Synthesis Method for Hierarchical Interface-Based Supervisory Control

Ryan. J. Leduc, Pengcheng Dai, and Raoguang Song

Abstract

Hierarchical Interface-Based Supervisory Control (HISC) decomposes a discrete-event system (DES) into a high-level subsystem which communicates with $n \geq 1$ low-level subsystems, through separate interfaces. It provides a set of local conditions that can be used to verify global conditions such as nonblocking and controllability such that the complete system model never needs to be constructed.

Currently, a designer must create the supervisors himself and then verify that they satisfy the HISC conditions. In this paper, we develop a synthesis method that can take advantage of the HISC structure. We replace the supervisor for each level by a corresponding specification DES. We then construct for each level a maximally permissive supervisor that satisfies the corresponding HISC conditions.

We define a set of language based fixpoint operators and show that they compute the required level-wise supremal languages. We then discuss the complexity of our algorithms and show that they potentially offer significant savings over the monolithic approach. We also briefly discuss a symbolic HISC verification and synthesis method using Binary Decision Diagrams, that we have also developed.

A large manufacturing system example (worst case state space on the order of 10^{30}) extended from the AIP example is briefly discussed. The example showed that we can now handle a given level with a statespace as large as 10^{15} states, using less than 160MB of memory. This represents a significant improvement in the size of systems that can be handled by the HISC approach. A software tool for synthesis and verification of HISC systems using our approach was also developed.

Index Terms

Discrete event systems, hierarchical systems, automata, interfaces, formal methods, synthesis.

R.J. Leduc, P. Dai, and R. Song are with the Dept. of Computing and Software, McMaster University, 1280 Main Street West, Hamilton, ON, Canada L8S 4K1. Email: leduc,daip,songr@mcmaster.ca

I. Introduction

In the area of Discrete-Event Systems (DES) [1]–[3], two common tasks are to verify that a composite system, based on a Cartesian product of subsystems, is (i) nonblocking and (ii) controllable. The main obstacle to performing these tasks is the combinatorial explosion of the product statespace.

The Hierarchical Interface-Based Supervisory Control (HISC) framework was proposed by Leduc et al. ([4]–[7]) to alleviate the state explosion problem. The HISC approach decomposes a system into a *high-level subsystem* which communicates with $n \geq 1$ parallel *low-level subsystems* through separate interfaces that restrict the interaction of the subsystems. It provides a set of local conditions that can be used to verify global conditions such as nonblocking and controllability. As each clause of the definition can be verified using a single subsystem, the complete system model never needs to be stored in memory, offering potentially significant savings in computational resources.

Currently, a designer must create the supervisors for a HISC system himself, and then verify that they satisfy the HISC conditions. If they do not, he must modify them until they do satisfy the conditions. For a complex system, it may be very non obvious how to achieve this. Also, the resulting supervisors may be more restrictive than they need to be. In this paper, we develop a synthesis method that can take advantage of the HISC structure. We replace the supervisor for each level by a corresponding specification DES. We then do a per level synthesis to construct for each level a maximally permissive supervisor that satisfies the corresponding HISC conditions. As the synthesis will be done on a per level basis, the complete system model never needs to be constructed. We thus expect to see similar savings in computation as in the HISC verification method. These savings should be even more pronounced as synthesis is an iterative process, thus typically requiring more computation.

However, both the HISC synthesis and verification method are currently based upon explicit state and transition listings which limit the size of a given level to about 10^7 states when 1GB of memory is used. To counter this problem, we have developed a set of symbolic, predicate based algorithms. We use Binary Decision Diagrams (BDD: [8], [9]) to represent the state space and transitions for each subsystem, and develop algorithms based on BDD representations to verify or synthesize supervisors. As we will see, by using BDD representation and algorithms we will be able to handle much larger subsystems, allowing HISC to be applied to even larger systems.

Using Integer Decision Diagrams (IDD, an extension of BDD) to verify and synthesize flat DES have previously been investigated by Zhang et al. [10], while Vahidi et al. [11] have investigated applying BDD to flat systems.

Ma et al. [12], [13] presented a top-down multi-level design model called State Tree Structures (STS), which was initiated in [14] by using the idea of state charts [15]. Ma used BDD based algorithms that allowed him to model and synthesize a state-based supervisor for a system with an estimated state-space on the order of 10^{24}.

There has also been some very promising recent work by Pena et al. [16], Flordal et al. [17], and Hill et al. [18] in using different forms of equivalence abstractions to verify flat systems in a modular fashion. These methods promise to increase the size of unstructured systems that we can verify. As individual levels (components) of an HISC system are treated as flat systems, these new results have the potential of being able to increase the individual component size that we can handle, if they can be adapted to verify the per component HISC conditions.

In this paper, we first discuss DES preliminaries, and then introduce the HISC approach in Section III. As the HISC method has already been explained and justified in detail in [5]–[7], we will only discuss it briefly here. For a small illustrative HISC example, please see [5].

For the remainder of the paper we will be presenting our new results from [19]–[22], beginning with an introduction to our HISC synthesis method. We will then define a set of language based fixpoint operators and show that they compute the required level-wise supremal languages. We will then show the equivalence between removing strings that fail the HISC conditions to removing DES states, as done in our algorithms.

