Organizational Control of Discrete-Event Systems: A Hierarchical Multi-World Supervisor Design

Kiam Tian Seow, *Senior Member, IEEE*

***Abstract*— An organizational control architecture for supervising a class of multi-level hierarchical discrete-event systems is proposed in this paper. The architecture can be built based on a standard, scalable hierarchical design method, formalizing a common design practice of structuring the control of a discrete-event organization bottom-up into a consistent multi-world control hierarchy. It is shown that under some mild condition of fairness, a multi-level recursive control law exists that is optimal and nonblocking. This law governs the hierarchy top-down as a dynamic programming recursion, over which an organizational control algorithm is obtained that computes the control decisions partially on-line, and in linear time. It is explained and illustrated how the approach reduces the complexity of off-line control synthesis and increases the on-line transparency of control operations.**

***Index Terms*— Discrete-event systems, hierarchical supervisory control, dynamic programming and recursion, automata.**

I. Introduction

In discrete-event system (DES) research, one major problem is the complexity of control [2], [3] that renders synthesis intractable or the resulting implementation non-transparent for large DES's. To mitigate this problem, an intuitive strategy is to reorganize a DES so that it becomes amenable to hierarchical control. Basic research in hierarchical control (e.g., [4], [5], [6]) has been concerned with the synthesis of a low-level supervisor for a two-level hierarchy. In this set-up, the high-level system is an aggregated model of the low-level process, and is driven by the latter via an information channel. Other strategies or architectures investigated include modular (e.g., [7], [8]) and decentralized control (e.g., [9], [10], [11], [12]).

To reduce control complexity, we propose a new multi-level architecture called organizational control for supervising a class of hierarchical DES's, in a manner reminiscent of commanding and controlling human organizations. Organizational control involves managing the interrelated command and control decisions of event enablement or disablement between levels in a multi-level control hierarchy. Specifically, for a class of modular DES's modeling (hierarchical) organizations, we formulate and implement the multi-level control architecture that can effectively ensure qualitative optimization of maximal permissiveness at every level, using recursion that supports a scalable hierarchical design method [3].

Our research on organizational control is based on a bottom-up design philosophy. Prior bottom-up approaches to hierarchical control have made important progress by developing different scalable design methods for achieving nonblocking (or marked) hierarchical consistency. These methods use suitably formulated hierarchical information maps for the standard DES model [3], the DES model with flexible marking [6] and that with multitasking [13]. Intuitively, achieving marked hierarchical consistency means that, in a two-level hierarchy, the nonblocking specification task of the high-level controller can be realized through nonblocking control implemented at the low level. Two types of event reporting for hierarchical maps have been proposed, namely, virtual [3] and natural projections [6], [13]. When scaling up to multiple levels, we posit that the former type induces a multi-world hierarchy - one where each hierarchical level represents a different world. Different worlds are modeled by different disjoint event sets as follows: The base level is modeled by only real events; the top level by only virtual events, i.e., events abstracted or projected from the level below; and an intermediate level is modeled by both real and virtual events. The latter type induces a single-world hierarchy - one where successively higher levels are modeled by successively smaller subsets of real events modeling the system world.

For the single-world paradigm [6], [13], recent efforts include attempts to leverage on modularity and decentralization of DES's in a hierarchical structure, proposing or enhancing different hierarchical design methods for nonblocking control synthesis (e.g., [14], [15], [16], [17], [13], [18]). All aim at reducing the computational effort using different methods of subsystem synthesis. Like their centralized versions, the resultant controllers residing at the same or different levels in a hierarchical architecture are realized by non-recursive action implementations. Unlike a centralized controller that acts on the overall system, each of these modular or decentralized controllers generally exercises its control decisions only on that part of the DES that matches its event set.

For the multi-world paradigm, the well-established bottom-up design framework [3], [4], [5], [19], [20] currently synthesizes the final controller as a flat, non-recursive action implementation, and does so from a base-level specification. The base-level specification is first obtained by recursively compiling the constraint specifications from higher levels and vertically collapsing them onto the base level. However, besides higher complexity of control synthesis, such base-level action implementation for a multi-world hierarchy cannot provide transparency (of the relationship) of on-line or runtime control operations between levels. The multi-world paradigm can and should facilitate this, in the sense of making explicit what stepwise control decisions are needed to carry out a specific command decision (i.e., control decision on a virtual event) from immediately above. For homogeneous systems

This paper revises and extends the preliminary conference version [1].

K.T. Seow is with the School of Computer Engineering, Nanyang Technological University, Republic of Singapore 639798. asktseow@ntu.edu.sg.

designed in the single-world paradigm reviewed above, the natural solution is a non-recursive control action implementation and no such operational transparency is needed. In our opinion, such transparency can be useful when monitoring complex heterogeneous systems that have to, or are better to be designed as a command and control organization in the multi-world paradigm.

In this paper, a review on two-level hierarchical control is presented in Section II. A multi-world hierarchical control problem is described and formulated, and an organizational control architecture for the multi-world paradigm is proposed as a solution in Section III. In Section IV, it is then shown that the two-level hierarchical control design method reviewed can be incrementally applied to synthesize the organizational control architecture bottom-up into a consistent multi-world control hierarchy. Under some mild condition of fairness, an optimal and nonblocking multi-level control law is shown to exist, and can govern this hierarchy top-down as a dynamic programming recursion [21]. An algorithm for implementing the multi-level control law to run such a control organization is developed. Section V presents a discussion explaining the approach and the merits and limitations of organizational control, and where relevant, in relation to existing work. An example is then provided to explain the overall design approach and illustrate the on-line control operations of the algorithm in Section VI, followed by some concluding remarks in Section VII.

II. Background

Let Σ be a finite alphabet of symbols representing individual events. A string is a finite sequence of events. Denote Σ^* as the set of all strings with events from Σ. Let ε denote the empty string (sequence with no events). Trivially, $\varepsilon \in \Sigma^*$. A string s' is a prefix of s if $(\exists t \in \Sigma^*)s't = s$.

A formal language L over Σ is a subset of Σ^*. A language L_1 is said to be a sublanguage of L_2, if $L_1 \subseteq L_2$. The prefix closure $\overline{L}$ of L is the language consisting of all prefixes of strings of L, i.e., $\overline{L} = \{s \mid (\exists s')ss' \in L\}$. Clearly $L \subseteq \overline{L}$, and $\varepsilon \in \overline{L}$ provided $L \neq \emptyset$. A language L is called closed if $L = \overline{L}$.

For $\Sigma^1 \subseteq \Sigma^2$, the natural projection $P_{\Sigma^2,\Sigma^1} : \Sigma^{2,*} \to \Sigma^{1,*}$ is defined recursively as follows: $P_{\Sigma^2,\Sigma^1}(\varepsilon) = \varepsilon$ and $(\forall s \in \Sigma^{2,*})(\forall \sigma \in \Sigma^2)\ P_{\Sigma^2,\Sigma^1}(s\sigma) = P_{\Sigma^2,\Sigma^1}(s)\sigma$ if $\sigma \in \Sigma^1$, and $P_{\Sigma^2,\Sigma^1}(s\sigma) = P_{\Sigma^2,\Sigma^1}(s)$ otherwise. The definition of projection can be extended to $P_{\Sigma^2,\Sigma^1}(L) \subseteq \Sigma^{1,*}$ for $L \subseteq \Sigma^{2,*}$, as follows: $P_{\Sigma^2,\Sigma^1}(L) = \{P_{\Sigma^2,\Sigma^1}(s) \mid s \in L\}$.

A regular language [22] is a language that can be generated by an automaton with a finite state set. An automaton A is a 5-tuple $(Q, \Sigma, \delta, q_0, Q_m)$ where (i) Q is the finite set of states, (ii) Σ is the finite set of events, (iii) $\delta : \Sigma \times Q \to Q$ is the (partial, deterministic) transition function, (iv) q_0 is the initial state, and (v) $Q_m \subseteq Q$ is the subset of marked states. That an event $\sigma \in \Sigma$ is defined at a state $q \in Q$ is denoted by $\delta(\sigma, q)!$, and, for an event subset $\Sigma' \subseteq \Sigma$ and a state $q \in Q$, define $\Sigma'(q) = \{\sigma \in \Sigma' \mid \delta(\sigma, q)!\}$. The definition of δ can be extended to Σ^* as follows: $\delta(\varepsilon, q) = q$ and $(\forall \sigma \in \Sigma)(\forall s \in \Sigma^*)\delta(s\sigma, q) = \delta(\sigma, \delta(s, q))$, and is defined when both $q' = \delta(s, q)$ and $\delta(\sigma, q')$ are defined. Following, the behavior may then be described by two languages: $L(A) = \{s \in \Sigma^* \mid \delta(s, q_0)!\}$ and $L_m(A) = \{s \in L(A) \mid \delta(s, q_0) \in Q_m\}$. $L(A)$ is called the prefix-closed language and $L_m(A)$, the marked language. By definition, $L_m(A) \subseteq L(A)$.

A state $q \in Q$ is reachable (from the initial state q_0) if $(\exists s \in \Sigma^*)\delta(s, q_0) = q$, and coreachable if $(\exists s \in \Sigma^*)\delta(s, q) \in Q_m$. Automaton A is reachable if all its states are reachable, and coreachable if all its states are coreachable and so $\overline{L_m(A)} = L(A)$. Finally, automaton A is trim if it is both reachable and coreachable.

