Command and Control of Discrete-Event Systems: Towards On-line Hierarchical Control Based on Feasible System Decomposition

Quang Ha Ngo and Kiam Tian Seow, *Senior Member, IEEE*

***Abstract*— A new operational design for hierarchical control of discrete-event systems is proposed. The design brings the structure of command and control from concept to realization for on-line control operation. For a command reference input, a new concept for output control feasibility of a discrete-event system modeled by a Moore automaton is characterized; and a system decomposition of a suitably structured Moore automaton into a controllable subsystem and an uncontrollable subsystem is formulated. Based on these results, the new command and control design for controller operation is realized, examined and discussed.**

***Note to Practitioners*— In the academic literature, the command and control theory of hierarchical control for discrete-event systems [2] is well established for meeting control specifications of safety with nonblockingness. This paper proposes a new operational design that uses the two-level structure of high-level command and low-level control, hitherto only a theoretical concept, for on-line translation of command to control during runtime hierarchical control. The design is realized with a two-level control algorithm and a reusable 'control technology' for the real discrete-event system at the low level, developed using the fresh theoretical findings in this paper. As the first steps in filling the theory-to-practice gap, the practical advantages of this new operational design include facilitating a deeper understanding of control with on-line causal clarity of command over control, and a significant reduction in off-line synthesis complexity with fast on-line control computation. In laying an algorithmic foundation for online hierarchical control, the proposed design has potential applications for many engineering control problems, where command and control is the inherent mode of runtime operation, or is needed to provide operational clarity when subjecting the control system to validation tests by simulation and observation. Problems include the design of logical command and control systems for supervising smart grids, traffic light systems and mass rapid transit networks, where the manager in the central command center may issue high-level commands to the operators to control the low-level physical system.**

***Index Terms*— Discrete-event systems, hierarchical control, on-line supervision, formal languages and automata.**

I. Introduction

Since its founding [3], the supervisory control of discrete-event systems (DES's) [2], [4], [5] has evolved into an increasingly important science of automation for many engineering control problems, with particular application success reported for automated manufacturing systems (see e.g., [6], [7], [8]). In this paper, for problems where command and control is required, a new theoretical and algorithmic basis for operational design of hierarchical discrete-event control is investigated.

Hierarchical control of DES's is a research area of current vitality. Broadly speaking, two types of event reporter maps have been proposed to model a DES hierarchy, namely, virtual [2], [9] (as is referred to in this paper) and natural [10], [11], [12] projections. In the basic two-level hierarchy, by the former projection, the high-level DES is modeled by virtual events, i.e., events abstracted or projected as a symbolically different set from that of the low-level DES, whereas, by the latter projection, the high-level DES is modeled by a subset of the events modeling the low-level DES. This paper contributes to the virtual projection paradigm; a new approach to hierarchical control realization in a command and control fashion is described.

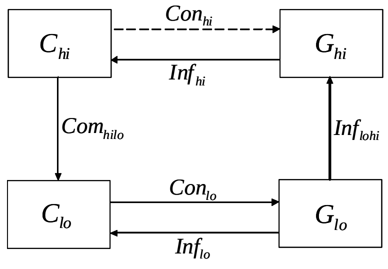


Fig. 1. The command & control concept for hierarchical control [13]

The proposed research is based on the seminal work on hierarchical control [13]. There, a two-level control hierarchy for DES's is first introduced, and is founded on the concept of command and control shown in Fig. 1. In a standard feedback fashion, the low-level DES G_{lo} is a real system to be supervised by the low-level controller C_{lo}, and the high-level DES G_{hi} is an abstract and simplified model to be supervised by the high-level controller C_{hi}. This is effected via the respective control channels, Con_{hi} and Con_{lo}, and the information feedback channels, Inf_{hi} and Inf_{lo}. The two levels are interconnected by the top-down command channel Com_{hilo} and the bottom-up information channel Inf_{lohi}. The controller C_{hi} is said to be virtual since G_{hi} is an abstract model driven by G_{lo} via the channel Inf_{lohi}; and it

This paper revises and extends the 2012 IEEE CASE version [1].

Q.H. Ngo and K.T. Seow are with the School of Computer Engineering, Nanyang Technological University, Republic of Singapore 639798. QUANG5@e.ntu.edu.sg, asktseow@ntu.edu.sg

This work was partially supported by the Singapore Agency for Science, Technology and Research (A*Star), under a Thematic Scientific Research Programme Grant.

summarizes the behaviour of G_{lo} that is important. Significant information from G_{lo} is reported to G_{hi} via the channel Inf_{lohi}, and in turn is fed back to C_{hi} via the channel Inf_{hi}. In response, the virtual control of C_{hi} on G_{hi}, via the channel Con_{hi}, is transmitted to C_{lo} as command via the channel Com_{hilo}, which in turn controls G_{lo} via the channel Con_{lo} as commanded.

In the setup described, the structural conditions for hierarchical consistency (HC) [13] and that with marking (HCM) [14], [15], to be met by a suitably formulated hierarchical information map for a properly structured DES, have been developed. Intuitively, achieving HC means that a specification task of the high-level controller can be realized supremally through low-level control, whereas achieving HCM means that it can be realized supremally with nonblockingness through nonblocking low-level control. However, under HC or HCM, the existing operational design is a flat low-level controller [2], [13], [14], [15], [16] synthesized from a base-level specification converted from a given high-level specification. Besides incurring high complexity of control synthesis, this design consolidates in a flat structure all the possible controls translated off-line from the regulated commands at the high level, with the result that the command and control structure as conceptualized in Fig. 1 is lost in implementation. In contrast to the consolidated flat structure, the design proposed in this paper implements the command and control structure, hitherto only a theoretical concept, through a two-level control algorithm for on-line translation of high-level commands for low-level control. In so doing, the control algorithm can facilitate a deeper understanding of control during runtime operation. To explain and discuss later, the novelty of this new design idea lies in basing the on-line command-to-control translation on some reusable 'control technology' developed for the low-level DES, and the algorithmic cooperation between the two levels.

Among recent efforts in the virtual projection paradigm, hierarchical control has been extended to handle partial observation of the low-level DES [9], and to ensure control robustness [17]. It has also been generalized to handle partial observation of the low-level system modeled as a fuzzy DES [18].

Related work based on the other paradigm of natural projection [10], [11], [19] develops different methods of subsystem synthesis aimed at reducing the computational effort. Recent efforts include proposing or enhancing different hierarchical design methods for nonblocking control synthesis (e.g., [11], [20], [21], [22], [23], [24]), by exploiting modularity and decentralization of DES's in a hierarchical structure. In essence, unlike a centralized controller that has to be constructed to act fully on the overall system, modular or decentralized controllers are synthesized to exercise their control only on that part of the DES that matches their event set. Hierarchical control has also been extended to handle partial observation of the low-level DES [12], and to general ω-languages using natural projection defined for infinite strings [25]. In another development, strict decoupling between levels achieved through the use of well-defined interfaces has been proposed (e.g., [26], [27]), followed by an extension to multiple levels [28] and an alternative interfaced-based approach [29].

The rest of this paper is organized as follows. In formalizing and examining the proposed design idea in the virtual projection paradigm, the necessary background and motivation are presented in Section II. In Section III, the system-output control feasibility and decomposition of a DES into a controllable subsystem and an uncontrollable subsystem, given a command reference input, are first formulated; and are illustrated by an example. Based on these results, the implementability of the new command and control design for hierarchical control is investigated in Section IV, and a resultant control algorithm is developed with its design complexity examined. Section V summarizes the contribution of the paper and points to future research.

II. Background & Motivation

A. Basics for Supervisory Control & DES

The relevant components of the language and automata framework for supervisory control [2], [3], [5], [30] are first reviewed.

1) Languages & Automata for DES Modeling: Let Σ be a finite set 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, including the empty string (sequence with no events) denoted by ε; and let $\Sigma^+ = \Sigma^* - \{\varepsilon\}$. A string s' is a prefix of s if $(\exists t \in \Sigma^*) s't = s$. It is a strict prefix of s, denoted by $s' < s$, 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 $L = \overline{L}$.

A regular language [31] is a language that can be generated by a finite state automaton [2]. Formally, an automaton G 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 the two languages generated by G: $L(G) = \{s \in \Sigma^* \mid \delta(s, q_0)!\}$ and $L_m(G) = \{s \in L(G) \mid \delta(s, q_0) \in Q_m\}$. $L(G)$ is called the prefix-closed language and $L_m(G)$, the marked language. By definition, $L_m(G) \subseteq L(G)$.