We will then discuss the complexity of our algorithms and show that they potentially offer significant improvement over the monolithic approach. Next, we will very briefly discuss the symbolic methods we have developed. We close by discussing a large manufacturing system example (estimated worst case statespace on the order of 10^{30}) which was extended from the AIP example in [4], [7]. The example demonstrates the improved scalability that our symbolic approach offers.

II. DES PRELIMINARIES

Supervisory control theory provides a framework for the control of discrete-event systems (DES), systems that are discrete in space and time. For a detailed exposition of DES, see [3]. Below, we present a summary of the terminology that we use in this paper.

Let Σ be a finite set of distinct symbols (*events*), and Σ^* be the set of all finite sequences of events plus ϵ, the *empty string*. For strings $s, t \in \Sigma^*$, we say t is a *prefix* of s (written $t \leq s$) if $s = tu$, for some $u \in \Sigma^*$. We also say that t can be *extended* to s. We can now define the *extension operator* for language $L \subseteq \Sigma^*$.

Definition 1: For language L, we define the function $\text{Ext}_L : \text{Pwr}(\Sigma^*) \rightarrow \text{Pwr}(\Sigma^*)$, for arbitrary $K \in \text{Pwr}(\Sigma^*)$ as follows:

$$\text{Ext}_L(K) := \{t \in L |\, s \leq t \text{ for some } s \in K\}$$

In essence, $\text{Ext}_L(K)$ is the set of all strings in L that have prefixes in K. If we have $K \subseteq L$, we would then have $K \subseteq \text{Ext}_L(K)$ as $s \leq s$ for any given string.

The *prefix closure* of language L, denoted $\overline{L}$, is defined as $\overline{L} = \{t \in \Sigma^* \,|\, t \leq s \text{ for some } s \in L\}$. We say that L is *closed* if $L = \overline{L}$. Let $\text{Pwr}(\Sigma)$ denote the power set of Σ (i.e. all possible subsets of Σ). For language L, the eligibility operator $\text{Elig}_L : \Sigma^* \rightarrow \text{Pwr}(\Sigma)$ is given by $\text{Elig}_L(s) := \{\sigma \in \Sigma \,| s\sigma \in L\}$ for $s \in \Sigma^*$.

Let X be an arbitrary set. We say a function $f : X \rightarrow X$ is *monotone* if for all x, x' in X, $x \leq x' \Rightarrow f(x) \leq f(x')$. We say an element $x \in X$ is a *fixpoint* of f if $f(x) = x$. Furthermore, we say x is the *greatest fixpoint* of f if for all x' in X, $f(x') = x' \Rightarrow x' \leq x$. We will also use the notation $f^i(x)$, $i \in \{0, 1, 2, \ldots\}$, to mean i applications of f in a row with $f^0(x) := x$. i.e. $f^1(x) = f(x)$, $f^2(x) = f(f(x))$ and so on.

The *Nerode equivalence relation* over Σ^* with respect to L is defined for $s, t \in \Sigma^*$ as: $s \equiv_L t$ iff $(\forall u \in \Sigma^*) su \in L \Leftrightarrow tu \in L$. We say L is regular if relation $\equiv_L$ has a finite number of equivalence classes.

A DES automaton is represented as a 5-tuple $\mathbf{G} = (Y, \Sigma, \delta, y_o, Y_m)$ where Y is the state set, Σ is the event set, the partial function $\delta : Y \times \Sigma \rightarrow Y$ is the transition function, y_o is the initial state, and Y_m is the set of marker states. The function δ is extended to $\delta : Y \times \Sigma^* \rightarrow Y$ in the natural way. The notation $\delta(y, s)!$ means that δ is defined for $s \in \Sigma^*$ at state y. For DES $\mathbf{G}$, its closed behavior is denoted by $L(\mathbf{G})$, and is defined to be $L(\mathbf{G}) := \{s \in \Sigma^* |\; \delta(y_o, s)!\}$. The *marked behavior* of $\mathbf{G}$, is defined as $L_m(\mathbf{G}) := \{s \in L(G) |\ \delta(y_o, s) \in Y_m\}$.

The reachable state subset of DES $\mathbf{G}$, denoted Y_r, is: $Y_r := \{y \in Y |\ (\exists s \in \Sigma^*)\, \delta(y_o, s) = y\}$. A DES $\mathbf{G}$ is *reachable* if $Y_r = Y$. We will always assume that a DES has a finite state and event set, and is deterministic.

The *cross product* of DES $\mathbf{G}_i = (Y_i, \Sigma_i, \delta_i, y_{o_i}, Y_{m_i})$ $(i = 1, 2)$, is defined as $\mathbf{G_1} \times \mathbf{G_2} := (Q, \Sigma, \delta, q_o, Q_m)$, where $Q = Q_1 \times Q_2, \delta = \delta_1 \times \delta_2, q_o = (q_{o1}, q_{o2})$, and $Q_m = Q_{m1} \times Q_{m2}$, with $(\delta_1 \times \delta_2)((q_1, q_2), \sigma) := (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma))$, whenever $\delta_1(q_1, \sigma)!$ and $\delta_2(q_2, \sigma)!$.