On 'equivalence' of two automata A_1 and A_2, we write $A_1 = A_2$ if their edge-labelled directed graphs are identical in structure (including marked states); and $A_1 \equiv A_2$ if the automata generate the same prefix-closed and marked languages. So $(A_1 = A_2) \overset{\text{(implies)}}{\Longrightarrow} (A_1 \equiv A_2)$ but the converse is not true in general.

The synchronous product $\|$ [23] of two automata A_1 and A_2 is the automaton $A = A_1 \| A_2$. The standard synchronous operator $\|$ is useful for modeling a complex DES as a modular system of interacting discrete-event automata. Using the same syntax $\|$ as synchronous operator over automata and languages [3], $L(A_1 \| A_2) = L(A_1) \| L(A_2)$ and $L_m(A_1 \| A_2) = L_m(A_1) \| L_m(A_2)$.

Finally, let $A_1 \sqcap A_2$ denote the cartesian operation between two automata A_1 and A_2, such that (i) $L(A_1 \sqcap A_2) = L(A_1) \cap L(A_2)$ and (ii) $L_m(A_1 \sqcap A_2) = L_m(A_1) \cap L_m(A_2)$. If $\Sigma_1 = \Sigma_2$, where Σ_i, $i = 1, 2$, is the event set of A_i, the synchronous operation $\|$ between A_1 and A_2 reduces to this cartesian operation $\sqcap$ between the two.

A. Local Level Supervisory Control

Consider a DES $G = (Q, \Sigma, \delta, q_0, Q_m)$, with the event set Σ partitioned into the controllable event set Σ_c and the uncontrollable event set Σ_u [2]. A local level specification language $K \subseteq \Sigma^*$ is said to be controllable [2] with respect to (w.r.t) G if $\overline{K}\Sigma_u \cap L(G) \subseteq \overline{K}$. This controllability condition complies with the fact that a supervisor that exists for DES G cannot physically disable an uncontrollable event, and so only the occurrence of any uncontrollable event always not exiting the bounds of $\overline{K}$ can guarantee non-violation of the specification K. If K is not controllable, there exists a supremal (or largest) controllable sublanguage of $K \cap L_m(G)$. This sublanguage can be generated by the trim automaton returned by $Supcon(G, K)$ [24].

The automaton $Supcon(G, K)$ is often larger in state size than is necessary to achieve the same control action because it has 'embedded' in it all the *a priori* transitional constraints embodied in DES G itself, as well as some auxiliary constraints. The trim automaton returned by a polynomial state reduction procedure called $Supreduce$ [25] is often a greatly state-reduced supervisor (automaton) $S = (X^S, \Sigma^S, \delta^S, x_0, -)$ [25] for G, fully achieving $L_m(S \sqcap G)$ such that $\Sigma^S = \Sigma$ and $S \sqcap G \equiv Supcon(G, K)$.

Any state-reduced supervisor S, including $Supcon(G, K)$ itself, is said to be nonblocking (for DES G) since

$\overline{L_m(Supcon(G,K))} = L(Supcon(G,K))$ for a trim and hence coreachable $Supcon(G,K)$.

Of interest is the associated control data set $Condat$ [3] for the state-reduced supervisor S.

$$Condat(G,S) = \{\ \Delta(q,x) \subseteq \Sigma_c \mid (\exists s \in \Sigma^*)\ \delta(s,q_0) = q \in Q \text{ and } \delta^S(s,x_0) = x \in X^S\ \},$$

where $\Delta(q,x) = \Sigma_c(q) - \Sigma_c^S(x)$. Intuitively, each data element $\Delta(q,x)$ specifies the controllable events to be disabled at a composite state $(q,x) \in Q \times X^S$ reachable by a common $s \in \Sigma^*$. By default, the events in $\Sigma(q) \cap \Sigma^S(x)$ are enabled at state (q,x). $\Delta(\delta(s,q_0),\delta^S(s,x_0))$ is also denoted by $Condat(G \sqcap S, s)$.

A control law for a DES G specifies a set of controllable events to be disabled following every input history $s \in L(G)$. Formally, a control law f is a function $f : L(G) \to 2^\Sigma$ with the constraint $(\forall s \in L(G))\,(\Sigma_u \cap \Sigma(\delta(s,q_0)) \subseteq (\Sigma - f(s)))$. The prefix-closed language that results from imposing f on G is denoted by $L(f,G)$ and defined as follows: $\varepsilon \in L(f,G)$ and $(\forall s \in L(f,G))(\forall s\sigma \in L(G))s\sigma \notin L(f,G) \Leftrightarrow \sigma \in f(s)$. The control law f is said to be nonblocking if $L(f,G) = \overline{L_m(f,G)}$, where $L_m(f,G) \subseteq L(f,G) \cap L_m(G)$ is defined such that when the law f is implementable by a supervisor S for the DES G according to $L(f,G) = L(S \sqcap G)$, $L_m(f,G) = L_m(S \sqcap G)$.

B. Two-level Hierarchical Control

Consider a two-level hierarchy consisting of a low-level DES G_{lo}^h and low-level supervisor S_{lo}^h, coupled with a high-level DES G_{hi}^{h+1} and high-level supervisor S_{hi}^{h+1}, as shown in Fig. 1. The DES and supervisor at each level are interconnected in a standard feedback fashion. G_{hi}^{h+1} is an abstract model of G_{lo}^h, and the high-level supervisor S_{hi}^{h+1} is to be completely implemented or realized by the low-level supervisor S_{lo}^h.

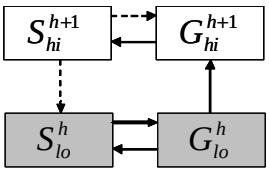


Fig. 1. Two-level hierarchy set-up

In the following, we review the basic theory of hierarchical structuring of a DES G_{lo}^h for nonblocking hierarchical control [3], [4], i.e., low-level nonblocking control synthesis for G_{lo}^h to realize the high-level supervisor given a high-level specification for G_{hi}^{h+1}.

A given DES G with event set Σ is first structured into a Moore automaton[1] [26] - an automaton $G_{lo}^h = (Q_{lo}^h, \Sigma_{lo}^h, \delta_{lo}^h, q_{lo,0}^h, Q_{lo,m}^h)$ with $\Sigma_{lo}^h = \Sigma$, associated with an information channel defined by a vocalization map $V : Q_{lo}^h \to \Sigma_{hi}^{h+1} \cup \{\tau_o\}$, such that $G_{lo}^h \equiv G$. Σ_{hi}^{h+1} denotes the high-level (virtual) event set for a two-level hierarchy, and the symbol $\tau_o \notin \Sigma_{hi}^{h+1}$ denotes a 'silent output'. The Moore construction [26] for the DES G is based on a reporter map - a virtual projection $\theta : L(G) \to \Sigma_{hi}^{h+1,*}$, defined such that $\theta(\varepsilon) = \varepsilon$ and, for $\sigma \in \Sigma$ and $s\sigma \in L(G)$, $\theta(s\sigma)$ is either $\theta(s)$ or $\theta(s)\tau$ for some $\tau \in \Sigma_{hi}^{h+1}$. For the constructed G_{lo}^h, the vocalization map V for every $s' \in L(G_{lo}^h)$ is defined by

$$V(\delta_{lo}^h(s', q_{lo,0}^h)) = \begin{cases} \tau_o, & \text{if } s' = \varepsilon \\ & \text{or } \delta_{lo}^h(s', q_{lo,0}^h) \notin Q_{lo,voc}^h \\ \tau \in \Sigma_{hi}^{h+1}, & \text{otherwise,} \end{cases}$$

where the selected subset $Q_{lo,voc}^h \subseteq Q_{lo}^h$, called vocal state set, is defined as follows. For $\sigma \in \Sigma_{lo}^h$ and $s' = s\sigma$,

$$\delta_{lo}^h(s\sigma, q_{lo,0}^h) \begin{cases} \notin Q_{lo,voc}^h, & \text{if } \theta(s\sigma) = \theta(s) \\ \in Q_{lo,voc}^h, & \text{if } \theta(s\sigma) = \theta(s)\tau. \end{cases}$$

The reporter map θ can be extended to $\theta(K) \subseteq \Sigma_{hi}^{h+1,*}$ for $K \subseteq L(G_{lo}^h)$ as follows: $\theta(K) = \{\theta(s) \mid s \in K\}$. The inverse reporter map for $t \in \Sigma_{hi}^{h+1,*}$ is then defined as follows: $\theta^{-1}(t) = \{s \in L(G_{lo}^h) \mid \theta(s) = t\}$. The inverse reporter map θ^{-1} can be extended to $\theta^{-1}(E) \subseteq L(G_{lo}^h)$ for $E \subseteq \Sigma_{hi}^{h+1,*}$ as follows: $\theta^{-1}(E) = \bigcup_{t \in E} \theta^{-1}(t)$.

Through the map V, G_{lo}^h outputs events in Σ_{hi}^{h+1} to drive some high-level θ-image model G_{hi}^{h+1} whenever it reaches a vocal state $q \in Q_{lo,voc}^h$, and otherwise outputs the silent symbol $\tau_o \notin \Sigma_{hi}^{h+1}$ to signal no 'significant' change for the high level. For concreteness of reference, we write $G_{hi}^{h+1} = Higen(G_{lo}^h)$ to denote that G_{hi}^{h+1} is the high-level image of G_{lo}^h, such that $L(G_{hi}^{h+1}) = \{\theta(s^h) \mid s^h \in L(G_{lo}^h)\}$ and $L_m(G_{hi}^{h+1}) = \{\theta(s^h) \mid s^h \in L_m(G_{lo}^h)\}$. G_{hi}^{h+1} is said to generate events of Σ_{hi}^{h+1} under the θ-map on $L(G_{lo}^h)$.

