedback arrangement, this paper presents an architecture of RDES where a bank of supervisors is identified for each controlled DES, with one and only one supervisor acting at any instant of time. Each supervisor is designed to accommodate a restricted subset of all possible plant operational conditions. Switching among supervisors is based on the supervisory command or on the performance of the overall process. Changes in the assignment of supervisors may introduce transients that are designed to decay sufficiently fast so that the new desired behaviour is achieved. To this effect, three supervisory tasks are identified: control invariance, region avoidance and convergence. The control invariance task requires that, after an event firing sequence, a specified predicate remains invariantly satisfied whenever initially satisfied. The region avoidance task requires that the system must not satisfy undesirable predicates when traversing the state space. Finally, the convergence task requires the system to converge to a specified target predicate from given initial conditions. This paper defines and discusses the methodology and algorithms for these tasks. Computationally efficient solutions to these tasks have been identified from cited references.

The potential of driving a controlled DES from arbitrary initial states to a specified target region is of interest in a more general framework. To this end, the issue of convergence under the proposed framework has been investigated. A stochastic setting for supervisory task specifications is also outlined. Experimental evaluations of the ideas presented in this paper have been partially reported by Garcia (1993), Garcia and Edwards (1993a), Garcia *et al.* (1995). Additional experimental work is needed and tests are currently being planned for future research.

ACKNOWLEDGMENT

The research work reported in this paper was supported in part by the Department of Energy under Grant No. DE-FG-07-89ER 12889 and under contract W-31-109-Eng-38.

Appendix

Predicate transformers. Predicate transformers were introduced by Dijkstra and Scholten (1990) to define precedent or successive predicative conditions with a special emphasis on computer semantics. A predicate transformer f maps a family $\mathscr{P}$ of predicates into itself. To express these transforming operations under a given operation, the following notation (Dijkstra and Scholten 1990) is used

$$Q = (f.\mathrm{Op}).P \equiv f.\mathrm{Op}.P$$

where

P is a given predicate

Op is a specified (process) operation or execution

f is a defined predicate transformer (e.g. wp, wlp, or sp)

Q is the resulting predicate after transformation of P under $f.\mathrm{Op}$

Therefore, $f.\mathrm{Op}: \mathscr{P} \to \mathscr{P}$ is a pre or post-conditional predicate transformer under the operation Op. To study the dynamics of certain DES, the predicate transformers, wp, wlp, and sp, were introduced to the DES arena by Ramadge and Wonham (1987), Wonham and Ramadge (1988) where the specific computational operation is given by the event transition function f_σ. While $\mathrm{wp}.f_\sigma.P$ is the weakest precondition under which f_σ is guaranteed to establish the post condition P; $\mathrm{wlp}.f_\sigma.P$ is the weakest precondition under which f_σ is guaranteed to establish the post condition P if the event σ fires. On the other hand, $\mathrm{sp}.f_\sigma$ is the strongest post condition under which f_σ is guaranteed to establish the post condition $\mathrm{sp}.f_\sigma.P$ whenever started from P. Formally, $\mathrm{wlp}.f_\sigma.P$ holds in those states where each computation of f_σ results in a state that satisfies P, or for which f_σ is undefined; i.e.,

$$\mathrm{wlp}.f_\sigma.P = Q, \quad \text{where} \quad \mathscr{X}_Q = \{x \in \mathscr{X} : P.f_\sigma(x) = 1 \vee \sigma \notin d(x)\}$$

and $d(\cdot)$ is defined in (5). Similarly, $\mathrm{wp}.f_\sigma.P$ holds only in those states where each computation of f_σ results in a state that satisfies P; i.e.,

$$\mathrm{wp}.f_\sigma.P = Q, \quad \text{where} \quad \mathscr{X}_Q = \{x \in \mathscr{X} : P.f_\sigma(x) = 1\}$$

On the other hand, $\mathrm{sp}.f_\sigma.P$ holds in those states for which there exists a computation that starts in an initial state satisfying P; i.e.,

$$\mathrm{sp}.f_\sigma.P = Q, \quad \text{where } \mathscr{X}_Q = \{x \in \mathscr{X} : \exists y \in \mathscr{X}_P \text{ subject to } f_\sigma(y) = x\}$$

Proposition A.1: $\mathrm{wlp}.f_\sigma$ *is conjunctive.*

Proof: It is required to show that

$$(\forall P, Q \in \mathscr{P})\, \mathrm{wlp}.f_\sigma.P \wedge \mathrm{wlp}.f_\sigma.Q = \mathrm{wlp}.f_\sigma.(P \wedge Q) \qquad \text{(A 1)}$$

The proof is done in two steps.

Step 1

Show that $\text{wlp}.f_\sigma.P \wedge \text{wlp}.f_\sigma.Q \leqslant \text{wlp}.f_\sigma.(P \wedge Q)$.