Let $\Sigma = \Sigma_1 \cup \Sigma_2$, $L_1 \subseteq \Sigma_1^*$, and $L_2 \subseteq \Sigma_2^*$. For $i = 1, 2$, $s \in \Sigma^*$, and $\sigma \in \Sigma$, we define the *natural projection* $P_i : \Sigma^* \rightarrow \Sigma_i^*$ according to:

$$P_i(\epsilon) = \epsilon, \quad P_i(\sigma) = \begin{cases} \epsilon \text{ if } \sigma \notin \Sigma_i \\ \sigma \text{ if } \sigma \in \Sigma_i \end{cases}, \quad P_i(s\sigma) = P_i(s)P_i(\sigma)$$

The *synchronous product of languages* L_1 and L_2, denoted $L_1||L_2$, is defined to be:

$$L_1||L_2 = P_1^{-1}(L_1) \cap P_2^{-1}(L_2)$$

where $P_i^{-1} : \text{Pwr}(\Sigma_i^*) \rightarrow \text{Pwr}(\Sigma^*)$ is the inverse image function of P_i.

The *synchronous product of DES* $\mathbf{G}_i = (Y_i, \Sigma_i, \delta_i, y_{o_i}, Y_{m_i})$ $(i = 1, 2)$, denoted $\mathbf{G}_1||\mathbf{G}_2$, is defined to be a reachable DES $\mathbf{G}$ with event set $\Sigma = \Sigma_1 \cup \Sigma_2$ and properties:

$$L_m(\mathbf{G}) = L_m(\mathbf{G}_1)||L_m(\mathbf{G}_2), \quad L(\mathbf{G}) = L(\mathbf{G}_1)||L(\mathbf{G}_2)$$

For our purposes, we will assume that $\mathbf{G}_1||\mathbf{G}_2$ is implemented by first adding selfloops to all states of $\mathbf{G}_i$ $(i = 1, 2)$ for events in $\Sigma - \Sigma_i$, and then constructing the cross product of the two automata.

A DES $\mathbf{G}$ is *nonblocking* if $\overline{L_m(\mathbf{G})} = L(\mathbf{G})$. We say that DES $\mathbf{G}$ represents a language $K \subseteq \Sigma^*$ if $\mathbf{G}$ is nonblocking and $L_m(\mathbf{G}) = \mathbf{K}$. We thus have $L(\mathbf{G}) = \overline{\mathbf{K}}$.

For controllability, we assume the standard event partition $\Sigma = \Sigma_u \dot{\cup} \Sigma_c$, splitting our alphabet into *uncontrollable* and *controllable events*. To control a given plant $\mathbf{G}_1 = (Y_1, \Sigma_1, \delta_1, y_{o_1}, Y_{m_1})$, we define a *supervisor* represented as an automaton $\mathbf{S} = (X, \Sigma_S, \xi, x_o, X_m)$.

Definition 2: Let $\Sigma := \Sigma_1 \cup \Sigma_S, P_1 : \Sigma^* \rightarrow \Sigma_1^*$ and $P_S : \Sigma^* \rightarrow \Sigma_S^*$. Define $\mathcal{L}_{\mathbf{G}_1} := P_1^{-1}(L(\mathbf{G}_1))$ and $\mathcal{L}_{\mathbf{S}} := P_S^{-1}(L(\mathbf{S}))$. A supervisor $\mathbf{S}$ is *controllable* for a plant $\mathbf{G}_1$ if $\mathcal{L}_{\mathbf{S}}\Sigma_u \cap$

$\mathcal{L}_{\mathbf{G}_1} \subseteq \mathcal{L}_{\mathbf{S}}$ or, equivalently, $(\forall s \in \mathcal{L}_{\mathbf{G}_1} \cap \mathcal{L}_{\mathbf{S}})\ \mathrm{Elig}_{\mathcal{L}_{\mathbf{G}_1}}(s) \cap \Sigma_u \subseteq \mathrm{Elig}_{\mathcal{L}_{\mathbf{S}}}(s)$.

III. Hierarchical Interface Based Supervisory Control

A HISC system currently is a two-level system which includes one *high-level subsystem* and $n \geq 1$ *low-level subsystems*. The high-level subsystem communicates with each low-level subsystem through a separate *interface.*

In HISC there is a master-slave relationship. A *high-level subsystem* sends a command to a particular *low-level subsystem,* which then performs the indicated task and returns an answer. Fig. 1 shows conceptually the structure and information flow of the system. This style of interaction is enforced by an interface that mediates communication between the two subsystems.

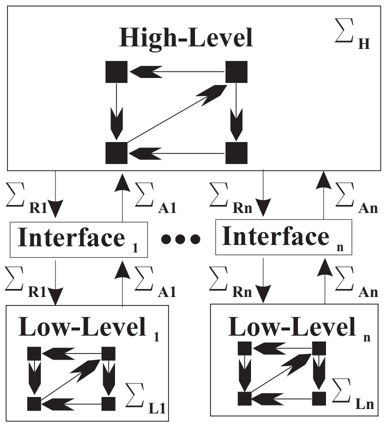

Fig. 1. Interface Block Diagram.

In order to restrict information flow and decouple the subsystems, the system alphabet is partitioned into pairwise disjoint alphabets

$$\Sigma := \Sigma_H \,\dot{\cup} \dot{\bigcup_{j=1,\ldots,n}} [\Sigma_{L_j} \dot{\cup} \Sigma_{R_j} \dot{\cup} \Sigma_{A_j}] \quad (1)$$

where we use "$\dot{\cup}$" to represent disjoint union.