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 G is reachable if all its states are reachable, and coreachable if all its states are coreachable and so $\overline{L_m(G)} = L(G)$. Finally, automaton G is trim if it is both reachable and coreachable.

2) Supervisory Control: Let a reachable automaton $G = (Q, \Sigma, \delta, q_0, Q_m)$ model a DES, with the event set Σ partitioned into the controllable event set Σ_c and the uncontrollable event set Σ_u. A specification language $K \subseteq \Sigma^*$ is said to be controllable 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 marked sublanguage of the DES G that lies within the language K. This sublanguage can be generated by the trim automaton returned by $Supcon(G, K)$ [30], which has worst-case time complexity of $O(lm^2)$, where l and m are the respective cardinality of the event set Σ and the cross-product state set of DES G and the automaton modeling K.

$Supcon(G, K)$ is a supervisor automaton $S = (X, \Sigma, \xi, x_0, X_m)$, and is said to be nonblocking (for DES G) since $\overline{L_m(S)} = L(S)$ for a trim and hence coreachable $S = Supcon(G, K)$. Its associated control data set $Condat$ [2] w.r.t G is given by

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

where $\Delta(q, x) = \Sigma_c(q) - \Sigma_c(x)$. $Condat(G, S)$ has worst-case time complexity of $O(cm)$, where c is the cardinality of the controllable event set Σ_c.

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

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

$$\varepsilon \in L(f, G),$$

$$(\forall s \in L(f, G))(\forall s\sigma \in L(G))s\sigma \notin L(f, G) \Leftrightarrow \sigma \in f(s).$$

3) Moore Automaton for Low-level DES Modeling in Hierarchical Control: In the study of two-level hierarchical control (e.g., [13]), the real system at the base or low level is a DES that is equipped with an output function to send significant events to the high-level. A Moore automaton [32] is used to model a class of such DES's.

In general, a DES automaton G with event set Σ needs to be re-structured into a Moore automaton - an automaton[1] $G_{lo} = (Q, \Sigma, \delta, q_0, Q_m)$ associated with an information channel defined by a vocalization map $V : Q \to T \cup \{\tau_o\}$ - such that $L(G_{lo}) = L(G)$ and $L_m(G_{lo}) = L_m(G)$. T denotes the high-level (virtual) event set, and the symbol $\tau_o \notin T$ denotes a 'silent output'. The Moore construction [32] for the DES G is based on a given reporter map - a virtual projection $\theta : L(G) \to T^*$, 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 $\tau \in T$. For the constructed G_{lo}, the vocalization map V for every $s' \in L(G_{lo})$ is defined by

$$V(\delta(s', q_0)) = \begin{cases} \tau_o, & \text{if } s' = \varepsilon \\ & \text{or } \delta(s', q_0) \notin Q_{voc} \\ \tau \in T, & \text{otherwise,} \end{cases}$$

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

$$\delta(s\sigma, q_0) \begin{cases} \notin Q_{voc}, & \text{if } \theta(s\sigma) = \theta(s) \\ \in Q_{voc}, & \text{if } \theta(s\sigma) = \theta(s)\tau. \end{cases}$$

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

Through the map V, G_{lo} outputs events in T to drive some high-level θ-image model G_{hi} whenever it reaches a vocal state $q \in Q_{voc}$, and otherwise outputs the silent symbol $\tau_o \notin T$ to signal no 'significant' change for the high level. The high-level image of G_{lo} that results is an automaton G_{hi}, such that $L(G_{hi}) = \{\theta(s) \mid s \in L(G_{lo})\}$ and $L_m(G_{hi}) = \{\theta(s) \mid s \in L_m(G_{lo})\}$. G_{hi} is said to generate events of T under the θ-map on $L(G_{lo})$.

The high-level event set T of G_{hi} is partitioned into the controllable event set T_c and the uncontrollable event set T_u. To ensure that every such high-level event $\tau \in T$ defined and output by (G_{lo}, V) is unambiguously controllable or uncontrollable, the Moore automaton (G_{lo}, V), or simply G_{lo} when V is understood, needs to be refined so that it becomes output-control consistent (OCC). Formally, (the Moore transition structure of) a DES G_{lo} is said to be OCC [2] if, for every string $s \in L(G_{lo})$ of the form

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

where $s' \in \Sigma^+$, with

- $V(\delta(\sigma_1\sigma_2 \cdots \sigma_i, q_0)) = \tau_o \quad (1 \le i \le k-1)$,
- $V(\delta(s, q_0)) = \tau \in T$

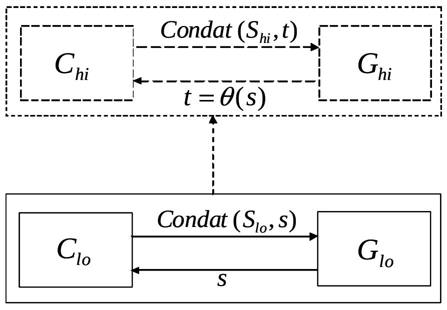
or, respectively,

- $V(\delta(s', q_0)) \in T$,
- $V(\delta(s'\sigma_1\sigma_2 \cdots \sigma_i, q_0)) = \tau_o \quad (1 \le i \le k-1)$,
- $V(\delta(s, q_0)) = \tau \in T$,

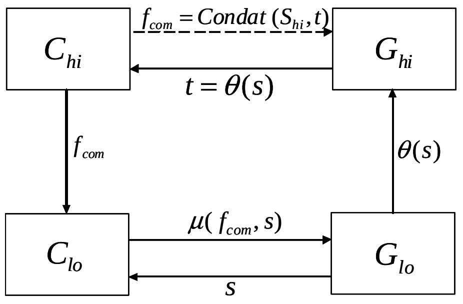
it is the case that

- if $\tau \in T_c$, then for some i $(1 \le i \le k)$, $\sigma_i \in \Sigma_c$,
- if $\tau \in T_u$, then for all i $(1 \le i \le k)$, $\sigma_i \in \Sigma_u$.

[1]Although the same 5-tuple notation is used as in Section II-A.2, it should be clear in the context that the structure of G_{lo} is in general not the same as that of a given DES G.



(a) Existing [2]: Uses base-level control only



(b) Proposed: Uses command and on-line control

Fig. 2. Operational designs for hierarchical control

B. *Research Motivation*

In [2], the command and control structure, theoretically conceptualized for hierarchical control as depicted in Fig. 1, is not implemented as such for runtime operation, once hierarchical consistency between a low-level DES G_{lo} and its high-level abstraction G_{hi} is assured[2]. Instead, given a high-level specification E for DES G_{hi}, hierarchical control according to high-level supervision $S_{hi} = Supcon(G_{hi}, E)$ is actually implemented using a low-level supervisor automaton S_{lo}[3], which is either $Supcon(G_{lo}, \theta^{-1}(E))$ or $Supcon(\overline{G}_{lo}, \theta^{-1}(L(S_{hi})))$, where $\overline{G}_{lo}$ is G_{lo} but with all its states marked, and the associated control data. Besides the issue of synthesis complexity, such a consolidated 'flat' design, shown in Fig. 2(a), may not be adequate for two other reasons. Firstly, command and control may be the inherent mode of runtime operation for the engineering control problem of interest. Secondly, there is limited operational clarity when command is not explicitly linked to control on-line to show what stepwise controls are exercised in carrying out the commands from above. Such on-line causal clarity of command over control is important to designers, when subjecting the control system to validation tests by simulation and observation.

In this paper, an operational design linking command and control is proposed as depicted in Fig. 2(b), bringing the concept depicted in Fig. 1 to realization. Referring to Fig. 2(b), the design uses the high-level supervisor automaton S_{hi} and a f_{com}-output control law μ, also called a command-to-control 'transfer' function, to continually translate command data in f_{com} to control data during runtime. Clearly, this design obviates the need for the low-level supervisor automaton S_{lo}. The details and discussion of this alternative approach are presented in the rest of the paper.

III. System-Output Control with Command Input

A. *Preliminaries*

Consider the feedback control loop shown in Fig. 3.

The closed loop interconnecting the controller and DES allows the former to modify the latter's dynamics based on feedback. In a standard setting, this loop applies a principle of feedback and control, namely, the control design of dynamics through feedback. Set in the Moore DES paradigm, the design dynamics refers to that generated by the system under control, without vocalizing or outputting via the reporter the virtual events specified in the command reference input f_{com} (from the instant the reference is set). Based on the system feedback string s generated under control, the controller continually computes the next control input.