The proof is by contradiction. Assume that

$$\exists x \in \mathscr{X}: (\text{wlp}.f_\sigma.P \wedge \text{wlp}.f_\sigma.Q).x = 1 \quad \text{and} \quad \text{wlp}.f_\sigma.(P \wedge Q).x = 0$$

By definition, if $\text{wlp}.f_\sigma.P.x = 1$ and $\text{wlp}.f_\sigma.Q.x = 1$, this indicates that after the firing of σ, the resultant image of x satisfies P and Q or f_σ fails to terminate. However, this contradicts the assumption that $\text{wlp}.f_\sigma.(P \wedge Q).x = 0$.

Step 2

Show that $\text{wlp}.f_\sigma.(P \wedge Q) \leqslant \text{wlp}.f_\sigma.P \wedge \text{wlp}.f_\sigma.Q$.

The proof is by contradiction. Assume that

$$\exists x \in \mathscr{X}: \text{wlp}.f_\sigma.(P \wedge Q).x = 1 \quad \text{and} \quad (\text{wlp}.f_\sigma.P \wedge \text{wlp}.f_\sigma.Q).x = 0$$

By definition, $\text{wlp}.f_\sigma.(P \wedge Q).x = 1$ indicates that after the firing of σ, the resulting image of x satisfies P and Q or f_σ fails to terminate. However, this contradicts our assumption that $\text{wlp}.f_\sigma.P.x = 0$ and $\text{wlp}.f_\sigma.Q.x = 0$. From Steps 1 and 2, the proof is completed. □

Proposition A.2: $(\forall P, Q \in \mathscr{P})\, \text{wlp}.f_\sigma.P \vee \text{wlp}.f_\sigma.Q \leqslant \text{wlp}.f_\sigma.(P \vee Q)$

Proof: First, notice that $(\forall P, Q \in \mathscr{P})$ $P \leqslant P \vee Q$ and $Q \leqslant P \vee Q$. From Proposition A.1, it is known that $\text{wlp}.f_\sigma$ is conjunctive. Because $\text{wlp}.f_\sigma$ is conjunctive, it has been proved (Ushio 1989) that $\text{wlp}.f_\sigma$ is strict and monotone. Using monotonicity property, it can be concluded that $(\forall P, Q \in \mathscr{P})\, \text{wlp}.f_\sigma.P \leqslant \text{wlp}.f_\sigma.(P \vee Q)$ and $\text{wlp}.f_\sigma.Q \leqslant \text{wlp}.f_\sigma.(P \vee Q)$. These two inequalities indicate that $(\forall P, Q \in \mathscr{P})\, \text{wlp}.f_\sigma.P \vee \text{wlp}.f_\sigma.Q \leqslant \text{wlp}.f_\sigma.(P \vee Q)$ and the proof is completed. □

Reachable and attractable predicates. Let us denote the disjunctive and conjunctive DO-loops predicate transformations as follows

$$g.DO(f_\sigma)^k := \bigvee_{\substack{s \in \Sigma^* \\ |s| = k}} g.f_s \quad \text{and} \quad g.OD(f_\sigma)^k := \bigwedge_{\substack{s \in \Sigma^* \\ |s| = k}} g.f_s$$

with g denoting a given predicate transformer, s denoting a given event sequence of length $|s|$ and f_s denoting the transition function extended for s.

Definition A.1: For a discrete event mechanism M, the reachable predicate generated after k (or less) event transitions, starting from a given predicate P, is given as

$$R_e(M, P, k) := \bigvee_{0 \leqslant i \leqslant k} \text{sp}.DO(f_\sigma)^i.P$$

If the number of transition steps is not bounded, the generated predicate becomes

$$R_e(M, P) := \bigvee_{k \geqslant 0} R_e(M, P, k)$$ □

Definition A.2: For a DES M, the attractable predicate denoted by $R_e^{-1}(M, P, k)$ from which a given predicate P is reachable in no more than k steps is defined as

$$R_e^{-1}(M, P, k) := \bigvee_{0 \leqslant i \leqslant k} \text{wlp}.OD(f_\sigma)^i.P$$

If there is no bound on k, then the above definition is modified as

$$R_{\mathrm{e}}^{-1}(M, P) := \bigvee_{i \geqslant 0} \mathrm{wlp} . OD(f_\sigma)^i . P \qquad \square$$

References

Brave, Y., and Heymann, M., 1989, On stabilization of discrete-event processes. *Proceedings of the 28th Conference on Decision and Control*, Tampa, Florida, U.S.A., December 1989, pp. 2737–2742; 1940, Stabilization of discrete-event processes. *International Journal of Control*, **51**, 1101–1117.

Cao, X. R., 1989, A comparison of the dynamics of continuous and discrete event systems. *Proceedings of the IEEE*, **77**, 7–13.

Chung, S. L., Lafortune, S., and Lin, F., 1992, Limited lookahead policies in supervisory control of discrete event systems. *IEEE Transactions on Automatic Control*, **37**, 1921–1935; 1993, Recursive computation of limited lookahead supervisory controls for discrete event systems. *Discrete Event Dynamic Systems: Theory and Applications*, **3**, 71–100.