The events in Σ_H are called *high-level events* and the events in Σ_{L_j} $(j = \{1, \ldots, n\})$ are *low-level events* as these events appear only in the high-level and low-level subsystem models, $\mathbf{G}_H$ and $\mathbf{G}_{L_j}$, respectively. We define our *flat system* to be $\mathbf{G} = \mathbf{G}_H || \mathbf{G}_{I_1} || \mathbf{G}_{L_1} || \ldots || \mathbf{G}_{I_n} || \mathbf{G}_{L_n}$. By flat system we mean the equivalent DES if we ignored the interface structure. For the remainder of this paper, the index j has range $\{1, \ldots, n\}$.

The j^{th} *interface*, $\mathbf{G}_{I_j}$, is defined over the events that are common to both levels of the hierarchy, $\Sigma_{R_j} \dot{\cup} \Sigma_{A_j}$. The events in Σ_{R_j}, called *request events*, represent commands sent from the high-level subsystem to the j^{th} low-level subsystem. The events in Σ_{A_j} are *answer events* and represent the low-level subsystem's responses to the request events. Request and answer events are collectively known as the set of *interface events*, defined as $\Sigma_I := \dot{\cup}_{k \in \{1,\ldots,n\}} [\Sigma_{R_k} \dot{\cup} \Sigma_{A_k}]$.

In order to enforce the serialization of requests and answers, we restrict the interfaces to the subclass of command-pair interfaces defined below. We present below a state based interface definition as it is more straightforward to verify. We show in [19] that it is equivalent to the language based definition given in [5]–[7] as long as $\mathbf{G}_{I_j}$ is reachable.

Definition 3: The j^{th} interface DES $\mathbf{G}_{I_j} = (X_j,\ \Sigma_{R_j} \dot\cup \Sigma_{A_j}, \xi_j, x_{o_j}, X_{m_j})$ is a *command-pair interface* if:

1) $x_{o_j} \in X_{m_j}$
2) $(\forall x \in X_{m_j})(\forall \sigma \in \Sigma_{R_j} \dot\cup \Sigma_{A_j})\, \xi_j(x,\sigma)! \Rightarrow \sigma \in \Sigma_{R_j} \wedge \xi_j(x,\sigma) \in X_j - X_{m_j}$
3) $(\forall x \in X_j - X_{m_j})(\forall \sigma \in \Sigma_{R_j} \dot\cup \Sigma_{A_j})\, \xi_j(x,\sigma)! \Rightarrow \sigma \in \Sigma_{A_j} \wedge \xi_j(x,\sigma) \in X_{m_j}$

To simplify notation in our exposition, we bring in the following event sets, natural projections, and languages.

$$\begin{array}{lll}
\Sigma_{I_j} := \Sigma_{R_j} \cup \Sigma_{A_j}, & P_{I_j} : \Sigma^* \to \Sigma^*_{I_j}, & \mathcal{H} := P_{IH}^{-1}(L(\mathbf{G}_H)), \quad \mathcal{H}_m := P_{IH}^{-1}(L_m(\mathbf{G}_H)) \\
\Sigma_{IL_j} := \Sigma_{L_j} \cup \Sigma_{I_j}, & P_{IL_j} : \Sigma^* \to \Sigma^*_{IL_j}, & \mathcal{L}_j := P_{IL_j}^{-1}(L(\mathbf{G}_{L_j})), \quad \mathcal{L}_{m_j} := P_{IL_j}^{-1}(L_m(\mathbf{G}_{L_j})) \\
\Sigma_{IH} := \Sigma_H \ \cup \bigcup_{k \in \{1,\ldots,n\}} \Sigma_{I_k} & P_{IH} : \Sigma^* \to \Sigma^*_{IH}, & \mathcal{I}_j := P_{I_j}^{-1}(L(\mathbf{G}_{I_j})), \quad \mathcal{I}_{m_j} := P_{I_j}^{-1}(L_m(\mathbf{G}_{I_j})) \\
\mathcal{I} := \cap_{k \in \{1,\ldots,n\}} \mathcal{I}_k, & \mathcal{I}_m := \cap_{k \in \{1,\ldots,n\}} \mathcal{I}_{m_k} &
\end{array}$$

We now present the properties that the system must satisfy to ensure that it interacts with the interfaces correctly. The point 5 in the definition below is actually slightly different than the one given in [4]–[7]. We use it since it makes the synthesis definitions clearer. We show in [19] that our new definition is equivalent.

Definition 4: The n^{th} degree $(n \geq 1)$ interface system composed of DES $\mathbf{G}_H, \mathbf{G}_{I_1}, \mathbf{G}_{L_1}, \ldots, \mathbf{G}_{I_n}, \mathbf{G}_{L_n}$, is *interface consistent* with respect to the alphabet partition given by (1), if for all $j \in \{1,\ldots,n\}$, the following conditions are satisfied:

Multi-level Properties

1) The event set of $\mathbf{G}_H$ is Σ_{IH}, and the event set of $\mathbf{G}_{L_j}$ is Σ_{IL_j}.
2) $\mathbf{G}_{I_j}$ is a command-pair interface.