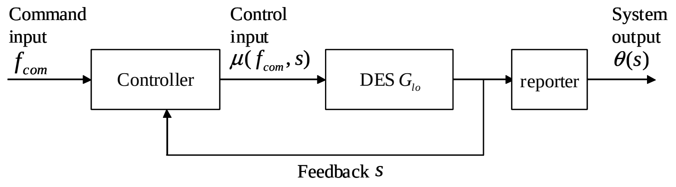


Fig. 3. Feedback control of a DES with command input and system output

The fundamental problem, then, is whether a feasible output control law μ for the reference input f_{com} exists for controller realization or implementation. To explain more formally later, a feasible output control law can prevent the vocalization of events specified in the reference input set. In the rest of this section, this is theoretically investigated for a single reference and an extension to some regulated sequences of references.

B. *Problem Approach: Uncontrollable & Controllable States*

Without loss of generality, in this research, a Moore automaton is always constructed such that its initial state is non-vocal or silent.

Now, consider a (reachable) Moore DES G_{lo} with $\theta : L(G_{lo}) \to T^*$. For $q \in Q_{voc}, V(q) = \tau$, let

$$B_u[q, V(q)] = \{p \in Q - \{q_0\} \mid (\exists t \in \Sigma_u^*)(\delta(t,p) = q \;\&\; (\forall t' < t) V(\delta(t',p)) = \tau_o)\}.$$

Excluding the initial state $q_0 \in Q$, $B_u[q, V(q)]$[4] defines a set of τ-uncontrollable states which are non-vocal other than the state q, and each non-vocal state can reach state q via a string of uncontrollable events defined at non-vocal states.

[2]The reader is referred to the literature for details of HC [2], [13], HCM [2], [14], [15], and a more specialized concept of HCM [2] relevant to this paper (mentioned later in Remark 1).

[3]It can be computed as $Supcon(G_{lo}, \theta^{-1}(E))$ if HCM holds; and as $Supcon(\overline{G}_{lo}, \theta^{-1}(L(S_{hi})))$ if the more specialized concept [2] of HCM holds.

[4]The definition of $B_u[q, V(q)]$ revises the initial formulation presented in the conference version [1].

The complete set of τ-uncontrollable states is

$$B_u^\tau = \bigcup_{V(q)=\tau} B_u[q, V(q)]. \tag{1}$$

Now, define

$$B_c[q, V(q)] = Q - B_u[q, V(q)].$$

Then

$$\begin{aligned} B_c^\tau &= \bigcap_{V(q)=\tau} B_c[q, V(q)] \\ &= Q - B_u^\tau \end{aligned}$$

is said to define the complete set of τ-controllable states if the following condition holds for virtual event $\tau \in T$:

$$\begin{aligned} &(\forall q' \in Q_{voc} \cup \{q_0\})(\forall w \in \Sigma^+) \\ &\qquad (V(\delta(w, q')) = \tau) \Rightarrow w \notin \Sigma_u^*. \end{aligned} \tag{2}$$

Informally, Condition (2) asserts that from the initial state or any vocal state, a nonempty string leading to a state vocalizing the high-level event τ is not uncontrollable.

If Condition (2) holds, then the following condition also holds:

$$\begin{aligned} &(\forall s\sigma \in L(G_{lo}), \sigma \in \Sigma_c)(\forall t \in \Sigma_u^*) \\ &(V(\delta(s\sigma t, q_0)) = \tau) \Rightarrow (\forall t' < t) V(\delta(s\sigma t', q_0)) = \tau_o. \end{aligned} \tag{3}$$

Informally, Condition (3) asserts that if a nonempty string of uncontrollable events immediately following a controllable event leads the DES to a reachable state q vocalizing the high-level event τ, all the states except q via which the uncontrollable string traverses are non-vocal.

In what follows, w.r.t $f_{com} \subseteq T$ of a DES G_{lo},

$$\begin{aligned} B_c^{f_{com}} &= \bigcap_{\tau \in f_{com}} B_c^\tau \\ &= Q - B_u^{f_{com}} \end{aligned} \tag{4}$$

is said to be the (f_{com}-)controllable state set, where

$$B_u^{f_{com}} = \bigcup_{\tau \in f_{com}} B_u^\tau \tag{5}$$

is the (f_{com}-)uncontrollable state set, if the following condition holds for f_{com}:

$$\begin{aligned} &(\forall q' \in Q_{voc} \cup \{q_0\})(\forall w \in \Sigma^+) \\ &\qquad (V(\delta(w, q')) \in f_{com}) \Rightarrow w \notin \Sigma_u^*. \end{aligned} \tag{6}$$

That Condition (6) holds implies that the following condition also holds[5]:

$$\begin{aligned} &(\forall s\sigma \in L(G_{lo}), \sigma \in \Sigma_c)(\forall t \in \Sigma_u^*) \\ &(V(\delta(s\sigma t, q_0)) \in f_{com}) \Rightarrow (\forall t' < t) V(\delta(s\sigma t', q_0)) = \tau_o. \end{aligned} \tag{7}$$

Then the command-to-control 'transfer' function μ, as shown in the proposed operational design depicted in Fig. 2(b), can be written as

$$\begin{aligned} \mu(f_{com}, s) = &\{\sigma \in \Sigma_c \mid (\exists t \in \Sigma_u^*)(V(\delta(s\sigma t, q_0)) \in f_{com} \\ &\& \ (\forall t' < t) V(\delta(s\sigma t', q_0)) = \tau_o)\}. \end{aligned} \tag{8}$$

[5]That (2) implies (3) and (6) implies (7) can be proved by contradiction. The proofs are simple and straightforward, and hence are omitted.

This function is an output control law that is said to be 'permissive', in that it disables a controllable event only when the event occurrence can otherwise lead the system uncontrollably to a state vocalizing a virtual event in the reference input f_{com}. Based on the characterization of B_u^τ (1) and the definition of $B_u^{f_{com}}$ (5), this law can be rewritten as

$$\mu(f_{com}, s) = \{\sigma \in \Sigma_c \mid \delta(s\sigma, q_0)! \& \delta(s\sigma, q_0) \in B_u^{f_{com}}\}. \tag{9}$$

C. Output Control Feasibility

Theorem 1: Consider a Moore DES (G_{lo}, V) constructed with $\theta : L(G_{lo}) \to T^*$, and an arbitrary $f_{com} \subseteq T$. Then for all $s \in L(G_{lo})$, μ (9) is the permissive control law for (f_{com}, s) such that $V(\delta(s, q_0)) \in f_{com}$ or $\delta(s, q_0) \in B_c^{f_{com}}$ implies

$$(\forall \sigma \in \Sigma_u(\delta(s, q_0)))\delta(s\sigma, q_0) \notin B_u^{f_{com}}$$

iff $B_c^{f_{com}}$ is the controllable state set.

Proof: See Appendix A. ■

Intuitively, Theorem 1 states the necessary and sufficient condition for the control law (9) to be feasible for (f_{com}, s). Being feasible means the output control law can always prevent the system from next entering $B_u^{f_{com}}$ if it is in a vocal state that just vocalized an event in f_{com} or in $B_c^{f_{com}}$, and thus guarantees subsequent nonoccurrence of events in f_{com}.

D. The Decomposition Theorem

In partitioning the states of a Moore DES $G_{lo} = (Q, \Sigma, \delta, q_0, Q_m)$ w.r.t an arbitrary $f_{com} \subseteq T$, it can be said that $B_c^{f_{com}}$ (4) induces a subsystem, denoted by $G_c^{f_{com}}$, that is controllable if it is the controllable state set, and $B_u^{f_{com}}$ (5) induces the corresponding uncontrollable subsystem denoted by $G_u^{f_{com}}$. The pair $(G_c^{f_{com}}, G_u^{f_{com}})$ is then called a feasible system decomposition.

Formally, the subsystems $G_c^{f_{com}}$ and $G_u^{f_{com}}$ can be modeled as automata[6] derived from the Moore automaton G_{lo}. For the subsystem $G_c^{f_{com}} = (Q_c, \Sigma, \delta_c, Q_{0,c}, Q_{m,c})$, the state set is $Q_c = B_c^{f_{com}} \subseteq Q$, the transition function $\delta_c : \Sigma \times Q_c \to Q_c$ is a restriction of δ to $\Sigma \times Q_c$, the initial state set is $Q_{0,c} = Q_c \cap (Q_{voc} \cup \{q_0\})$, defining an initial state as where f_{com} is computed following system initialization or a high-level event occurrence, and the marked state set is $Q_{m,c} = Q_m \cap Q_c$.