In the standard control setting, the high-level event set Σ_{hi}^{h+1} is partitioned into the controllable event set $\Sigma_{hi,c}^{h+1}$ and the uncontrollable event set $\Sigma_{hi,u}^{h+1}$. The Moore automaton (G_{lo}^h, V) needs to be refined so that it becomes output-control consistent (OCC). A DES G_{lo}^h is said to be OCC [3] if, for every string $s \in L(G_{lo}^h)$ of the form

$$s = \sigma_1\sigma_2\cdots\sigma_k \text{ or, respectively, } s = s'\sigma_1\sigma_2\cdots\sigma_k$$

where $s' \in \Sigma_{lo}^{h,*} - \{\varepsilon\}$, with

- $V(\delta_{lo}^h(\sigma_1\sigma_2\cdots\sigma_i, q_{lo,0}^h) = \tau_o \quad (1 \le i \le k-1)$,
- $V(\delta_{lo}^h(s, q_{lo,0}^h)) = \tau \in \Sigma_{hi}^{h+1}$

or, respectively,

- $V(\delta_{lo}^h(s', q_{lo,0}^h)) \in \Sigma_{hi}^{h+1}$,
- $V(\delta_{lo}^h(s'\sigma_1\sigma_2\cdots\sigma_i, q_{lo,0}^h)) = \tau_o \quad (1 \le i \le k-1)$,
- $V(\delta_{lo}^h(s, q_{lo,0}^h)) = \tau \in \Sigma_{hi}^{h+1}$,

it is the case that

- if $\tau \in \Sigma_{hi,c}^{h+1}$, then for some i $(1 \le i \le k)$, $\sigma_i \in \Sigma_{lo,c}^h$,
- if $\tau \in \Sigma_{hi,u}^{h+1}$, then for all i $(1 \le i \le k)$, $\sigma_i \in \Sigma_{lo,u}^h$.

Being OCC intuitively means that every such high-level event $\tau \in \Sigma_{hi}^{h+1}$ defined and output by G_{lo}^h is unambiguously controllable or uncontrollable.

Given an OCC G_{lo}^h, further re-structuring of G_{lo}^h and modification of the associated θ-map above are needed, to address the problem of rendering the resulting pair (G_{lo}^h, G_{hi}^{h+1})

[1] In this paper, a Moore automaton [26] is often simply denoted by G_{lo}^h instead of (G_{lo}^h, V); the map V is then implied.

hierarchically consistent with marking (HCM[2]) [5]:

$$\begin{array}{ll} \text{HCM of} \iff & (\forall E)(E \subseteq \Sigma_{hi}^{h+1,*}) \\ (G_{lo}^h, G_{hi}^{h+1}) & \theta\left(L_m(Supcon(G_{lo}^h, \theta^{-1}(E)))\right) \\ & = \left(L_m(Supcon(G_{hi}^{h+1}, E))\right). \end{array} \quad (1)$$

Relating (1) to the set-up in Fig. 1, $S_{hi}^{h+1} \sqcap G_{hi}^{h+1} \equiv Supcon(G_{hi}^{h+1}, E)$ and $S_{lo}^h \sqcap G_{lo}^h \equiv Supcon(G_{lo}^h, \theta^{-1}(E))$. Under HCM, therefore, we can synthesize a low-level nonblocking S_{lo}^h for an OCC G_{lo}^h, to implement the high-level nonblocking S_{hi}^{h+1} for G_{hi}^{h+1} for an arbitrary E.

To achieve HCM, one computational approach applies the weakest known sufficiency conditions of HCM developed in [3] to restructure G_{lo}^h and modify the associated θ-map. The first known key computational procedures available to render G_{lo}^h OCC and the pair (G_{lo}^h, G_{hi}^{h+1}) HCM are reported in [20].

III. Multi-level Hierarchical Control

A. Multi-World Control Hierarchy

We can now describe a multi-world control hierarchy. Consider a modular DES

$$G = G_0 \parallel G_1 \parallel \cdots \parallel G_h \parallel \cdots \parallel G_N, \quad (2)$$

where each G_h, $h = 0, \cdots, N$, with real event set denoted by Σ_h, is trim and defines the real system component situated at level h of a system hierarchy. Such a modular DES is said to model a discrete-event organization (DEO). The component event sets are pair-wise disjoint, i.e, $(\forall i, j \in [0, N]; i \neq j)\Sigma_i \cap \Sigma_j = \emptyset$.

The basic building block of the multi-world control hierarchy is the structure depicted in Fig. 2(a), where Δ^h and Δ_{lo}^h denote the respective control data. This building block requires, at level h, the DES model G^h with event set denoted by Σ^h, the nonblocking supervisor S_h for a given local specification $K^h \subseteq \Sigma^{h,*}$, the resultant local control model $G_{lo}^h \equiv S_h \sqcap G_h$, and a further nonblocking supervisor S_{lo}^h to realize a given high-level specification $E_{hi}^{h+1} \subseteq \Sigma_{hi}^{h+1,*}$, for some DES model G_{hi}^{h+1} at level $(h+1)$ with virtual event set Σ_{hi}^{h+1}. The model $G_{hi}^{h+1} = Higen(G_{lo}^h)$ is obtained by some virtual projection θ on (the languages generated by) the level-h local control model G_{lo}^h. The trim model G_{h+1}^θ is an abstraction of the resulting nonblocking control model denoted by $S_{lo}^h \sqcap G_{lo}^h$, or equivalently, $\widehat{S}_h \sqcap G^h$, where

$$\widehat{S}_h \stackrel{\text{def}}{=} S_{lo}^h \sqcap S^h. \quad (3)$$

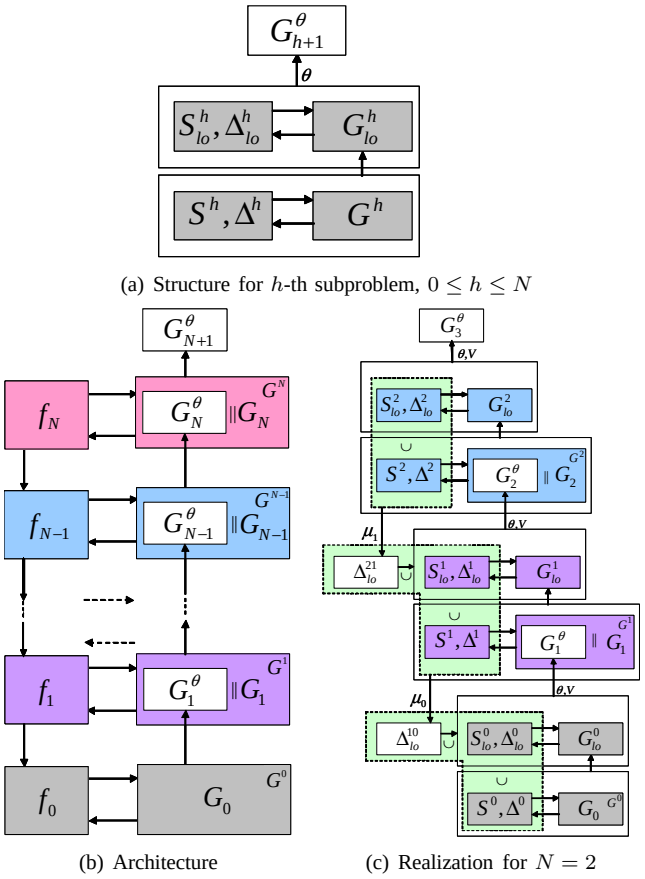
This abstraction is obtained under the virtual projection θ on the control model, such that

$$L_m(G_{h+1}^\theta) = \theta(L_m(\widehat{S}_h \sqcap G^h)). \quad (4)$$

For $h < N$, the real component G_{h+1} is composed with the virtual component G_{h+1}^θ to form the model G^{h+1}. The model G^h at every level, with event set Σ^h, is thus

$$G^h = \begin{cases} G_h, & \text{for } h = 0 \\ G_h^\theta \parallel G_h, & \text{for } h = 1, \cdots, N \\ G_h^\theta, & \text{for } h = N+1. \end{cases} \quad (5)$$

[2]Depending on context, HCM stands for either 'hierarchically consistent with marking' or 'hierarchical consistency with marking'.



(a) Structure for h-th subproblem, $0 \leq h \leq N$

(b) Architecture

(c) Realization for $N = 2$

Fig. 2. Organizational control for a multi-world control hierarchy

The virtual event set of G_{h+1}^θ is denoted by Σ_{h+1}^θ. $\Sigma^{h+1} = \Sigma_{h+1}^\theta \cup \Sigma_{h+1}$, and $\Sigma_{N+1} = \emptyset$. Referring to Fig. 1, one can also think of G_{h+1}^θ as the result of some nonblocking virtual supervision S_{hi}^{h+1} on the model G_{hi}^{h+1} with virtual event set $\Sigma_{hi}^{h+1} \supseteq \Sigma_{h+1}^\theta$, to meet the given high-level specification $E_{hi}^{h+1} \subseteq \Sigma_{hi}^{h+1,*}$. In meeting (4), the virtual supervision on G_{hi}^{h+1} at level $(h+1)$ is effectively implemented by $\widehat{S}_h$ (3) on G^h at the lower level. $\Sigma_{hi}^{h+1} \cap \Sigma_{h+1} = \emptyset$, and it follows that $\Sigma_{h+1}^\theta \cap \Sigma_{h+1} = \emptyset$.