High-Level Property

3) $(\forall s \in \mathcal{H} \cap \mathcal{I})\ \mathrm{Elig}_{\mathcal{I}_j}(s)\ \cap\ \Sigma_{A_j} \subseteq \mathrm{Elig}_{\mathcal{H}}(s)$

Low-Level Properties

4) $(\forall s \in \mathcal{L}_j \cap \mathcal{I}_j)\ \mathrm{Elig}_{\mathcal{I}_j}(s)\ \cap\ \Sigma_{R_j} \subseteq \mathrm{Elig}_{\mathcal{L}_j}(s)$

5) $(\forall s \in \mathcal{L}_j \cap \mathcal{I}_j)(\forall \rho \in \Sigma_{R_j})(\forall \alpha \in \Sigma_{A_j})\, s\rho\alpha \in \mathcal{I}_j \Rightarrow (\exists l \in \Sigma_{L_j}^*)\, s\rho l\alpha \in \mathcal{L}_j \cap \mathcal{I}_j$

6) $(\forall s \in \mathcal{L}_j \cap \mathcal{I}_j)\, s \in \mathcal{I}_{m_j} \Rightarrow (\exists l \in \Sigma_{L_j}^*)\, sl \in \mathcal{L}_{m_j} \cap \mathcal{I}_{m_j}$.

A. *Local Conditions for Global Nonblocking of the System*

We now provide the conditions that the subsystems and their interface(s) must satisfy in addition to the interface consistency properties, if the system $\mathbf{G}$ is to be nonblocking.

Definition 5: The n^{th} degree $(n \geq 1)$ interface system composed of DES $\mathbf{G}_H, \mathbf{G}_{I_1}, \mathbf{G}_{L_1}, \ldots, \mathbf{G}_{I_n}, \mathbf{G}_{L_n}$, is said to be *level-wise nonblocking* if the following conditions are satisfied:

(I) $\overline{\mathcal{H}_m \cap \mathcal{I}_m} = \mathcal{H} \cap \mathcal{I}$

(II) $(\forall j \in \{1, \ldots, n\})\ \overline{\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j}} = \mathcal{L}_j \cap \mathcal{I}_j$

Theorem 1 (from [7]): If the n^{th} degree $(n \geq 1)$ interface system composed of DES $\mathbf{G}_H, \mathbf{G}_{I_1}, \mathbf{G}_{L_1}, \ldots, \mathbf{G}_{I_n}, \mathbf{G}_{L_n}$, is level-wise nonblocking and interface consistent with respect to the alphabet partition given by (1), then $\overline{L_m(\mathbf{G})} = L(\mathbf{G})$, where $\mathbf{G} = \mathbf{G}_H || \mathbf{G}_{I_1} || \mathbf{G}_{L_1} || \ldots || \mathbf{G}_{I_n} || \mathbf{G}_{L_n}$.

B. *Local Conditions for Global Controllability of the System*

For controllability, we need to split the subsystems into their plant and supervisor components. To do this, we define the *high-level plant* to be $\mathbf{G}_H^p$, and the *high-level supervisor* to be $\mathbf{S}_H$ (both defined over event set Σ_{IH}). Similarly, the j^{th} *low-level plant* and *supervisor* are $\mathbf{G}_{L_j}^p$ and $\mathbf{S}_{L_j}$ (defined over Σ_{IL_j}). The high-level subsystem and the j^{th} low-level subsystem are then $\mathbf{G}_H := \mathbf{G}_H^p || \mathbf{S}_H$ and $\mathbf{G}_{L_j} := \mathbf{G}_{L_j}^p || \mathbf{S}_{L_j}$, respectively.

We can now define our *flat supervisor* and *plant* as well as some useful languages as follows:

$$\textbf{Plant} := \mathbf{G}_H^p || \mathbf{G}_{L_1}^p || \ldots || \mathbf{G}_{L_n}^p, \qquad \mathcal{H}^p := P_{IH}^{-1} L(\mathbf{G}_H^p), \quad \mathcal{S}_H := P_{IH}^{-1} L(\mathbf{S}_H)$$

$$\textbf{Sup} := \mathbf{S}_H || \mathbf{S}_{L_1} || \ldots || \mathbf{S}_{L_n} || \mathbf{G}_{I_1} || \ldots || \mathbf{G}_{I_n}, \quad \mathcal{L}_j^p := P_{IL_j}^{-1} L(\mathbf{G}_{L_j}^p), \quad \mathcal{S}_{L_j} := P_{IL_j}^{-1} L(\mathbf{S}_{L_j})$$

For the controllability requirements at each level, we adopt the standard partition $\Sigma = \Sigma_u \dot\cup \Sigma_c$, splitting our alphabet into *uncontrollable* and *controllable events.* Note that this partition may, in general, be independent of the partition given by (1). In fact, point 3 of Defn. 4 allows answer events to be controllable at the low-level, but forces high-level supervisors to treat them like uncontrollable events. Point 4 produces a similar result for request events and low-level supervisors.