For the uncontrollable subsystem $G_u^{f_{com}} = (Q_u, \Sigma, \delta_u, Q_{0,u}, Q_{m,u})$, the state set is $Q_u = B_u^{f_{com}} \subseteq Q$, the transition function $\delta_u : \Sigma \times Q_u \to Q_u$ is a restriction of δ to $\Sigma \times Q_u$, the initial state set is $Q_{0,u} = \{q \in Q_u \mid (\exists q' \in B_c^{f_{com}})(\exists \sigma \in \Sigma) q = \delta(\sigma, q')\}$, defining an initial state as the first state of entry into the subsystem, and the marked state set is $Q_{m,u} = Q_m \cap Q_u$.

Each subsystem can be represented by a (possibly unconnected) subgraph of the state-transition graph representing the DES G_{lo}.

Below, Theorem 2 states that a (feasible) system decomposition, as conceptually depicted in Fig. 4, exists for a DES G_{lo} that is OCC, with command input $f_{com} \subseteq T_c$.

[6]Note that, with some abuse of notation, the initial state q_0 in the usual 5-tuple formalization of an automaton is extended to a state set.

The proof of Theorem 2 requires the following lemma.

Lemma 1: For every $\tau \in T_c$ of an OCC DES G_{lo}, Condition (2) holds.

Proof: See Appendix B. ■

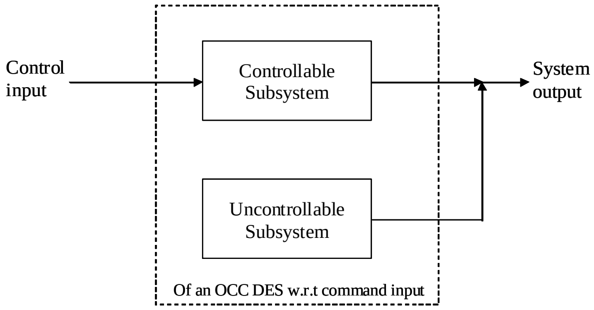


Fig. 4. Conceptual decomposition of a DES

Theorem 2: An OCC Moore automaton G_{lo}, w.r.t an arbitrary $f_{com} \subseteq T_c$, can be decomposed into a controllable subsystem $G_c^{f_{com}}$ and an uncontrollable subsystem $G_u^{f_{com}}$.

Proof: See Appendix C. ■

Illustrative Example: Consider the OCC DES $G_{lo} = (Q, \Sigma, \delta, q_0, Q_m)$ with high-level event set T for system output, as adapted from [13] and shown in Fig. 5. The DES model is represented by an edge-labeled 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' labeled with the symbol σ of an event whose occurrence it represents. The symbol for a controllable and an uncontrollable event is indicated with a superscript '+' and '-', respectively. $\Sigma = \{\sigma_i^+ \mid i = 1,2,3,4\} \cup \{\sigma_j^- \mid j = 5,6,7,8,9\}$, and $T = \{\tau_i^+ \mid i = 1,2,3\} \cup \{\tau_4^-\}$. Every state is denoted by a number. The initial state $q_0 = 0$ 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.

The τ-uncontrollable state sets for T_c are given in the following: $B_u^{\tau_1^+} = B_u[1, V(1)] \cup B_u[8, V(8)]$; $B_u^{\tau_2^+} = B_u[5, V(5)]$ and $B_u^{\tau_3^+} = B_u[7, V(7)]$, where $B_u[q, V(q)]$ is respectively depicted in Fig. 5. One can easily verify that every arbitrary $f_{com} \subseteq T_c$ decomposes the OCC G_{lo} into a controllable and an uncontrollable part (Theorem 2), such that when the system is in a state of the controllable part (induced by $Q - B_u^{f_{com}}$) or a state that has vocalized a high-level event in f_{com} , it can be prevented from entering the uncontrollable part by disabling some controllable low-level events (Theorem 1), and hence disallowing the events in f_{com} as desired.

IV. Command & Control Operational Design

A. The Implementability Theorem

The implementability result is now established for the command and control design shown in Fig. 2(b). The result is based on the feasibility and decomposition Theorems 1 and 2, respectively. Formally, it shows that the design can be realized for a controllable high-level specification, w.r.t a high-level DES G_{hi} obtained under a language map θ on a low-level DES G_{lo} that is OCC.

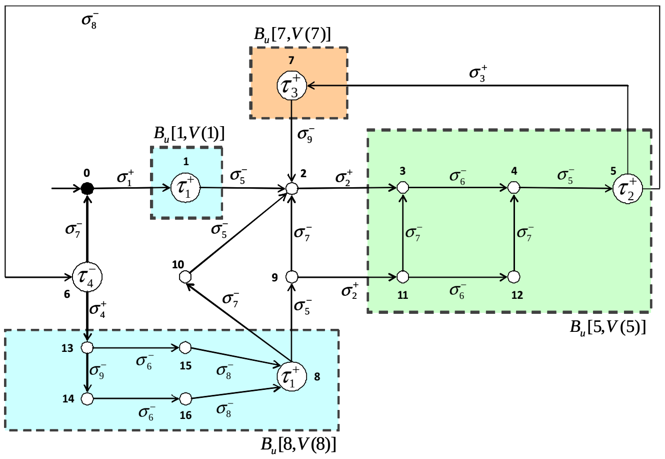


Fig. 5. An example OCC DES G_{lo} and its τ-uncontrollable state sets, $\tau \in T_c$

Theorem 3: Given that an OCC Moore automaton G_{lo} is constructed with $\theta : L(G_{lo}) \to T^*$ such that $L(G_{lo}) \mapsto L(G_{hi})$ and $L_m(G_{lo}) \mapsto L_m(G_{hi})$; and $E \subseteq T^*$. Let $S_{hi} = Supcon(G_{hi}, E)$ and $S_{lo} = Supcon(\overline{G}_{lo}, \theta^{-1}(L(S_{hi})))$, where $\overline{G}_{lo}$ is G_{lo} but with all its states marked. Assume $L(S_{hi}) \neq \emptyset$. Then

$$L(\mu, G_{lo}) = L(S_{lo}),$$

where $\mu(f_{com}, s)$ (9) is the permissive control law, with $s \in L(S_{lo})$ and $f_{com} = Condat(S_{hi}, \theta(s))$. ■

Proof: See Appendix D. ■

Since $B_u^{f_{com}} = \bigcup_{\tau \in f_{com}} B_u^{\tau}$ according to (5), under Theorem 3, B_u^{τ} (1) for all $\tau \in T_c$ form the set of τ-components that is said to provide the 'control technology' for implementing command and control using the law μ (9).

Remark 1: Note that $L(S_{hi}) = \overline{L_m(S_{hi})}$. In general, $\theta(L(S_{lo})) \subseteq L(S_{hi})$. Thus Theorem 3 alone does not imply that the high-level controllable sublanguage $L(S_{hi})$ can be fully met by low-level control via the law μ (9), and without high-level blocking during runtime caused by a low-level string $s \in L(S_{lo})$ that cannot be extended, by any $s' \in \Sigma^*$, to $ss' \in L(S_{lo})$ such that $\theta(ss') \in L_m(S_{hi})$. To meet such a high-level (nonblocking) expectation through the control law μ, the assurance of both HC [13] and hierarchical nonblockingness (N) [2] - a more specialized concept (HCN) of HCM - must first be provided, for which additional structural conditions have been developed [2]. ■

B. Algorithmic Procedures

Based on Theorem 3, hierarchical control depicted in Fig. 2(b) can be realized by a control algorithm computing the law (9) on-line, with a command and a control level coupled by top-down command f_{com} and bottom-up event vocalization feedback V implementing the reporter map θ.

The control algorithm contains two procedures `HI-manager` and `LO-operator`, shown respectively in Figs. 6 and 7. Through the command of the high-level DES G_{hi} by `HI-manager`, `LO-operator` computes on-line the control of the low-level DES G_{lo}. Without loss of generality, the algorithm is assumed to be non-terminating, hence the use of the unconditional 'while' loop in the two procedures.

Let ξ and x_0 be the transition function and initial state of the high-level supervisor automaton S_{hi}, respectively. The following notation is used in the procedures: For an evolving $t \in L(G_{hi})$ and $s \in L(G_{lo})$, $\theta(s) = t$, the current high-level state of S_{hi} is $x = \xi(t, x_0)$; the current low-level state of G_{lo} is $q = \delta(s, q_0)$; the command or high-level control data to transmit is $f_{com}(S_{hi}, x) = Condat(S_{hi}, t)$ and the translated low-level control data is

$$f_{con}(f_{com}, q) = \mu(f_{com}, s), \tag{10}$$

where μ is the control law (9).