In modeling G^h as a different world, the event sets of G^i and G^j are pair-wise disjoint, i.e., $(\forall i, j \in [0, N+1]; i \neq j)\Sigma^i \cap \Sigma^j = \emptyset$. $(G^0, G^1, \cdots, G^{N+1})$ constitutes a multi-world control hierarchy.

B. Organizational Control: Solution Architecture

To realize the multi-world control hierarchy, we propose an organizational control architecture as depicted in Fig. 2(b). Therein, the control law $f_h : L(G^h) \to 2^{\Sigma^h}$, $h = 0, \cdots, N$, with $f_{N+1} = \emptyset$, is said to be optimal w.r.t each pair of high-level and local specifications (E_{hi}^{h+1}, K^h), if all the following conditions hold.

1) $G_{lo}^h = Supcon(G^h, K^h)$.
2) $G_{h+1}^\theta = Supcon(G_{hi}^{h+1}, E_{hi}^{h+1})$, where $G_{hi}^{h+1} = Higen(G_{lo}^h)$.
3) $\sigma \in f_h(G^h, s^h) - Condat(G^h \sqcap \widehat{S}_h, s^h)$ implies either
   - $(\exists t \in \Sigma_{lo,u}^{h,*})(\exists \sigma' \in \Sigma_{lo,u}^h)\left(s^h \sigma t \sigma' \in L(G_{lo}^h)\right.$ and $\left.\theta(s^h \sigma t \sigma') = \theta(s^h t)\tau = \theta(s^h)\tau\right)$ or

- $s^h\sigma \in L(G^h_{lo})$ and $\theta(s^h\sigma) = \theta(s^h)\tau$,

and $\tau \in f_{h+1} \cap \Sigma^\theta_{h+1}$.

We may now state the problem.

Problem 1: Given a multi-world control hierarchy $(G^0, G^1, \cdots, G^{N+1})$ as described, construct an optimal nonblocking multi-level control law f_h over $h = 0, \cdots, N$, such that $\theta(L_m(f_h, G^h)) \subseteq L_m(G^\theta_{h+1})$.

IV. Control Synthesis and Implementation

A. Hierarchical Control Synthesis

In addressing the organizational control Problem 1, we first consider building the model G^θ_{h+1} for a different world G^{h+1}, $h \geq 0$, as the optimization subproblem of two-level hierarchical control unified with local level supervision.

The subproblem is defined as follows. Subject to a specification pair (E^{h+1}_{hi}, K^h), $h = 0, \cdots, N$, for a DES G^h (5), let $E_h = L_m(G^h_{lo}) \cap \theta^{-1}(E^{h+1}_{hi})$, where $G^h_{lo} = Supcon(G^h, K^h) \equiv S^h \sqcap G^h$ for some state-reduced supervisor S^h; and $G^{h+1}_{hi} = Higen(G^h_{lo})$. Then the subproblem is to find $\widehat{S}_h$ (3) for G^h, $Supcon(G^h, E_h) \equiv \widehat{S}_h \sqcap G^h$, such that

$$L_m(G^\theta_{h+1}) = \theta(L_m(\widehat{S}_h \sqcap G^h)),$$

where $G^\theta_{h+1} = Supcon(G^{h+1}_{hi}, E^{h+1}_{hi})$. In essence, the subproblem is to meet (4) that relates consecutive worlds (G^h, G^{h+1}).

Now, we can construct a state-reduced supervisor S^h_{lo} such that $Supcon(G^h_{lo}, \theta^{-1}(E^{h+1}_{hi})) \equiv S^h_{lo} \sqcap G^h_{lo}$. In what follows, Theorem 1 states the condition for solving the subproblem.

Theorem 1: By the foregoing definitions and synthesis, if the pair (G^h_{lo}, G^{h+1}_{hi}) is HCM (1), $L_m(G^\theta_{h+1}) = \theta(L_m(\widehat{S}_h \sqcap G^h))$.

Proof: By the HCM of (G^h_{lo}, G^{h+1}_{hi}), $L_m(G^\theta_{h+1}) = \theta(L_m(S^h_{lo} \sqcap G^h_{lo}))$. Since $\widehat{S}_h \sqcap G^h \equiv S^h_{lo} \sqcap G^h_{lo}$, it follows that $L_m(G^\theta_{h+1}) = \theta(L_m(\widehat{S}_h \sqcap G^h))$. Hence the theorem. ■

Under HCM of (G^h_{lo}, G^{h+1}_{hi}), an approach to building a multi-world control hierarchy is to first solve, successively bottom-up from level $h = 0$, the overlapping hierarchical optimization subproblems of the same kind described above and depicted in Fig. 2(a). The 'overlap' in the structures between consecutive levels is the aggregated virtual model G^θ_{h+1} in G^h (5).

Once HCM is achieved for the pair (G^h_{lo}, G^{h+1}_{hi}), the same constructions can be applied again by assigning state outputs in G^{h+1}_{lo} - the component obtained under local level-$(h+1)$ control S^{h+1} of G^{h+1} for K^{h+1}, and bringing in the next level G^{h+2}_{hi}. Clearly, HCM can be achieved for the pair $(G^{h+1}_{lo}, G^{h+2}_{hi})$, constructed without disturbing the HCM of (G^h_{lo}, G^{h+1}_{hi}), and so on. Referring to Fig. 2(a), at each level h, the component supervisors of S^h_{lo} and S^h for the world G^h can be constructed given the specification pair (E^{h+1}_{hi}, K^h).

In what follows, a control law for the constructed multi-world hierarchy $(G^0, G^1, \cdots, G^{N+1})$ of a DEO G (2) is proposed. Over past event dynamics or event history $s^h \in L(G^h)$, $h = 0, \cdots, N$, it is a dynamic programming recursion

$$\begin{aligned} f_h(G^h, s^h) = {} & Condat(G^h \sqcap S^h, s^h) \\ & \cup\ Condat(G^h_{lo} \sqcap S^h_{lo}, s^h) \\ & \cup\ \mu_h(f_{h+1}(G^{h+1}, s^{h+1}) \cap \Sigma^\theta_{h+1}, G^h_{lo}, s^h), \\ & \qquad h = 0, \cdots, N-1, \\ f_N(G^N, s^N) = {} & Condat(G^N \sqcap S^N, s^N) \\ & \cup\ Condat(G^N_{lo} \sqcap S^N_{lo}, s^N), \end{aligned} \tag{6}$$

$$\text{where } \mu_h(.) = \Big\{\sigma \in \Sigma^h_{lo,c} \mid (\exists t \in \Sigma^{h,*}_{lo,u}) V(\delta^h_{lo}(s^h\sigma t, q^h_{lo,0})) \in f_{h+1}(G^{h+1}, s^{h+1}) \cap \Sigma^\theta_{h+1}\Big\}.$$

The set $\Sigma^{comdat}_{h+1} \stackrel{\text{def}}{=} (f_{h+1} \cap \Sigma^\theta_{h+1})$ in μ_h of f_h (6) is said to constitute the command decisions at level $(h+1)$ for level h, due to the constraints imposed at higher levels ($> h$).[3] These commands are control decisions on G^θ_{h+1} of G^{h+1}.

To depict the overall approach, Fig. 2(c) shows a realization of the control hierarchy depicted in Fig. 2(b), based on the subproblem structure depicted in Fig. 2(a). For $N = 2$, the figure shows the consecutively overlapping control structures of three optimization subproblems of the same kind, addressed to build an organizational control architecture for a DEO $G = G_0 \parallel G_1 \parallel G_2$. Under control law f_h, μ_h determines $\Delta^{(h+1)h}_{lo}$, the set of control decisions at level h to carry out the commands in Σ^{comdat}_{h+1}.

B. Optimal and Nonblocking Control Law

We now introduce a mild assumption that asserts some fairness of command at level $(h+1)$, in that the transitions of every controllable (virtual or command) event of $\Sigma^\theta_{h+1} \subseteq \Sigma^{h+1}$ are never permanently disabled in G^θ_{h+1}:

Assumption 1: For $h = 0, \cdots, N-1$,

$$P_{\Sigma^{h+1}, \Sigma^\theta_{h+1}}\left(L\left(\widehat{S}_{h+1} \sqcap G^{h+1}\right)\right) = L\left(G^\theta_{h+1}\right).$$

Under this assumption, $\widehat{S}_{h+1}$ is said to 'preserve' the aggregated virtual model G^θ_{h+1}.

Theorem 2: Given that, for $h = 0, \cdots, N$, G^h_{lo} is OCC and (G^h_{lo}, G^{h+1}_{hi}) is HCM. Then, under Assumption 1, f_h (6) is a solution control law for Problem 1.

Proof: We need to establish the optimality and nonblockingness of f_h (6), and show that $\theta(L_m(f_h, G^h)) \subseteq L_m(G^\theta_{h+1})$.