Definition 6: The n^{th} degree $(n \geq 1)$ interface system composed of DES $\mathbf{G}^p_H$, $\mathbf{G}^p_{L_1}, \ldots,$ $\mathbf{G}^p_{L_n}$, $\mathbf{S}_H$, $\mathbf{S}_{L_1}, \ldots, \mathbf{S}_{L_n}$, $\mathbf{G}_{I_1}, \ldots, \mathbf{G}_{I_n}$, is *level-wise controllable* with respect to the alphabet partition given by (1), if for all $j \in \{1, \ldots, n\}$ the following conditions hold:

(I) The alphabet of $\mathbf{G}^p_H$ and $\mathbf{S}_H$ is Σ_{IH}, the alphabet of $\mathbf{G}^p_{L_j}$ and $\mathbf{S}_{L_j}$ is Σ_{IL_j}, and the alphabet of $\mathbf{G}_{I_j}$ is Σ_{I_j}

(II) $(\forall s \in \mathcal{L}^p_j \cap \mathcal{S}_{L_j} \cap \mathcal{I}_j)\ \text{Elig}_{\mathcal{L}^p_j}(s) \cap \Sigma_u \subseteq \text{Elig}_{\mathcal{S}_{L_j} \cap \mathcal{I}_j}(s)$

(III) $(\forall s \in \mathcal{H}^p \cap \mathcal{I} \cap \mathcal{S}_H)\ \text{Elig}_{\mathcal{H}^p \cap \mathcal{I}}(s) \cap \Sigma_u \subseteq \text{Elig}_{\mathcal{S}_H}(s)$

Theorem 2 (from [7]): If the n^{th} degree $(n \geq 1)$ interface system composed of plant components $\mathbf{G}^p_H$, $\mathbf{G}^p_{L_1}, \ldots, \mathbf{G}^p_{L_n}$, supervisors $\mathbf{S}_H$, $\mathbf{S}_{L_1}, \ldots, \mathbf{S}_{L_n}$, and interfaces $\mathbf{G}_{I_1}, \ldots, \mathbf{G}_{I_n}$, is level-wise controllable with respect to the alphabet partition given by (1), then

$$(\forall s \in L(\textbf{Plant}) \cap L(\textbf{Sup}))\ \text{Elig}_{L(\textbf{Plant})}(s) \cap \Sigma_u \subseteq \text{Elig}_{L(\textbf{Sup})}(s)$$

IV. HISC Synthesis Method

In Section III, we describe a system composed of plant DES $\mathbf{G}^p_H$, $\mathbf{G}^p_{L_1}, \ldots,$ $\mathbf{G}^p_{L_n}$, supervisor DES $\mathbf{S}_H$, $\mathbf{S}_{L_1}, \ldots, \mathbf{S}_{L_n}$, and interface DES $\mathbf{G}_{I_1}, \ldots, \mathbf{G}_{I_n}$ as a n^{th} degree interface system. When we specify a n^{th} degree interface system and give supervisors (as opposed to specifications), we will refer to such a system as a n^{th} *degree supervisor interface system.*

For this system, we showed how the properties of interface consistency, level-wise nonblocking, and level-wise controllable could be used to verify that the flat system is nonblocking, and that the flat supervisor is controllable for the flat plant. However, if the system fails one of these properties, we need a way to automatically modify the system to correct this. We need a synthesis method that can take advantage of the HISC structure and thus provide a similar level of scalability.

A. Synthesis Setting

For synthesis, we will replace supervisor $\mathbf{S}_H$ by specification DES $\mathbf{E}_H$ (defined over Σ_{IH}), and we will replace supervisor $\mathbf{S}_{L_j}$ by specification DES $\mathbf{E}_{L_j}$ (defined over Σ_{IL_j}). We thus get the system illustrated in Fig. 2. We will refer to such a system as a n^{th} *degree specification interface system.*

As a starting point for synthesis, we need to make sure that our specification interface system meets certain basic requirements. These are portions of the HISC conditions that we will not be able to correct for as part of our synthesis procedure.

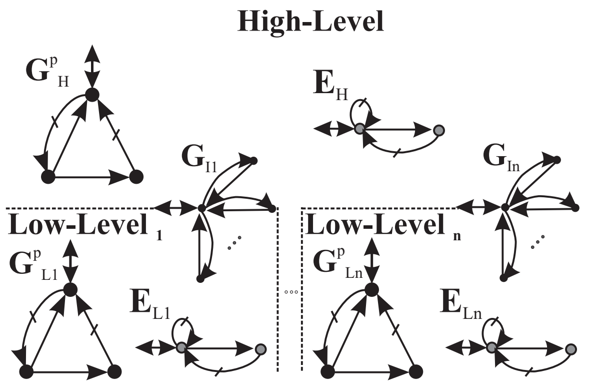

Fig. 2. Structure of System With Specifications.

Definition 7: The n^{th} degree specification interface system composed of plant DES $\mathbf{G}^p_H$, $\mathbf{G}^p_{L_1}, \ldots,$ $\mathbf{G}^p_{L_n}$, specification DES $\mathbf{E}_H$, $\mathbf{E}_{L_1}, \ldots, \mathbf{E}_{L_n}$, and interface DES $\mathbf{G}_{I_1}, \ldots, \mathbf{G}_{I_n}$ is *HISC-valid* with respect to the alphabet partition given by (1), if for all $j \in \{1, \ldots, n\}$, the following conditions are satisfied:

1) The event set of $\mathbf{G}^p_H$ and $\mathbf{E}_H$ is Σ_{IH}, and the event set of $\mathbf{G}^p_{L_j}$ and $\mathbf{E}_{L_j}$ is Σ_{IL_j}.
2) $\mathbf{G}_{I_j}$ is a command-pair interface.