To sharpen causal clarity to a finer level of ascertaining the control data from each individual event command, the law f_{con} can be refined to $f_{con}(f_{com}, q) = \bigcup_{\tau \in f_{com}} f_{con}(\tau, q)$, where $f_{con}(\tau, q) = \{\sigma \in \Sigma_c \mid \delta(\sigma, q)!\&\delta(\sigma, q) \in B_u^\tau\}$ is the τ-commanded control data set, i.e., the set of low-level events to be disabled as commanded by (the control data for virtual event) $\tau \in f_{com}$.

Procedure `HI-manager` **(High Level)**

```
begin
    initialize
        Initialize state in $S_{hi}$;
        Use initial state $x_0$ to retrieve off-line Condat $f_{com}$;
        Send $f_{com}$ down to low level;
    while true do
        Wait to receive high-level event occurrence (from $T$);
        Update state in $S_{hi}$;
        Use current state $x$ to retrieve off-line Condat $f_{com}$;
        Send $f_{com}$ down to low level;
end
```

Fig. 6. Procedural realization for high-level controller C_{hi}

Procedure `LO-operator` **(Low Level)**

```
begin
    initialize
        Initialize state in $G_{lo}$;
        Wait to receive updated $f_{com}$ from high level;
        Use current $f_{com}$ and state $q$ to compute Condat $f_{con}$, and apply it to
        $G_{lo}$;
    while true do
        Wait to receive low-level event occurrence (from $\Sigma$);
        Update state in $G_{lo}$;
        if current state $q$ is vocal then
            Send vocalized event $V(q)$ up to high level;
            Wait to receive updated $f_{com}$ from high level;
        Use current $f_{com}$ and state $q$ to compute Condat $f_{con}$, and apply
        it to $G_{lo}$;
end
```

Fig. 7. Procedural realization for low-level controller C_{lo}

C. Computational Complexity

Off-line Complexity Reduction: It is assumed that the hierarchical structuring of a low-level DES to achieve HCN [2] (see Remark 1) is already applied. Under the same HCN setup for a trim low-level DES, the worst-case time complexity comparison is made between the existing design [2], [15] and the proposed design for hierarchical controller operation, as follows.

- To implement the existing controller design (realized by S_{lo} in Theorem 3 and the associated control data), the low-level specification automaton has to be computed [15] first for the inverse mapping θ^{-1} of the closure of the supremal controllable sublanguage of a given high-level specification $E \subseteq T^*$, before $Supcon$ and $Condat$ are computed accordingly for the low-level DES .
- To implement the proposed design (realized by the control algorithm in Figs. 6 and 7 regulating f_{con} (10)), every τ-uncontrollable state set B_u^τ (1) for $\tau \in T_c$ needs to be computed for the low-level DES G_{lo} (to implement μ in Theorem 3 for f_{con}), along with $Supcon$ and $Condat$ for the high-level DES G_{hi}.

The various component complexities and the overall complexity are summarized in Table I. The complexity of B_u^τ for all $\tau \in T_c$ is based on the analysis in Appendix E. The rest are based on the results in [2], [15], [33].

The low-level specification automaton generating the language $\theta^{-1}(L(S_{hi})) \subseteq L(G_{lo})$, $S_{hi} = Supcon(G_{hi}, E)$, is constructed using the method developed in [15, p. 58], and is assumed to be of minimal state cardinality for the complexity analysis. This method, in the final step, obtains the trim specification automaton by computing a natural projection $P : \overline{L_a} \rightarrow \Sigma^*$, where $L_a \subseteq (\Sigma \cup T)^*$ is the closed language generated by the (trim) cartesian product of two automata. One automaton is a T-embedded and completely state-marked version of the trim DES G_{lo}, and the other is a Σ-self-looped version of the automaton that is S_{hi} but is completely state-marked to model $L(S_{hi})$. Both these versions are computed in earlier steps of the method. For an optimistic (or best) worst-case complexity analysis of the existing design, it is assumed that the projection P is an observer of the (closed) language L_a. By the complexity results in [33] and using the notation defined in Table I, the exponential worst case of computing natural projection, in time complexity of $O(2^{n_e n_{hi}(n_{voc}+n)})$ and returning an automaton of state cardinality $O(2^{n_e n_{hi}(n_{voc}+n)})$, can then be avoided. In fact, the automaton returned has worst-case state cardinality of only $n_e n_{hi}(n_{voc} + n)$. However, this projection step alone still incurs a high polynomial time complexity of $O((n_e n_{hi}(n_{voc}+n))^4 + l(n_e n_{hi}(n_{voc}+n))^5 + l^2(n_e n_{hi}(n_{voc}+n))^7)$, as derived based on the complexity result in [33] for such an observer P.

The hierarchical construction for HCN of (G_{lo}, G_{hi}) [2] entails structuring the reporter map θ as an observer. It is assumed that G_{hi} is a minimal-state recognizer of the language $\theta(L_m(G_{lo}))$. From the results on observers [34] and using the notation defined in Table I, it then follows that $n_{hi} \leq n$. Thus, implementing the proposed design incurs an off-line control synthesis complexity that is significantly lower in general due to the following:

1) the smaller state cardinality of the high-level DES involved in $Supcon$ synthesis instead of the low-level DES;

TABLE I

CONTROL OPERATIONAL DESIGN OF A HIGH-LEVEL REGULAR SPECIFICATION LANGUAGE $E \subseteq T^*$ FOR A HCN PAIR (G_{lo}, G_{hi}): AN OFF-LINE, WORST-CASE TIME COMPLEXITY COMPARISON

Component $O(.)$	For existing design [2], [15] [Fig. 2(a)]	For proposed design [Fig. 2(b)]
High-level $Supcon$	$l_{hi}(n_e n_{hi})^2$	$l_{hi}(n_e n_{hi})^2$
High-level $Condat$	—	$v n_e n_{hi}$
θ^{-1} mapping	$n_{voc} + l n_e n_{hi} + l n_e n_{hi}(n_{voc}+n) + (n_e n_{hi})^4 (n_{voc}+n)^4 + l(n_e n_{hi})^5 (n_{voc}+n)^5 + l^2 (n_e n_{hi})^7 (n_{voc}+n)^7$	—
Low-level $Supcon$	$l(n_e n_{hi}(n_{voc}+n))^2$	—
Low-level $Condat$	$c n_e n_{hi}(n_{voc}+n)$	—
B_u^τ for all $\tau \in T_c$	—	$u(2n_{voc}-v)(n-n_{voc}-1) + vu(n-n_{voc}-1)^2$
Overall complexity	$n_{voc} + c n_e n_{hi}(n_{voc}+n) + l n_e n_{hi}(1+n_{voc}+n) + n_e^2 n_{hi}^2 (l_{hi} + l(n_{voc}+n)^2) + n_e^4 n_{hi}^4 (n_{voc}+n)^4 + l n_e^5 n_{hi}^5 (n_{voc}+n)^5 + l^2 n_e^7 n_{hi}^7 (n_{voc}+n)^7$	$v n_e n_{hi} + l_{hi} n_e^2 n_{hi}^2 + u(2n_{voc}-v)(n-n_{voc}-1) + vu(n-n_{voc}-1)^2$

Nomenclature:

For G_{lo}	**: Cardinality**	**For** G_{hi}	**: Cardinality**
n	: state set of G_{lo} with event set Σ	n_{hi}	: state set of G_{hi} with event set T
n_{voc}	: vocal state set of G_{lo}	n_e	: state set of the trim automaton for E
l	: event set Σ	l_{hi}	: event set T
c	: controllable event set Σ_c	v	: controllable event set T_c
u	: uncontrollable event set Σ_u		

2) the τ-uncontrollable state sets which can be computed in polynomial time complexity in the (non-initial) non-vocal state cardinality of the low-level DES; and
3) not having at all to compute the low-level specification automaton for the inverse mapping θ^{-1} of the closure of the supremal controllable sublanguage of a given high-level specification language.

Besides, computing τ-uncontrollable state sets is a one-off exercise for a given DES G_{lo}. For a different high-level specification, only the high-level supervisor needs to be recomputed. In this sense, B_u^τ (1) for $\tau \in T_c$ is reusable.

Low On-line Complexity: Not surprisingly, the potentially significant savings in off-line synthesis comes at the expense of incurring on-line low-level control (data) computation. Fortunately, however, this on-line time complexity is generally low, as explained below.