1) *Proof of optimality*: The three conditions for optimality w.r.t each specification pair (E^{h+1}_{hi}, K^h), as stated in Section III-B, are satisfied as follows:
By construction, $G^h_{lo} = Supcon(G^h, K^h)$ and $G^\theta_{h+1} = Supcon(G^{h+1}_{hi}, E^{h+1}_{hi})$, where $G^{h+1}_{hi} = Higen(G^h_{lo})$.
Since G^h_{lo} is OCC, μ_h as in f_h (6) is such that:
$\sigma \in \mu_h(f_{h+1} \cap \Sigma^\theta_{h+1}, G^h_{lo}, s^h)$ iff either
   - $(\exists t \in \Sigma^{h,*}_{lo,u})(\exists \sigma' \in \Sigma^h_{lo,u}) \left(s^h\sigma t\sigma' \in L(G^h_{lo}) \text{ and } \theta(s^h\sigma t\sigma') = \theta(s^h\sigma t)\tau = \theta(s^h)\tau\right)$ or

[3] It should be clear that Σ^{comdat}_{h+1} is not a constant set and depends in general on the event history $s^{h+1} \in L(G^{h+1})$.

- $s^h\sigma \in L(G^h_{lo})$ and $\theta(s^h\sigma) = \theta(s^h)\tau$,

and $\tau \in f_{h+1} \cap \Sigma^\theta_{h+1}$.
Since $G^h \sqcap \widehat{S}_h \equiv (G^h \sqcap S^h) \sqcap (G^h_{lo} \sqcap S^h_{lo})$,

$$Condat(G^h \sqcap \widehat{S}_h, s^h) = Condat(G^h \sqcap S^h, s^h) \cup Condat(G^h_{lo} \sqcap S^h_{lo}, s^h).$$

Therefore, together with (6) we have

$$\Big(f_h(G^h, s^h) - Condat(G^h \sqcap \widehat{S}_h, s^h)\Big) \subseteq \mu_h(-, G^h_{lo}, s^h).$$

The last stated condition is thus satisfied. Hence f_h is optimal.

2) *Proof of nonblockingness*: Stating (4) which holds under HCM of (G^h_{lo}, G^{h+1}_{hi}) by Theorem 1:

$$\theta(L_m(\widehat{S}_h \sqcap G^h)) = L_m(G^\theta_{h+1}). \tag{7}$$

Implicitly, $\theta(\overline{L(\widehat{S}_h \sqcap G^h)}) = L(G^\theta_{h+1})$ since, by construction, $\overline{L_m(\widehat{S}_h \sqcap G^h)} = L(\widehat{S}_h \sqcap G^h)$ and $\overline{L_m(G^\theta_{h+1})} = L(G^\theta_{h+1})$. Then, under Assumption 1 and OCC G^h_{lo}, it can be shown that $L(f_h, G^h) = L(\widehat{S}_h \sqcap G^h)$, and therefore

$$L_m(f_h, G^h) = L_m(\widehat{S}_h \sqcap G^h). \tag{8}$$

It follows easily that $\overline{L_m(f_h, G^h)} = L(f_h, G^h)$. Hence f_h is nonblocking.

3) *Proof of* $\theta(L_m(f_h, G^h)) \subseteq L_m(G^\theta_{h+1})$: From (7) and (8), it follows that

$$\theta(L_m(f_h, G^h)) = L_m(G^\theta_{h+1}) \subseteq L_m(G^\theta_{h+1}).$$

Hence the theorem. ∎

In building a multi-world control hierarchy for Problem 1, we can first apply the HCM synthesis method [3], [20] to render every G^h_{lo} OCC and every pair (G^h_{lo}, G^{h+1}_{hi}) HCM.

C. Organizational Control Algorithm

The organizational control architecture can be realized as a control algorithm with a command and control recursion coupling top-down control law f_h (6) to bottom-up information feedback (by event vocalization V).

Function $com(\Sigma^{comdat}_{h+1}, G^h_{lo}, q^h_{lo})$

Input: Σ^{comdat}_{h+1}, G^h_{lo}, and q^h_{lo}. **Output:** $\Delta^{(h+1)h}_{lo}$.
begin
  if $\Sigma^{comdat}_{h+1} = \emptyset$ **then**
    $\Delta^{(h+1)h}_{lo} := \emptyset$;
  else
    $Q^Z_m := \bigcup_{\tau \in \Sigma^{comdat}_{h+1}} Q^h_{lo,voc}[\tau]$; $z_0 := q^h_{lo}$;
    $Z := (Q^h_{lo}, \Sigma^h_{lo}, \delta^h_{lo}, z_0, Q^Z_m)$; $Z := RTrim(Z)$;
    if $Q^Z = \emptyset$ **then** $\Delta^{(h+1)h}_{lo} := \emptyset$;
    else $\Delta^{(h+1)h}_{lo} := \Sigma^Z(z_0)$;
  return $\Delta^{(h+1)h}_{lo}$;
end

Fig. 3. A basic on-line computational function com

Key to the recursive control algorithm is a command-to-control function com that computes μ_h on-line, and returns

Procedure `HI-manager` **(Level** $h = N$**)**

begin
initialize
  Initialize states in S^N, S^N_{lo}, G^N_{lo} and G^N;
  Use initial states to apply off-line condat $[\Delta^N \cup \Delta^N_{lo}]$ to G^N;
  Compute $\Sigma^{comdat}_N := [\Delta^N \cup \Delta^N_{lo}] \cap \Sigma^\theta_N$ and send it down to level $(N-1)$;
while *true* **do**
  Wait to receive level-N event occurrence (from Σ^N) and update states in S^N, S^N_{lo}, G^N_{lo} and G^N;
  Use current states to update off-line condat $[\Delta^N \cup \Delta^N_{lo}]$, and apply it to G^N;
  Compute $\Sigma^{comdat}_N := [\Delta^N \cup \Delta^N_{lo}] \cap \Sigma^\theta_N$ and send it down to level $(N-1)$;
end

Fig. 4. Organizational control: Level N

Procedure `MID-manager` **(Level** $h = 1, \cdots, N-1$**)**

begin
initialize
  Initialize states in S^h, S^h_{lo}, G^h_{lo} and G^h;
  Use initial states to apply off-line condat $[\Delta^h \cup \Delta^h_{lo}]$ to G^h;
  Wait to receive updated Σ^{comdat}_{h+1} from level $(h+1)$;
  Use current Σ^{comdat}_{h+1} and q^h_{lo} to compute condat $\Delta^{(h+1)h}_{lo} := com$, and apply it to G^h;
  Compute $\Sigma^{comdat}_h := [\Delta^h \cup \Delta^h_{lo} \cup \Delta^{(h+1)h}_{lo}] \cap \Sigma^\theta_h$ and send it down to level $(h-1)$;
while *true* **do**
  Wait to receive level-h event occurrence (from Σ^h) **or** updated Σ^{comdat}_{h+1} from level $(h+1)$;
  if *level-h event is received* **then**
    Update states in S^h, S^h_{lo}, G^h_{lo} and G^h;
    Use current states to update off-line condat $[\Delta^h \cup \Delta^h_{lo}]$, and apply it to G^h;
    if *current state q^h_{lo} is vocal* **then**
      Send vocalized event $V(q^h_{lo})$ up to level $(h+1)$;
      Wait to receive updated Σ^{comdat}_{h+1} from level $(h+1)$;
  Use current Σ^{comdat}_{h+1} and q^h_{lo} to compute condat $\Delta^{(h+1)h}_{lo} := com$, and apply it to G^h;
  Compute $\Sigma^{comdat}_h := [\Delta^h \cup \Delta^h_{lo} \cup \Delta^{(h+1)h}_{lo}] \cap \Sigma^\theta_h$ and send it down to level $(h-1)$;
end

Fig. 5. Organizational control: Mid-level

that as part of the control decisions on G^h. Fig. 3 presents a basic function for com, where the input event history s^h is replaced by the corresponding state $q^h_{lo} = \delta^h_{lo}(s^h, q^h_{lo,0})$ of G^h_{lo}; and $RTrim(Z)$ returns $Trim(Z \sqcap R)$, a trim automaton of $Z \sqcap R$ for some $Z = (Q^Z, \Sigma^Z, \delta^Z, z_0, Q^Z_m)$ that is G^h_{lo} but with initial state $z_0 := q^h_{lo}$ and marked state set

$$Q^Z_m := \bigcup_{\tau \in \Sigma^{comdat}_{h+1}} Q^h_{lo,voc}[\tau],$$

where $Q^h_{lo,voc}[\tau] = \{q \in Q^h_{lo,voc} \mid V(q) = \tau \in \Sigma^{h+1}_{hi}\}$; and R is a two-state trim automaton with $L_m(R) = \Sigma^h_{lo,c}\Sigma^{h,*}_{lo,u}$ if $\Sigma^h_{lo,c} \neq \emptyset$, and is otherwise an empty automaton.

Specifically, the control algorithm contains three procedures `HI-manager`, `MID-manager` and `LO-manager` that implement the hierarchical control at levels $h = N \geq 1$, $h = 1, \cdots, N-1$ and $h = 0$, as shown respectively in Figs. 4 to 6, wherein, following an evolving history s^h, the current state $q^h_{lo} = \delta^h_{lo}(s^h, q^h_{lo,0})$, $\Delta^h = Condat(G^h \sqcap S^h, s^h)$ and

Procedure `LO-manager` **(Level** $h = 0$**)**

begin
initialize
Initialize states in S^0, S^0_{lo}, G^0_{lo} and G^0;
Use initial states to apply off-line condat $[\Delta^0 \cup \Delta^0_{lo}]$ to G^0;
Wait to receive updated Σ_1^{comdat} from level 1;
Use current Σ_1^{comdat} and q^0_{lo} to compute condat $\Delta^{10}_{lo} := com$, and apply it to G^0;
while *true* **do**
Wait to receive level-0 event occurrence (from Σ^0) **or** updated Σ_1^{comdat} from level 1;
if *level-*0 *event is received* **then**
Update states in S^0, S^0_{lo}, G^0_{lo} and G^0;
Use current states to update off-line condat $[\Delta^0 \cup \Delta^0_{lo}]$, and apply it to G^0;
if *current state* q^0_{lo} *is vocal* **then**
Send vocalized event $V(q^0_{lo})$ up to level 1;
Wait to receive updated Σ_1^{comdat} from level 1;
Use current Σ_1^{comdat} and q^0_{lo} to compute condat $\Delta^{10}_{lo} := com$, and apply it to G^0;
end

Fig. 6. Organizational control: Base level

$\Delta^h_{lo} = Condat(G^h_{lo} \sqcap S^h_{lo}, s^h)$.