Dijkstra, E. W., and Scholten, C. S., 1990, *Predicate Calculus and Program Semantics* (New York: Springer-Verlag).

Garcia, H. E., 1993, A reconfigurable hierarchical hybrid supervisory control system. Ph.D. dissertation, Pennsylvania State University, Pennsylvania.

Garcia, H. E., and Edwards, R. M., 1993a, In-plant results of a reconfigurable fault-tolerant system. IDCRL Report, Pennsylvania State University, University Park, Pennsylvania; 1993b, On-line performance; feedback for control system reconfiguration. IDCRL Report, Pennsylvania State University, University Park, Pennsylvania.

Garcia, H. E., and Ray, A., 1992, Reconfigurable discrete event system for process control. Presented at *SIAM in Control*, Minneapolis, Minnesota, 15–17 September, 1992.

Garcia, H. E., Edwards, R. M., and Ray, A., 1991a, On-line diagnostics and intelligent control for the steam plant at EBR-II. AI91 Frontiers in Innovative Computing for the Nuclear Industry, 15–18 September 1991.

Garcia, H. E., Ray, A., and Edwards, R. M., 1990, A reconfigurable strategy for distributed digital process control. *Proceedings of the Sixth Yale Workshop on Adaptive and Learning Systems*, 15–17 August, Yale University, Connecticut, U.S.A., pp. 184–188; 1991b, Reconfigurable control of power plants using learning automata. *IEEE Control Systems Magazine*, **11**, 85–92; 1995, A reconfigurable hybrid system and its application to power plant control. *IEEE Transactions on Control Systems Technology*, **3**(2), 157–170.

Garg, V. K., and Kumar, R., 1992, A state-variable approach for controlling discrete event systems. *Proceedings of the American Control Conference*, pp. 2809–2813.

Gollu, A., and Varaiya, P., 1989, Hybrid dynamical systems. *Proceedings of the 28th Conference on Decision and Control*, Tampa, Florida, U.S.A., pp. 2708–2712.

Hopcroft, J. E., and Ullman, J. D., 1979, *Introduction to Automata Theory, Languages and Computation* (Reading, Mass: Addison Wesley).

Kaufmann, A., and Arnaud Henry-Labordere, A., 1977, *Integer and mixed programming, theory and applications* (New York: Academic Press).

Kumar, R., Garg, V., and Marcus, S. I., 1993, Predicates and predicate transformers for supervisory control of discrete event dynamic systems. *IEEE Transactions on Automatic Control*, **38**(2), 232–247.

Lapin, L., 1985, *Quantitative Methods for Business Decisions, with Cases*, third edition (San Diego, Calif.: Harcourt Brace Jovanovich).

Li, Y., 1991, Control of vector discrete-event systems. System Control Group Report No. 9106, University of Toronto, Canada.

Li, Y., and Wonham, W. M., 1988a, Controllability and observability in the state-feedback control of discrete-event systems. *Proceedings of the 27th Conference on Decision and Control*, Austin, Texas, U.S.A., pp. 203–208; 1988b, A state-variable approach to the modeling and control of discrete-event systems. *Proceedings of the 2nd Allerton Conference on CCC*, pp. 1140–1149.

Lin, F., 1992, Robust and adaptive supervisory control of discrete event systems. *Proceedings of American Control Conference*, Chicago, Illinois, U.S.A., pp. 2804–2808.

Lin, F., and Wonham, W. M., 1990, Decentralized control and coordination of discrete-event systems with partial observation. *IEEE Transactions on Automatic Control*, **35**, 1330–1337.

Narendra, K., and Thathachar, M. A. L., 1989, *Learning Automata: an Introduction* (Englewood Cliffs, N.J.: Prentice Hall).

Passino, K. M., and Michel, A. N., 1992, Stability and boundedness analysis of discrete event systems. *Proceedings of American Control Conference*, Chicago, Illinois, U.S.A., 3201–3205.

Passino, K. M., Michel, A. N., and Antsaklis, P. J., 1991, Lyapunov stability of a class of discrete event systems. *Proceedings of American Control Conference*, Boston, Massachusetts, U.S.A., pp. 2911–2916.

Ramadge, P. J., and Wonham, W. M., 1987, Modular feedback logic for discrete-event systems. *SIAM Journal on Control and Optimization*, **25**, 1202–1218; 1989, The control of discrete-event systems. *Proceedings of the Institute of Electrical and Electronics Engineers*, **77**, 81–98.

Sreenivas, R. S., and Krogh, B. H., 1991, On condition/event systems with discrete state realizations. *Discrete Event Dynamics Systems: Theory and Applications* **1**, 209–236.

Ushio, T., 1989, Controllability and control-invariance in discrete-event systems. *International Journal of Control*, **50**, 1507–1515.

Wonham, W. M., and Ramadge, P. J., 1987, On the supremal controllable sublanguage of a given language. *SIAM Journal of Control and Optimization*, **25**, 637–659; 1988, Modular supervisory control of discrete-event systems. *Mathematics of Control, Signals, and Systems*, **1**, 13–30.