In the control algorithm, f_{con} (10) is recomputed each time an event in G_{lo} occurs, and in absorbing the cardinality of Σ_c in the constants of $O(.)$, each on-line computation incurs, in the worst case, only a linear time complexity of $O(r)$, where r is the cardinality of the set $B_u^{f_{com}}$, given $f_{com} \subseteq T_c$.

In a related but different work [35], a baseline procedure for computing the control law (8) on-line has linear time complexity of $O(n)$ in the worst case, where n is the state cardinality of the low-level DES G_{lo}. When compared with this procedure, computing the equivalent law f_{con} in $O(r)$ as used in the proposed design is faster. This is because the state cardinality r of a set $B_u^{f_{com}}$ is often very much smaller than the state cardinality n of the low-level DES.

V. CONCLUSION

Based on the results of output control feasibility and system decomposition, the command and control design proposed for hierarchical control is shown to be implementable. The implementation entails the off-line computation of the high-level controlled DES and the reusable τ-uncontrollable state sets, instead of the larger scale, low-level controlled DES. This operational design is attractive as it offers the intrinsic merit of furnishing causal clarity of command over control during on-line operation, along with off-line complexity reduction and fast on-line computation. These practical advantages are useful for engineering control problems, where the required runtime operational mode is command and control, or the control solution needs to be subjected to validation tests by simulation and observation.

To do away with using the high-level supervisor automaton without significantly increasing on-line computation, one approach might be to deploy on-line limited lookahead control [36] at the high level of the command and control hierarchy. And together with treating every high-level specification as a control task, the foundation laid in this paper could pave the way towards on-line hierarchical control of DES's that can support operationally clearer command-to-control transfer and flexible sequential or nonconcurrent multi-tasking. This is the subject for future research.

APPENDIX

A. Proof of Theorem 1

(If) The sufficiency proof proceeds as follows. For $s \in L(G_{lo})$, let $q' = \delta(s, q_0)$. There are two cases to consider. To prove by contradiction for each case, assume that there exists a $\sigma \in \Sigma_u(q')$ such that $\delta(\sigma, q') \in B_u^{f_{com}}$.

- *Case 1*: Suppose $q' \in B_c^{f_{com}} - (Q_{voc} \cup \{q_0\})$. Then by definition of $B_u^{f_{com}}$ and the assumption, it follows that $q' \in B_u^{f_{com}}$. Therefore, $q' \in B_u^{f_{com}} \cap B_c^{f_{com}}$, contradicting the fact that $B_u^{f_{com}} \cap B_c^{f_{com}} = \emptyset$.
- *Case 2*: Suppose $q' \in B_c^{f_{com}} \cap (Q_{voc} \cup \{q_0\})$ (implying $V(q') \notin f_{com}$), or $V(q') \in f_{com}$ (i.e., $q' \in B_u^{f_{com}} \cap Q_{voc}$). Then by definition of $B_u^{f_{com}}$ and the assumption, there is an uncontrollable string $w = \sigma t \in \Sigma_u^+$ such that $V(\delta(\sigma t, q')) \in f_{com}$, contradicting the fact that w contains a controllable event by Condition (6).

(Only If) The necessity proof proceeds as follows. Given an arbitrary $B_c^{f_{com}}$, $f_{com} \subseteq T$:

- Suppose, for every string $s \in L(G_{lo})$ such that $q' = \delta(s, q_0) \in B_c^{f_{com}}$, or $V(q') \in f_{com}$ and hence $q' \in Q_{voc}$,
$$(\forall \sigma \in \Sigma_u(q'))\delta(\sigma, q') \notin B_u^{f_{com}}.$$
Since by definition, $B_u^{f_{com}} \cup B_c^{f_{com}} = Q$ and $B_u^{f_{com}} \cap B_c^{f_{com}} = \emptyset$, $\delta(\sigma, q') \notin B_u^{f_{com}}$ iff $\delta(\sigma, q') \in B_c^{f_{com}}$. By definition, for $V(q) = \tau \in f_{com}$, $B_u[q, \tau]$ of $B_u^\tau \subseteq B_u^{f_{com}}$ contains non-vocal states (excluding the initial state $q_0 \in Q$) and the state $q \in Q_{voc}$, such that each non-vocal member state can reach state q via a string of uncontrollable events defined at non-vocal member states. As a result, if $q' = \delta(s, q_0) \in B_c^{f_{com}}$ or $V(q') \in f_{com}$, then every event $\sigma \in \Sigma$ for which $\delta(\sigma, q') \in B_u^{f_{com}}$ must be controllable; and so is it if $q' \in B_c^{f_{com}} \cap (Q_{voc} \cup \{q_0\})$ or $V(q') \in f_{com}$. Therefore, every string $w \in \Sigma^+$ that can bring every such $q' \in Q$ to some state q vocalizing a $\tau \in f_{com}$, i.e., $V(\delta(w, q')) = V(q) = \tau \in f_{com}$, must contain at least one controllable event σ_c, such that $w = s'\sigma_c t$ for some $s' \in \Sigma^*$ and $t \in \Sigma_u^*$. Together with the fact that $q_0 \in B_c^{f_{com}}$ and G_{lo} is reachable, it follows that Condition (6) holds, since $w = s'\sigma_c t \notin \Sigma_u^*$.

Hence the theorem.

B. Proof of Lemma 1

Consider a string $s' \in L(G_{lo})$, where $q' = \delta(s', q_0) \in Q_{voc} \cup \{q_0\}$. For an OCC G_{lo}, if the string s' can be extended to a state vocalizing an event $\tau \in T_c$, i.e., $(\exists w \in \Sigma^+)(q = \delta(s'w, q_0) = \delta(w, q') \in Q_{voc}$ and $V(q) = \tau \in T_c)$, then $w = s\sigma_c t$ for some $s \in \Sigma^*$, $\sigma_c \in \Sigma_c$ and $t \in \Sigma_u^*$. This in turn implies that Condition (2) holds since $w = s\sigma_c t \notin \Sigma_u^*$. Hence the lemma.

C. Proof of Theorem 2

For a Moore automaton G_{lo} that is OCC, by Lemma 1, Condition (2) holds for every $\tau \in T_c$. It is easy to show that, for all $\tau \in f_{com} \subseteq T_c$, the conjunctions of Condition (2) constitute Condition (6). Hence, for an arbitrary $f_{com} \subseteq T_c$, the set $B_c^{f_{com}}$ (4) is controllable, inducing a controllable subsystem $G_c^{f_{com}}$, with $B_u^{f_{com}}$ (5), where $B_c^{f_{com}} \cap B_u^{f_{com}} = \emptyset$, inducing the corresponding uncontrollable subsystem $G_u^{f_{com}}$. Hence the theorem.

D. Proof of Theorem 3

Given that an OCC Moore automaton G_{lo} is constructed with $\theta : L(G_{lo}) \to T^*$ such that $L(G_{lo}) \mapsto L(G_{hi})$ and $L_m(G_{lo}) \mapsto L_m(G_{hi})$; and that for some $E \subseteq T^*$, $S_{hi} = Supcon(G_{hi}, E)$ and $S_{lo} = Supcon(\overline{G}_{lo}, \theta^{-1}(L(S_{hi})))$, where $\overline{G}_{lo}$ is G_{lo} but with all its states marked, and $f_{com} = Condat(S_{hi}, \theta(s))$ for an $s \in L(S_{lo})$.

For an arbitrary output subset $f \subseteq T_c$ of the OCC G_{lo}, by Theorem 2, the controllable state set B_c^f exists.