Note that, where it is necessary to have a procedure to track the states in G^θ_{N+1} during operation, procedure `HI-manager` can be easily modified to send vocalized events to the procedure that updates the current state in G^θ_{N+1}.

V. Discussion

A. Incremental Synthesis

The approach to organizational control as described in Section IV-A can be said to apply incremental synthesis of HCM to build a multi-world control hierarchy using virtual projection. Incremental synthesis in general entails successive construction of a hierarchy by synchronizing some remaining system components with abstracted controlled subsystems for control synthesis at consecutively higher levels, until no component is left at the highest level of the hierarchy. This helps to mitigate synthesis complexity by avoiding computation over the synchronous product of all system components.

The conceptual idea of incremental synthesis is found to originate with the work of [27] in DES control research. Therein, however, it is formalized using natural projection as abstraction, and applied to synthesize modular controllers entirely off-line, whereas it is formalized in our work using virtual projection as abstraction, and applied to build a control organization that performs control computations partially on-line based on hierarchical system evolution.

B. Design Reusability

The organizational control architecture as proposed and depicted in Fig. 2 offers reusability in the sense that a level-$k \geq 1$ component other than G^θ_k can be modified without having to redesign the lower levels $0, \cdots, (k-1)$. It is also possible that the higher levels $(i+1), \cdots, N$ for some smallest $i \geq k$ need not be redesigned. However, this is provided the modification and redesign made at level k does not change the virtual component G^θ_{i+1}, which may not always happen.

C. Computational Complexity

1) Off-line versus On-line & Response Time: It is known [23] that $Meet$ (implementing operator $\sqcap$) has (an order of) complexity in the product of the state sizes of the two input automata, and $Trim$ has linear complexity in the state size of the input automaton. Following which, it can be easily shown that the on-line computational complexity for com at level h (see Fig. 3) is linear in the state size of G^h_{lo}.

Now, given f_h (6), it can be shown that for some $S^{(h+1)h}_{lo}$,

$$L\left(S^{(h+1)h}_{lo} \sqcap \left((\widehat{S}_h \sqcap G^h) \parallel G_{h+1} \cdots \parallel G_N\right)\right)$$
$$= L(f_h, G^h) \parallel L(f_{h+1} \cap \Sigma_{h+1}, G^h \parallel G_{h+1})$$
$$\cdots \parallel L(f_N \cap \Sigma_N, G^h \parallel G_{h+1} \cdots \parallel G_N).$$

Clearly, the on-line function com for μ_h of f_h (6) in the organizational control architecture does away altogether the computationally intensive synthesis of $S^{(h+1)h}_{lo}$ that would otherwise be required as in existing work [3], [20]. This evidently reduces the off-line time complexity and memory requirements in organizational control built according to Theorem 2, but comes at the expense of increasing the on-line time complexity. Though linear, the total time incurred due primarily to on-line computation by com at every responding control level $h = 0, \cdots, N-1$ needs to be accommodated by the required response time, if the proposed hierarchical control approach is to be preferred over that of having to construct $S_h \stackrel{\text{def}}{=} S^{(h+1)h}_{lo} \parallel \widehat{S}_h$, where S_0 is effectively the control solution for DES G (2). Investigating this real-time feasibility issue is beyond the logical DES scope of this paper.

2) Memoization Strategy in Dynamic Programming: The functional solution equation (6) furnishes the basis of a caching or memoization strategy in the dynamic programming control implementation (Figs. 4 to 6). By memoization, the valid on-line decision cache previously computed is reused when computing new control decisions. So by this strategy, the computation of the next control decisions following an event $\sigma \in \Sigma$ occurrence can continually reuse the on-line command cache Σ^{comdat}_{h+1} previously computed, as long as the cache remains valid, in that the current state q^h_{lo} (of G^h_{lo}) entered is due to $\sigma \in \Sigma^h$ and is not vocal. It can also reuse the off-line control cache $[\Delta^h_{lo} \cup \Delta^h]$ without updating it, as long as the event $\sigma \in \Sigma$ that occurs is from the higher levels, $\bigcup_{i=h+1}^N \Sigma_i$. This means that in response to an event occurrence, the caching strategy can always keep the control data of the solution control S_0 validated, without having to recompute or redundantly update any valid command or control decisions.

D. On-line Transparency of Control Operations

The individual off-line synthesis of supervisors S^h and S^h_{lo} that constitute the subproblem solution $\widehat{S}_h$ leads to on-line (or runtime) transparency of control operations for the respective specifications K^h and E^{h+1}, namely, we can know what stepwise control decisions are needed to enforce the respective specifications. By being able to successively explicate the level-h control decisions made specifically to

carry out level-$(h+1)$ commands through com, the on-line transparency is increased to include control operations between consecutive levels, namely, we can also know what stepwise control decisions are needed to carry out a command decision from immediately above. This is a beneficial byproduct of organizational control.

VI. Illustrative Example

We consider a robot transfer manipulator moving workpieces from an input buffer to an output buffer in an obstacle-constrained manufacturing workspace. This simple but interesting example is adapted and modified from [19].

A. Problem Description

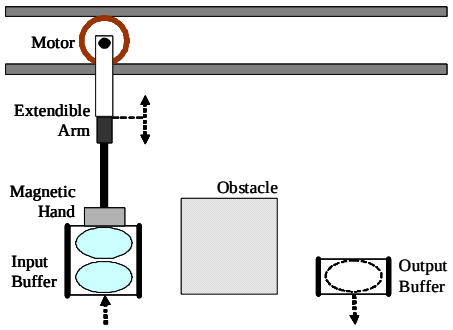


Fig. 7. Physical layout of a robot transfer manipulator and its workspace

As shown in the physical layout in Fig. 7, the robot manipulator consists of an extendible arm and a magnetic hand, and is motor driven to traverse horizontally along a track between two buffers. The capacities of the input and output buffers are two and one, and are initially full and empty, respectively. Initially, the arm rests extended, with the hand left open (by electrically turning off the magnet) and touching the foremost workpiece in the input buffer. Initially on the left, the motor can drive the manipulator back-and-forth, between the left and the right. The workpieces enter the system when they are put and held in the input buffer, and exit the system when they are taken out of the output buffer that they were transferred into. The arm can contract, or extend so that the hand attached to its end can reach a buffer to pick or to drop a workpiece. The hand picks and drops a workpiece by closing and opening, respectively.

By designer intuition, the DES can be modeled and hierarchically modularized as $G = G_0 \parallel G_1 \parallel G_2$, where $G_0 = \text{ARM} \parallel \text{MOTOR}$, $G_1 = \text{HAND}$ and $G_2 = \text{InBUF} \parallel \text{OutBUF}$ (look for these component automata in Fig. 8, which depicts the overall system and multi-level control).

In Fig. 8, an automaton is represented by an edge-labelled directed graph with a state represented by a node, and a transition $\delta(\sigma, q) = q'$ by a directed edge from state q to q' labelled with the symbol σ of an event whose occurrence it represents. A controllable event may be indicated by an optional tick on its edge ($\circ\!\!-\!\!\!\!\rightarrow\!\!\circ$). The initial state is represented by a node with an entering arrow, a marked state by a darkened node, and a vocal state by a node containing the symbol of an event that it vocalizes.

B. Bottom-up Design Modeling: Specifications & Abstractions

With $N = 2$, the organizational control structure has four worlds at levels $0, \cdots, 3$ (see Fig. 8), and can be built by successively solving each overlapping subproblem (with control structure as depicted in Fig. 2(a)) bottom-up, as follows.

- At level 0 is the robot joint space (G^0) to be controlled so that the obstacle can always be avoided. Given local level specification K^0 for obstacle avoidance, $G^0_{lo} \equiv Supcon(G^0, K^0)$, which is reconstructed with $G^1_{hi} = Higen(G^0_{lo})$ such that the pair (G^0_{lo}, G^1_{hi}) is HCM. Not shown, G^1_{hi} is an automaton that contains the same transition structure as G^0_{lo}, except that every transition into a state is labelled with an event vocalized by the latter's corresponding state.
- Next, we need level-1 to model the robot task space, which is an abstracted model of transporting workpieces (G^θ_1) without oscillation between adjacent states (resulting from some high-level specification E^1_{hi}), interleaving with the hand operations (G_1). $G^\theta_1 = Supcon(G^1_{hi}, E^1_{hi})$. Ensuring no oscillation entails underlying supervision S^0_{lo} since $L_m(G^\theta_1) \subset L_m(G^1_{hi})$.