For the rest of this section, we will use Φ to stand for the n^{th} degree HISC-valid specification interface system that respects the alphabet partition given by (1) and is composed of the plant DES $\mathbf{G}^p_H$, $\mathbf{G}^p_{L_1}, \ldots,$ $\mathbf{G}^p_{L_n}$, specification DES $\mathbf{E}_H$, $\mathbf{E}_{L_1}, \ldots, \mathbf{E}_{L_n}$, and interface DES $\mathbf{G}_{I_1}, \ldots, \mathbf{G}_{I_n}$, that we are considering.

In Section III, we introduced the languages $\mathcal{H}^p$, $\mathcal{L}^p_j$, $\mathcal{I}_j$, and $\mathcal{I}_{m_j}$. We now introduce a few more useful languages.

$$\mathcal{H}^p_m = P^{-1}_{IH}(L_m(\mathbf{G}^p_H)), \quad \mathcal{L}^p_{m_j} = P^{-1}_{IL_j}(L_m(\mathbf{G}^p_{L_j})), \quad \mathcal{E}_H = P^{-1}_{IH}(L(\mathbf{E}_H)), \quad \mathcal{E}_{H_m} = P^{-1}_{IH}(L_m(\mathbf{E}_H))$$

$$\mathcal{E}_{L_j} = P^{-1}_{IL_j}(L(\mathbf{E}_{L_j})), \quad \mathcal{E}_{L_{j,m}} = P^{-1}_{IL_j}(L_m(\mathbf{E}_{L_j}))$$

We will refer to the DES that represents the high-level of Φ as $\mathbf{G}_{\text{HL}} = \mathbf{G}^p_H || \mathbf{E}_H || \mathbf{G}_{I_1} || \ldots || \mathbf{G}_{I_n}$. We can now define the languages for $\mathbf{G}_{\text{HL}}$ over Σ^* as:

$$\mathcal{Z}_H := P^{-1}_{IH}(L(\mathbf{G}_{\text{HL}})) = \mathcal{H}^p \cap \mathcal{E}_H \cap \mathcal{I}, \qquad \mathcal{Z}_{H_m} := P^{-1}_{IH}(L_m(\mathbf{G}_{\text{HL}})) = \mathcal{H}^p_m \cap \mathcal{E}_{H_m} \cap \mathcal{I}_m$$

We will refer to the DES that represents the j^{th} low-level of Φ as $\mathbf{G}_{\text{LL}_j} = \mathbf{G}^p_{L_j} || \mathbf{E}_{L_j} || \mathbf{G}_{I_j}$.

We can now define the languages for $\mathbf{G}_{\mathrm{LL}_j}$ over Σ^* as:

$$\mathcal{Z}_{L_j} := P_{IL_j}^{-1}(L(\mathbf{G}_{\mathrm{LL}_j})) = \mathcal{L}_j^p \cap \mathcal{E}_{L_j} \cap \mathcal{I}_j, \qquad \mathcal{Z}_{L_{j,m}} := P_{IL_j}^{-1}(L_m(\mathbf{G}_{\mathrm{LL}_j})) = \mathcal{L}_{m_j}^p \cap \mathcal{E}_{L_{j,m}} \cap \mathcal{I}_{m_j}$$

B. High-Level Synthesis

We start by examining how, given system Φ, we can synthesize a supervisor for the high-level. Our first step is to capture the HISC properties that the supervisor's marked language must satisfy.

Definition 8: Let $Z \subseteq \Sigma^*$. For system Φ, language Z is *high-level interface controllable (HIC)* if for all $s \in \mathcal{H}^p \cap \mathcal{I} \cap \overline{Z}$, the following conditions are satisfied:

1) $\mathrm{Elig}_{\mathcal{H}^p \cap \mathcal{I}}(s) \cap \Sigma_u \subseteq \mathrm{Elig}_{\overline{Z}}(s)$
2) $(\forall j \in \{1, \dots, n\})\ \mathrm{Elig}_{\mathcal{I}_j}(s) \cap \Sigma_{A_j} \subseteq \mathrm{Elig}_{\mathcal{H}^p \cap \overline{Z}}(s)$

These conditions are essentially point III of Defn. 6 and point 3 of Defn. 4, where we have substituted $\overline{Z}$ for any reference of the high-level supervisor's closed behavior, $\mathcal{S}_H$, and we have used the identity $\mathbf{G}_H := \mathbf{G}_H^p || \mathbf{S}_H$ for the high-level subsystem.

For an arbitrary language $E \subseteq \Sigma^*$, we now define the set of all sublanguages of E that are HIC with respect to Φ as:

$$\mathcal{C}_H(E) := \{Z \subseteq E |\, Z \text{ is HIC with respect to } \Phi\}$$

We now show that a supremal element always exists.

Proposition 1: Let $E \subseteq \Sigma^*$. For system Φ, $\mathcal{C}_H(E)$ is nonempty and is closed under arbitrary union. In particular, $\mathcal{C}_H(E)$ contains a (unique) supremal element that we will denote $\sup\mathcal{C}_H(E)$.