Let $L(S'_{hi}) = \theta(L(S_{lo}))$ and $f'_{com} = Condat(S'_{hi}, \theta(s)) \subseteq T_c$ with $s \in L(S_{lo})$. It follows that $L(S'_{hi}) \subseteq L(S_{hi})$ since $f'_{com} \supseteq f_{com}$ in general for the same control data $Condat(S_{lo}, s)$ following $s \in L(S_{lo})$. In what follows, it can be shown that
$$B_c^{f_{com}} = B_c^{f'_{com}} \cup [(\bigcup_{\tau \in f'_{com} - f_{com}} B_u^\tau) - B_u^{f_{com}}]. \quad (11)$$

By the characterization of the controllable state set $B_c^{f'_{com}}$ and the controllability of $L(S_{lo})$, for an arbitrary controlled string $s \in L(S_{lo})$,
$$V(\delta(s, q_0)) \notin f'_{com} \Longrightarrow \delta(s, q_0) \in B_c^{f'_{com}}.$$
In other words, $V(\delta(s, q_0)) \in f'_{com}$ or $\delta(s, q_0) \in B_c^{f'_{com}}$. However, since vocal states for different $\tau \in T_c$ reside only in their respective B_u^τ, for $V(\delta(s, q_0)) \in f'_{com} - f_{com}$, $\delta(s, q_0) \in [(\bigcup_{\tau \in f'_{com} - f_{com}} B_u^\tau) - B_u^{f_{com}}]$. Together with (11), it follows that
$$V(\delta(s, q_0)) \in f_{com} \text{ or } \delta(s, q_0) \in B_c^{f_{com}}. \quad (12)$$
Since $B_c^{f_{com}}$ is a controllable set, by Theorem 1, $V(\delta(s, q_0)) \in f_{com}$ or $\delta(s, q_0) \in B_c^{f_{com}}$ implies $\delta(s\sigma, q_0) \notin B_u^{f_{com}}$ for all $\sigma \in \Sigma_u(\delta(s, q_0))$. Applying (12) and the fact that $Q = B_u^{f_{com}} \cup B_c^{f_{com}}$ with $B_u^{f_{com}} \cap B_c^{f_{com}} = \emptyset$, the result is that, for all $s \in L(S_{lo})$,
$$(\forall \sigma \in \Sigma_u(\delta(s, q_0)))\delta(s\sigma, q_0) \in B_c^{f_{com}}.$$
Now, together with the controllability of $L(S_{lo})$, it follows that for all $\sigma \in \Sigma_u$, for all $s \in L(S_{lo})$, where $\delta(s\sigma, q_0)!$,
$$\delta(s\sigma, q_0) \in B_c^{f_{com}} \text{ iff } s\sigma \in L(S_{lo}).$$
Next, it needs to be proved that, for all $\sigma \in \Sigma_c$, for all $s \in L(S_{lo})$, where $\delta(s\sigma, q_0)!$,
$$\begin{aligned}\delta(s\sigma, q_0) \in B_c^{f_{com}} &\text{ iff } \delta(s\sigma, q_0) \notin B_u^{f_{com}} \\ &\text{ iff } s\sigma \in L(S_{lo}),\end{aligned}$$
as follows:

- *(If)* $s\sigma \in L(S_{lo})$ and $\sigma \in \Sigma_c$. Assume $\delta(s\sigma, q_0) \in B_u^{f_{com}}$. This implies there is a $\tau \in f_{com} = Condat(S_{hi}, \theta(s))$, unambiguously controllable since G_{lo} is OCC, that can uncontrollably occur by vocalization, contradicting the fact that $\theta(s)\tau \notin L(S_{hi})$. Thus $\delta(s\sigma, q_0) \notin B_u^{f_{com}}$ or equivalently, $\delta(s\sigma, q_0) \in B_c^{f_{com}}$.
- *(Only If)* $s \in L(S_{lo})$, $\delta(s\sigma, q_0)!$, $\delta(s\sigma, q_0) \in B_c^{f_{com}}$ and $\sigma \in \Sigma_c$. There are two cases:
  - *Case 1*: $\theta(s\sigma) = \theta(s) \in L(S_{hi})$.
    Since $B_c^{f_{com}}$ is controllable, and the controllable $L(S_{lo})$ is supremally permissive w.r.t $\theta^{-1}(L(S_{hi}))$, $s\sigma \in L(S_{lo})$.
  - *Case 2*: $\theta(s\sigma) = \theta(s)\tau$ for some $\tau \notin f_{com}$, and therefore $\theta(s)\tau \in L(S_{hi})$.
    Assuming $s\sigma \notin L(S_{lo})$ contradicts the fact that $\theta(s)\tau \in L(S_{hi})$. Thus, $s\sigma \in L(S_{lo})$.

Hence for all $\sigma \in \Sigma$, for all $s \in L(S_{lo})$, where $\delta(s\sigma, q_0)!$,
$$(s \in L(S_{lo}) \& \delta(s\sigma, q_0) \in B_c^{f_{com}}) \text{ iff } s\sigma \in L(S_{lo}).$$
Assuming that $L(S_{hi}) \neq \emptyset$, $\varepsilon \in L(S_{hi})$ and therefore $\varepsilon \in L(S_{lo})$. Thus, with μ (9) as the permissive control law for f_{com} according to
$$\mu(f_{com}, s) = \{\sigma \in \Sigma_c \mid \delta(s\sigma, q_0)! \& \delta(s\sigma, q_0) \in B_u^{f_{com}}\}$$

such that $(\forall \sigma \in \Sigma_u(\delta(s,q_0)))\delta(s\sigma,q_0) \in B_c^{f_{com}}$, it follows by induction on the string $s \in L(S_{lo})$, beginning with $s := \varepsilon \in L(S_{lo})$, that, for all $\sigma \in \Sigma$,

$$s\sigma \in L(\mu, G_{lo}) \text{ iff } s\sigma \in L(S_{lo});$$

or that $L(\mu, G_{lo}) = L(S_{lo})$. Hence the theorem.

E. Computational Analysis for all B_u^τ, $\tau \in T_c$

Note that for an OCC DES G_{lo}, a non-vocal state that is in a set B_u^τ may not be in another set $B_u^{\tau'}$, for $\tau \in T_c$ and $\tau' \in T_c - \{\tau\}$. In what follows, the total time complexity analysis, of computing B_u^τ for all $\tau \in T_c$, is based on first determining the worst-case time complexity of constructing every B_u^τ, using Procedure `BuStateSet-Compute` below that does not exploit this fact. The result is a (well-defined) theoretical upper bound of the worst-case total time complexity.

Procedure `BuStateSet-Compute`

```
input : Moore DES $(G_{lo}, V)$ with vocal state set $Q_{voc} \subseteq Q$, and $\tau \in T_c$.
output: $B_u^\tau$.
begin
1    initialize
2        Initialize $B_u^\tau := \{q \in Q_{voc} \mid V(q) = \tau\}$;
3        Initialize $B := \emptyset$;
4    while $B_u^\tau \neq B$ do
5        $C := B_u^\tau - B$;
6        $B := B_u^\tau$;
7        foreach $q \in Q - (Q_{voc} \cup \{q_0\} \cup B_u^\tau)$ do
             foreach $\sigma \in \Sigma_u$ do
                 if $\delta(\sigma, q)!$ and $\delta(\sigma, q) \in C$ then
                     $B_u^\tau := B_u^\tau \cup \{q\}$;
8    return $B_u^\tau$;
end
```

To begin with, refer to Table I for the definitions of n, n_{voc}, u and v. Let $a = n - (n_{voc} + 1) + n_0^j$, which is the number of states in $[Q - (Q_{voc} \cup \{q_0\})] \cup \{q \in Q_{voc} \mid V(q) = \tau_{c_j} \in T_c\}$, where n_0^j is the number of vocal states outputting $\tau_{c_j} \in T_c$ under V. Then the iterative computations of Procedure `BuStateSet-Compute` will produce the following complexity series:

$$(a - n_0)n_0u, (a - n_1)(n_1 - n_0)u, (a - n_2)(n_2 - n_1)u, \cdots, (a - n_i)(n_i - n_{i-1})u \cdots, 0,$$

where $n_0 = n_0^j$, $n_{i+1} = n_i + k_{i+1}$, with n_{i+1} and n_i denoting the cardinality of B_u^τ immediately after and before each **forloop** iteration $i \geq 0$ (from line 7 of `BUStateSet-Compute`), respectively; $(a-n_i)(n_i-n_{i-1})u$, with $n_{-1} \overset{\text{def}}{=} 0$, is the time complexity incurred at iteration i, and $k_{i+1} \geq 0$ is the number of new non-vocal states added to the iterating B_u^τ following iteration i. The series stops at some iteration m with 0, where $n_m = n_{m-1}$ (i.e., no more non-vocal state was added at iteration $m - 1$).

Let $n_{q+} = n-(n_{voc}+1)$, which is the total number of non-vocal states in the DES, excluding the initial state q_0. Then the series can be rewritten as

$$(n_{q+})n_0^ju, (n_{q+} - k_1)k_1u, (n_{q+} - \sum_{j=1}^{2} k_j)k_2u, \cdots,$$
$$(n_{q+} - \sum_{j=1}^{i} k_j)k_iu \cdots, 0.$$

Computationally, the theoretical worst case occurs if the iterating B_u^τ increases by one new state per iteration until iteration $n_{q+} - 1$, at which $k_{n_{q+}} = 0$ (i.e., $k_{i+1} = 1$, for all $0 \leq i < n_{q+} - 1$). In other words, the worst case occurs if the complexity series is

$$(n_{q+})n_0^ju, (n_{q+} - 1)u, (n_{q+} - 2)u, (n_{q+} - 3)u, \cdots, (1)u, 0.$$

Reversing the series, the terms from 0 to $(n_{q+} - 1)u$ form an arithmetic series of n_{q+} terms with a difference of $1u$ between consecutive terms.