  Following, the resultant task space model $G^1 = G^\theta_1 \parallel G_1$ is to be controlled to ensure that a proper workcycle is continually carried out. Given specification K^1 for workcycle, $G^1_{lo} \equiv Supcon(G^1, K^1)$, which is reconstructed with $G^2_{hi} = Higen(G^1_{lo})$ such that the pair (G^1_{lo}, G^2_{hi}) is HCM.
- Further up, level-2 needs to model the robot workspace, which is an abstracted pick-and-drop manipulator model (G^θ_2) resulting from $E^2_{hi} = L_m(G^2_{hi})$, interleaving with the buffer operations (G_2). Since $G^\theta_2 \equiv G^2_{hi}$, no underlying supervision is required.

  Following, the resultant workspace model $G^2 = G^\theta_2 \parallel G_2$ is to be controlled to ensure no overflow or underflow of the input and output buffers due to their capacity limits. Given specification K^2 for buffer limits, $G^2_{lo} = Supcon(G^2, K^2)$.
- Finally, level-3 needs to model the storage usage with the workpieces entering or exiting the system, and with the robot manipulator never holding on to a workpiece indefinitely (G^θ_3). It turns out that G^θ_3 and an equivalent model of G^3_{hi} can be obtained by projecting out all events except in and out in G^2_{lo}, and replacing them by $enter$ and $exit$, respectively. Clearly no underlying supervision is required since $G^\theta_3 \equiv G^3_{hi}$. Assuming that no level-3 state tracking of G^θ_3 is required, the design reconstruction of G^2_{lo} with $G^3_{hi} = Higen(G^2_{lo})$ to render the pair (G^2_{lo}, G^3_{hi}) HCM is not necessary.

C. Control Synthesis & Implementation

Since Q^h_{lo} can be used as the composite state set of $G^h \sqcap S^h$, the off-line local control data at each level $h = 2, 1, 0$ is computed and listed as $\Delta^h(q)$ in Tables I(a), I(b), and I(d), respectively, where $q \in Q^h_{lo}$. Denote Q^θ_1 as the state set of G^θ_1. Then a subset of the composite state set $Q^\theta_1 \times Q^0_{lo}$ can be used as the state set of S^0_{lo} such that for every reachable state (q, x) in $G^0_{lo} \sqcap S^0_{lo}$, $q \in Q^0_{lo}$ and $x = (q^\theta, q) \in Q^\theta_1 \times Q^0_{lo}$. Abbreviating

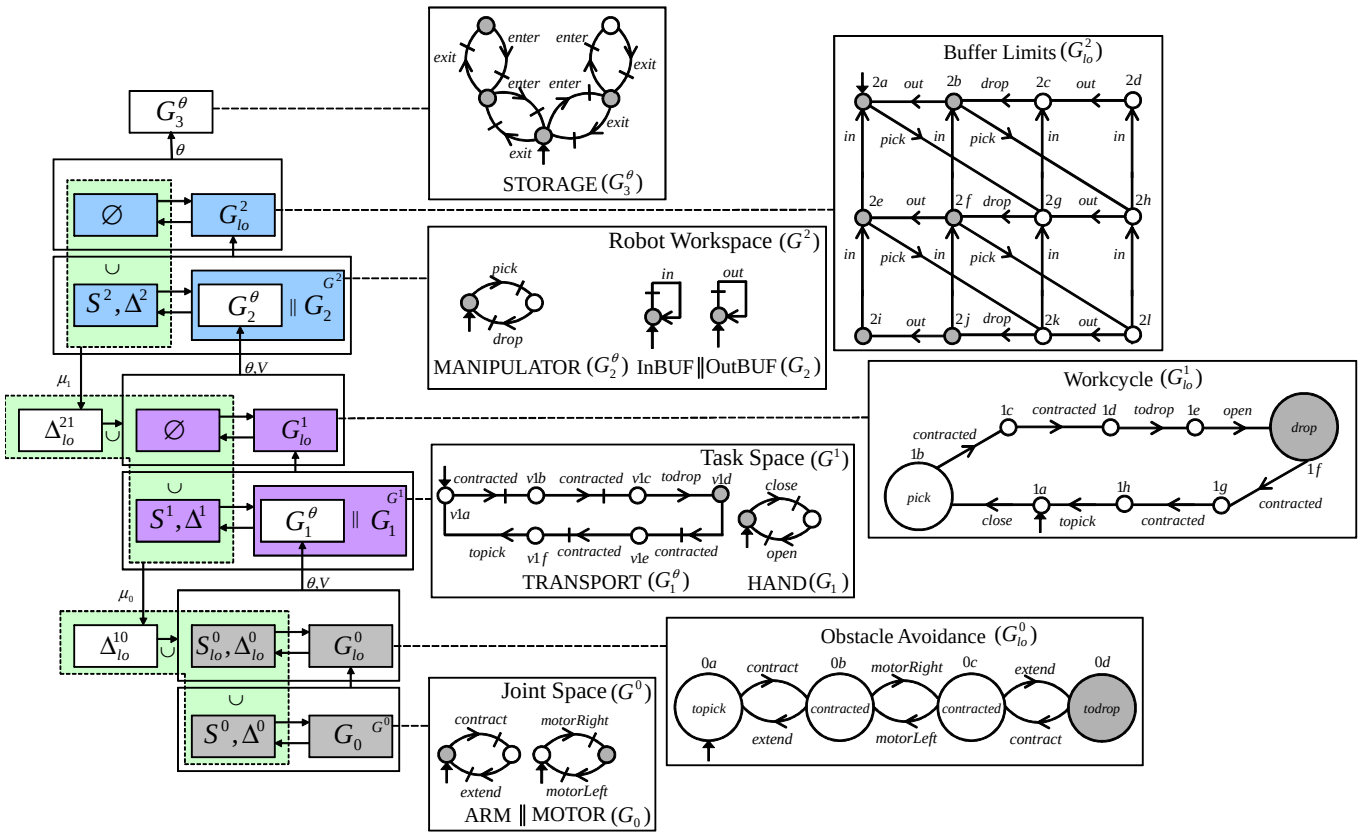


Fig. 8. Organizational control architecture: Bottom-up HCM systems modeling for multi-level control of a robot transfer manipulator

TABLE I

CONTROL DATA

(a) Local control data at $q \in Q_{lo}^2$ (Level 2)

$\triangle^2(2a)$	$=\{in, out\}$	$\triangle^2(2b)$	$=\{in\}$	$\triangle^2(2c)$	$=\{in, out\}$	$\triangle^2(2d)$	$=\{drop, in\}$
$\triangle^2(2e)$	$=\{out\}$	$\triangle^2(2f)$	$=\{pick\}$	$\triangle^2(2g)$	$=\{out\}$	$\triangle^2(2h)$	$=\{drop\}$
$\triangle^2(2i)$	$=\{pick, out\}$	$\triangle^2(2j)$	$=\{pick\}$	$\triangle^2(2k)$	$=\{out\}$	$\triangle^2(2l)$	$=\{drop\}$

(b) Local control data at $q \in Q_{lo}^1$ (Level 1)

$\triangle^1(1a)$	$=\{contracted\}$	$\triangle^1(1b)$	$=\{open\}$	$\triangle^1(1c)$	$=\{open\}$	$\triangle^1(1d)$	$=\{open\}$
$\triangle^1(1e)$	$=\{contracted\}$	$\triangle^1(1f)$	$=\{close\}$	$\triangle^1(1g)$	$=\{close\}$	$\triangle^1(1h)$	$=\{close\}$

(c) High-level control data at $x \in X^{(S_{lo}^0)} \subseteq Q_1^\theta \times Q_{lo}^0$ (Level 0)

$\triangle_{lo}^0(v1a, 0a)$	$=\emptyset$	$\triangle_{lo}^0(v1b, 0b)$	$=\{extend\}$
$\triangle_{lo}^0(v1c, 0c)$	$=\{motorLeft\}$	$\triangle_{lo}^0(v1d, 0d)$	$=\emptyset$
$\triangle_{lo}^0(v1e, 0c)$	$=\{extend\}$	$\triangle_{lo}^0(v1f, 0b)$	$=\{motorRight\}$

(d) Local control data at $q \in Q_{lo}^0$ (Level 0)

$\triangle^0(0a)$	$=\{motorRight\}$	$\triangle^0(0b)$	$=\emptyset$
$\triangle^0(0c)$	$=\emptyset$	$\triangle^0(0d)$	$=\{motorLeft\}$

$\Delta_{lo}^0(q, x)$ as $\Delta_{lo}^0(x)$, the off-line (high-level) control data due to $G_{lo}^0 \sqcap S_{lo}^0$ is computed and listed in Table I(c).

With the essential components of the architecture in place, the control implementation of the `HI-manager`, `MID-manager` and `LO-manager` is immediate (respectively from Figs. 4 to 6). It is easy to deduce that Assumption 1 holds. Hence the resultant control implementation is optimal and nonblocking.

To illustrate how the implementation mechanism works, we refer to Fig. 8 and Tables I and II. Suppose the

system is in state $(2h, 1d, vlc, 0c)$, at which $\Delta^2(2h) = \{drop\}$, virtually disabling event $drop$ at level 2; $\Delta^1(1d) = \{open\}$, disabling event $open$ at level 1; and $\Delta^0_{lo}(vlc, 0c) \cup \Delta^0(0c) = \{motorLeft\}$, disabling event $motorLeft$ at level 0. When, say, the enabled $extend$ at level 0 occurs, the state $[vlc, 0c]$ is updated to $[vld, 0d]$, and $\Delta^0_{lo}(vld, 0d) \cup \Delta^0(0d) = \{motorLeft\}$, disabling $motorLeft$ with state $0d$ vocalizing $todrop$ which is sent up to level 1. Upon receiving $todrop$, state $1d$ is updated to $1e$, following which $\Delta^1(1e) = \{contracted\}$, virtually disabling event $contracted$ at level 1. With $\Sigma_2^{comdat} := \{drop\}$ since $\Delta^2(2h) = \{drop\}$, condat $\Delta^{21}_{lo} := com(\{drop\}, G^1_{lo}, 1e)$, which returns $\{open\}$, disabling $open$ at level 1. Now, $\Sigma_1^{comdat} := \{contracted\}$, which is sent down to level 0. Upon receiving the updated Σ_1^{comdat} from level 1, condat $\Delta^{10}_{lo} := com(\{contracted\}, G^0_{lo}, 0d)$, which returns $\{contract\}$, disabling $contract$ at level 0. In this instance, the clear chain of command and control informs us that the control decisions to disable $open$ at level 1 and $contract$ at level 0 are because of the command decisions to disable $drop$ at level 2 and $contracted$ at level 1, respectively. The decisions in state $(2h, 1d, vlc, 0c)$ and the new decisions in state $(2h, 1e, vld, 0d)$ upon executing event $extend$ are summarized in Table II.

TABLE II

COMMAND AND CONTROL DECISIONS AT TWO DIFFERENT SYSTEM STATES

Level \ State	$(2h, 1d, vlc, 0c)$	$(2h, 1e, vld, 0d)$
2	$\underline{drop}$	$\underline{drop}$
1	$open$	$open$, $\underline{contracted}$
0	$motorleft$	$motorLeft$, $contract$

Note: Commands (or disabled virtual events) are underlined.

VII. CONCLUSION

In this paper, we have introduced, formulated and addressed nonblocking multi-world control for a class of multi-level hierarchical DES's in an organizational control architecture. Based on Theorem 1 and the ensuing discussion, our approach applies incremental synthesis of HCM, formalizing a common design practice of structuring the control of a DEO (2) bottom-up into a consistent and multi-world control hierarchy. Under command fairness and standard structural conditions, an optimal and nonblocking control law f_h that can govern this hierarchy top-down exists, by Theorem 2, as a dynamic programming recursion (6), over which an organizational control algorithm (Figs. 4 to 6) is obtained. By incremental synthesis of HCM and partial on-line computation of control [through μ_h of f_h (6)] incurring only linear time complexity, the approach can be of use in mitigating the complexity of off-line control synthesis along with increasing the on-line transparency of control operations. An example illustrates the bottom-up design modeling required of this approach and the on-line transparent control operations of the organizational control algorithm.

In conclusion, based on a standard hierarchical control theory [3], [5], [20], this research has laid a dynamic programming foundation of organizational control for a class of multi-level hierarchical DES's. A recent development presents a system decomposition method [28] that can be applied to an OCC DES G^h_{lo} for computing μ_h of control law f_h (6) even more efficiently. Many other issues of complexity for organizational control can be investigated in future, using advanced synthesis tools and incorporating architectural features such as modularity and decentralization, as well as structural formulation such as DES flexible marking [29].

ACKNOWLEDGMENTS

The author would like to thank Dong Sheng Zhao, Manh Tung Pham, and Chuan Ma for fruitful discussions on the various aspects of the work presented here. He would also like to thank the Associate Editor and all the anonymous referees for their critical but constructive comments on the review versions of this paper. However, the author remains solely responsible for the presented work.

REFERENCES

[1] K. T. Seow, "A dynamic programming approach to multi-level supervision," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Kobe, Japan, May 2009, pp. 908–913.

[2] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM Journal of Control and Optimization*, vol. 25, no. 1, pp. 206–230, January 1987.

[3] W. M. Wonham, *Supervisory Control of Discrete-Event Systems*. Systems Control Group, University of Toronto, Canada, July 2012 (Updated annually), http://www.control.toronto.edu/cgi-bin/dldes.cgi.

[4] H. Zhong and W. M. Wonham, "On the consistency of hierarchical supervision in discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 35, no. 10, pp. 1125–1134, October 1990.

[5] K. C. Wong and W. M. Wonham, "Hierarchical control of discrete-event systems," *Discrete Event Dynamic Systems : Theory and Applications*, vol. 6, no. 3, pp. 241–273, July 1996.

[6] A. E. C. da Cunha and J. E. R. Cury, "Hierarchical supervisory control based on discrete event systems with flexible marking," *IEEE Transactions on Automatic Control*, vol. 52, no. 12, pp. 2242–2253, December 2007.

[7] W. M. Wonham and P. J. Ramadge, "Modular supervisory control of discrete-event systems," *Mathematics of Control, Signals and Systems*, vol. 1, no. 1, pp. 13–30, January 1988.

[8] R. C. Hill, D. M. Tilbury, and S. Lafortune, "Modular supervisory control with equivalence-based abstraction and covering-based conflict resolution," *Discrete Event Dynamic Systems : Theory and Applications*, vol. 20, no. 1, pp. 139–185, March 2010.

[9] F. Lin and W. M. Wonham, "Decentralized supervisory control of discrete event systems," *Information Sciences*, vol. 44, pp. 199–224, 1988.

[10] K. Rudie and W. M. Wonham, "Think globally, act locally : Decentralized supervisory control," *IEEE Transactions on Automatic Control*, vol. 37, no. 11, pp. 1692–1708, November 1992.

[11] K. Rohloff and S. Lafortune, "On the synthesis of safe control policies in decentralized control of discrete event systems," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 1064–1068, June 2003.

[12] K. Hiraishi, "On solvability of a decentralized supervisory control problem with communication," *IEEE Transactions on Automatic Control*, vol. 54, no. 3, pp. 468–480, March 2009.

[13] K. Schmidt, M. H. de Queiroz, and J. E. R. Cury, "Hierarchical and decentralized multitasking control of discrete event systems," in *Proceedings of the 46th IEEE International Conference on Decision and Control*, New Orleans, LA, U.S.A, December 2007, pp. 5936–5941.

[14] B. Gaudin and H. Marchand, "Supervisory control of product and hierarchical discrete event systems," *European Journal of Control*, vol. 10, no. 2, pp. 131–145, 2004.

[15] K. Schmidt, T. Moor, and S. Perk, "Nonblocking hierarchical control of decentralized discrete event systems," *IEEE Transactions on Automatic Control*, vol. 53, no. 10, pp. 2252–2265, November 2008.

[16] K. Schmidt, H. Marchand, and B. Gaudin, “Modular and decentralized supervisory control of concurrent discrete event systems using reduced system models,” in *Proceedings of the 8th International Workshop on Discrete-Event Systems*, Ann Arbor, MI, USA, July 2006, pp. 149–154.
[17] L. Feng and W. M. Wonham, “Computationally efficient supervisor design: Abstraction and modularity,” in *Proceedings of the 8th International Workshop on Discrete-Event Systems*, Ann Arbor, MI, USA, July 2006, pp. 3–8.
[18] K. Schmidt and C. Breindl, “Maximally permissive hierarchical control of decentralized discrete event systems,” *IEEE Transactions on Automatic Control*, vol. 56, no. 4, pp. 723–737, April 2011.
[19] K. Q. Pu, “Modeling and Control of Discrete-Event Systems with Hierarchical Abstraction,” Graduate Department of Electrical and Computer Engineering, University of Toronto, Canada, Master of Applied Science (MASc) Thesis, March 2000.
[20] S. Yi, “Hierarchical Supervision with Nonblocking,” Graduate Department of Electrical and Computer Engineering, University of Toronto, Canada, Master of Applied Science (MASc) Thesis, June 2004.
[21] R. E. Bellman, *Dynamic Programming*. Princeton University Press, Princeton, USA, 1957.
[22] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages and Computation*. Reading, MA : Addison-Wesley, 1979.
[23] C. G. Cassandras and S. Lafortune, “Ch 2 : Languages and Automata,” in *Introduction to Discrete Event Systems*, 2nd ed. Springer-Verlag, New York, 2008, pp. 53–132.
[24] W. M. Wonham and P. J. Ramadge, “On the supremal controllable sublanguage of a given language,” *SIAM Journal of Control and Optimization*, vol. 25, no. 3, pp. 637–659, May 1987.
[25] R. Su and W. M. Wonham, “Supervisor reduction for discrete-event systems,” *Discrete Event Dynamic Systems : Theory and Applications*, vol. 14, no. 1, pp. 31–53, 2004.
[26] S. Eilenberg, *Automata, Languages and Machines : Volume A*. Academic Press, New York, 1974.
[27] R. C. Hill and D. M. Tilbury, “Modular supervisory control of discrete-event systems with abstraction and incremental hierarchical construction,” in *Proceedings of the 8th International Workshop on Discrete Event Systems (WODES 2006)*, Ann Arbor, Michigan, USA, July 2006, pp. 399–405.
[28] Q. H. Ngo and K. T. Seow, “Hierarchical control of discrete-event systems: A new command and control design based on feasible system decomposition,” in *Proceedings of the IEEE International Conference on Automation Science and Engineering (CASE’12)*, Seoul, Korea, August 2012, pp. 674–679.
[29] J. E. R. Cury, C. R. C. Torrico, and A. E. C. da Cunha, “Supervisory control of discrete event systems with flexible marking,” *European Journal of Control*, vol. 10, no. 1, pp. 47–60, 2004.