Summing the series and $(n_{q+})n_0^ju$ returns the result

$$(\frac{n_{q+}}{2}[n_{q+} - 1] + (n_{q+})n_0^j)u = 0.5un_{q+}(n_{q+} + 2n_0^j - 1).$$

Hence the worst-case complexity of Procedure `BuStateSet-Compute` is

$$O(un_{q+}^2 + (2n_0^j - 1)un_{q+})$$

or

$$O((2n_0^j - 1)u(n - n_{voc} - 1) + u(n - n_{voc} - 1)^2)$$

for each $\tau_{c_j} \in T_c$. Hence, the overall complexity for all $\tau_{c_j} \in T_c$ is

$$O(u\sum_{j=1}^{v}[(2n_0^j - 1)(n - n_{voc} - 1) + (n - n_{voc} - 1)^2])$$

or

$$O(u(2\alpha n_{voc} - v)(n - n_{voc} - 1) + vu(n - n_{voc} - 1)^2),$$

where $\alpha \leq 1$ and αn_{voc} refers to the total number of vocal states outputting events in T_c.

It follows that the worst-case total time complexity[7] occurs in the limit where $\alpha = 1$ (i.e., $T = T_c$), and is

$$O(u(2n_{voc} - v)(n - n_{voc} - 1) + vu(n - n_{voc} - 1)^2).$$

Acknowledgment

The authors would like to thank the Editor, the Associate Editor and all the anonymous referees for their constructive comments on the review versions of this paper. However, the authors remain solely responsible for the presented work.

References

[1] 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.

[2] 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.

[3] 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.

[4] B. Hrúz and M. Zhou, *Modeling and control of discrete-event dynamic systems: With petri nets and other tools*. Springer, 2007, vol. 59.

[5] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Springer, 2008.

[7]Based on the corrected definition of $B_u[q, V(q)]$, the revised worst-case complexity result is not significantly lower than that stated in the conference version [1].

[6] H. Hu, M. Zhou, Z. Li, and Y. Tang, "Deadlock-free control of automated manufacturing systems with flexible routes and assembly operations using petri nets," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 109–121, February 2013.

[7] J. S. Lee, M. Zhou, and P. L. Hsu, "A petri-net approach to modular supervision with conflict resolution for semiconductor manufacturing systems," *IEEE Transactions on Automation Science and Engineering*, vol. 4, no. 4, pp. 584–588, October 2007.

[8] R. J. Leduc, , M. Lawford, and P. Dai, "Hierarchical interface-based supervisory control of flexible manufacturing system," *IEEE Transactions on Control Systems Technology*, vol. 14, no. 4, pp. 654–668, July 2006.

[9] M. Z. Fekri and S. Hashtrudi-Zad, "Hierarchical supervisory control of discrete-event systems under partial observation," in *Proceedings of the 48th IEEE International Conference on Decision and Control*, Shanghai, China, December 2009, pp. 181 –186.

[10] 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.

[11] 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.

[12] O. Boutin, J. Komenda, T. Masopust, K. Schmidt, and J. H. van Schuppen, "Hierarchical control with partial observations: Sufficient conditions," in *Proceedings of the 51th IEEE International Conference on Decision and Control*, Orlando, FL, USA, December 2011, pp. 1817 – 1822.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] M. Z. Fekri and S. Hashtrudi-Zad, "Hierarchical robust supervisory control of discrete-event systems," in *Proceedings of the American Control Conference*, Seattle, Washington, USA, June 2008, pp. 1178–1183.

[18] A. Jayasiri, G. K. I. Mann, and R. G. Gosine, "Modular supervisory control and hierarchical supervisory control of fuzzy discrete-event systems," *IEEE Transactions on Automation Science and Engineering*, vol. 9, no. 2, pp. 353–364, April 2012.

[19] A. E. C. da Cunha, J. E. R. Cury, and B. H. Krogh, "An assume-guarantee reasoning for hierarchical coordination of discrete event systems," in *Proceedings of the 6th International Workshop on Discrete-Event Systems*, Zaragoza, Spain, October 2002, pp. 75–80.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] L. Feng and W. M. Wonham, "Supervisory control architecture for discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 53, no. 6, pp. 1449–1461, July 2008.

[25] C. Baier and T. Moor, "A hierarchical control architecture for sequential behaviours," in *Proceedings of the 11th International Workshop on Discrete-Event Systems*, Guadalajara, Mexico, October 2012, pp. 259–264.

[26] R. J. Leduc, B. A. Brandin, M. Lawford, and W. M. Wonham, "Hierarchical interface-based supervisory control - Part I: Serial case," *IEEE Transactions on Automatic Control*, vol. 50, no. 9, pp. 1322–1334, September 2005.

[27] R. J. Leduc, P. Dai, and R. Song, "Synthesis method for hierarchical interface-based supervisory control," *IEEE Transactions on Automatic Control*, vol. 54, no. 7, pp. 1548–1560, July 2009.

[28] R. C. Hill, J. E. R. Cury, M. H. de Queiroz, D. M. Tilbury, and S. Lafortune, "Multi-level hierarchical interface-based supervisory control," *Automatica*, vol. 46, no. 7, pp. 1152–1164, July 2010.

[29] R. Malik and R. J. Leduc, "Hierarchical interface-based supervisory control using the conflict preorder," in *Proceedings of the 11th International Workshop on Discrete-Event Systems*, Guadalajara, Mexico, October 2012, pp. 163–168.

[30] 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.

[31] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages and Computation*. Reading, MA : Addison-Wesley, 1979.

[32] S. Eilenberg, *Automata, Languages and Machines : Volume A*. Academic Press, New York, 1974.

[33] K. C. Wong, "On the complexity of projections of discrete-event systems," in *Proceedings of the 4th International Workshop on Discrete-Event Systems*. Cagliari, Italy: IEE Computing and Control Division, August 1998, pp. 201–206.

[34] K. C. Wong and W. M. Wonham, "On the computation of observers in discrete-event systems," *Discrete Event Dynamic Systems : Theory and Applications*, vol. 14, no. 1, pp. 55–107, January 2004.

[35] K. T. Seow, "Organizational control of discrete-event systems: A hierarchical multi-world supervisor design," *IEEE Transactions on Control Systems Technology*, Early On-line Access: March 2013.

[36] N. B. Hadj-Alouane, S. Lafortune, and F. Lin, "Variable lookahead supervisory control with state information," *IEEE Transactions on Automatic Control*, vol. 39, no. 12, pp. 2398–2410, December 1994.

Quang Ha Ngo received the B.Eng. (Hons) degree in computer engineering in 2008 from Nanyang Technological University (NTU), Singapore.

He is currently a Ph.D. candidate at the School of Computer Engineering, NTU. Prior to this, he was an associate consultant at Oracle Financial Services Software Pte Ltd, Singapore, from 2008 to 2010. His research interests include discrete-event systems and applications, data mining and big data analytics.

Kiam Tian Seow (SM'10) received the B.Eng. (Hons) degree in electrical engineering from the National University of Singapore, Singapore, in 1990 and the M.Eng. and Ph.D. degrees in electrical and computer engineering from Nanyang Technological University (NTU), Singapore, in 1993 and 1998, respectively.

In February 2003, he joined the School of Computer Engineering, NTU, where he has been a faculty member. He has held visiting research appointments with the Systems Control Group, University of Toronto, ON, Canada, in 1997; the Korea Advanced Institute of Science and Technology, Daejeon, Korea, in 2002; the Nippon Telegraph and Telephone Corporation (NTT) Communication Science Laboratories, Kyoto, Japan, in 2003; and the Institute of Information Science, Academia Sinica, Taipei, Taiwan, in 2005. His research interests include intelligent agents and multiagent systems, supervisory control of discrete-event systems and temporal logic, with emphasis on their mutual connections and applications.

Dr. Seow has been a member of Sigma Xi, the Scientific Research Honor Society since 2005. He has been an Associate Editor for the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING since 2009, and the IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS: SYSTEMS since 2013, and was an Associate Editor for the IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS-PART A: SYSTEMS AND HUMANS from 2010 to 2012.