Proof: Similar to the proof of Proposition 5. For specific details, refer to [19]. ■

We now note that if we take language $E = \mathcal{Z}_{H_m}$, we can conclude that $\sup\mathcal{C}_H(\mathcal{Z}_{H_m}) = \sup\mathcal{C}_H(\mathcal{H}_m^p \cap \mathcal{E}_{H_m} \cap \mathcal{I}_m)$ exists. As $\sup\mathcal{C}_H(\mathcal{Z}_{H_m}) \subseteq \mathcal{Z}_{H_m}$ by definition, it follows that $\sup\mathcal{C}_H(\mathcal{Z}_{H_m}) \cap \mathcal{Z}_{H_m} = \sup\mathcal{C}_H(\mathcal{Z}_{H_m})$. This implies that $\overline{\sup\mathcal{C}_H(\mathcal{Z}_{H_m})} \subseteq \mathcal{Z}_H$ as $\mathcal{Z}_{H_m} \subseteq \mathcal{Z}_H$, and $\mathcal{Z}_H$ is closed. It thus follows that if we take $\sup\mathcal{C}_H(\mathcal{Z}_{H_m})$ as the marked language of our high-level supervisor (assuming that $\sup\mathcal{C}_H(\mathcal{Z}_{H_m}) \neq \emptyset$), and $\overline{\sup\mathcal{C}_H(\mathcal{Z}_{H_m})}$ as the supervisor's closed behavior, then the supervisor will represent the closed loop-behavior of the high level. As this supervisor is clearly nonblocking, it follows that the high-level will be nonblocking, and thus point I of Defn. 5 will automatically be satisfied.

The reason that sup$\mathcal{C}_H(\mathcal{Z}_{H_m})$ allows us to construct the supremal HIC *nonblocking* supervisor follows from Defn. 8 where we refer to a language Z as being HIC, but the actual definition is in terms of $\overline{Z}$. We can think of $\mathcal{C}_H(E)$ as the set of marked sublanguages (Z) of language E, whose corresponding closed behavior ($\overline{Z}$) satisfies the terms of the HIC definition.

1) High-Level Fixpoint Operator: Now that we have shown that sup$\mathcal{C}_H(\mathcal{Z}_{H_m})$ exists, we need a means to construct it. We will do so by defining a fixpoint operator Ω_H, and show that our supremal element is the greatest fixpoint of the operator. To do this, we need to first define functions Ω_{HNB} and Ω_{HIC}.

Definition 9: For system Φ, we define the *high-level nonblocking operator*, $\Omega_{\text{HNB}} : \text{Pwr}(\Sigma^*) \rightarrow \text{Pwr}(\Sigma^*)$, for arbitrary $Z \in \text{Pwr}(\Sigma^*)$ as follows:

$$\Omega_{\text{HNB}}(Z) := Z \cap \mathcal{Z}_{H_m}$$

The way we will be using Ω_{HNB}, we would have $Z \subseteq \mathcal{Z}_H$ and closed, thus $\Omega_{\text{HNB}}(Z)$ would be the marked strings of the high-level that remain in Z. Clearly, operator Ω_{HNB} is monotone.

Definition 10: For system Φ, we define the *high-level interface controllable operator*, $\Omega_{\text{HIC}} : \text{Pwr}(\Sigma^*) \rightarrow \text{Pwr}(\Sigma^*)$, for arbitrary $Z \in \text{Pwr}(\Sigma^*)$ as follows:

$$\Omega_{\text{HIC}}(Z) := \overline{Z} - \text{Ext}_{\overline{Z}}(\text{FailHIC}(\overline{Z}))$$

where $\text{FailHIC}(\overline{Z}) := \{s \in \mathcal{H}^p \cap \mathcal{I} \cap \overline{Z} | \neg[\text{Elig}_{\mathcal{H}^p \cap \mathcal{I}}(s) \cap \Sigma_u \subseteq \text{Elig}_{\overline{Z}}(s)] \vee [(\exists j \in \{1, \ldots, n\}) \neg(\text{Elig}_{\mathcal{I}_j}(s) \cap \Sigma_{A_j} \subseteq \text{Elig}_{\mathcal{H}^p \cap \overline{Z}}(s))]\}$.

We first note that $\text{FailHIC}(\overline{Z}) \subseteq \overline{Z}$ and thus $\text{FailHIC}(\overline{Z}) \subseteq \text{Ext}_{\overline{Z}}(\text{FailHIC}(\overline{Z}))$, as $s \leq s$, for all $s \in \Sigma^*$. This operator takes our current estimate of our marked language, constructs its prefix closure($\overline{Z}$), then removes any strings (and their extensions) that would cause $\overline{Z}$ to fail the HIC definition. The reason we also remove the extensions, is so that $\Omega_{\text{HIC}}(Z)$ will always produce a prefix closed language.

We now show that operator Ω_{HIC} is monotone.

Proposition 2: For system Φ, the operator Ω_{HIC} satisfies:

$$(\forall Z, Z' \in \text{Pwr}(\Sigma^*))\ Z \subseteq Z' \Rightarrow \Omega_{\text{HIC}}(Z) \subseteq \Omega_{\text{HIC}}(Z')$$

Proof: Similar to the proof of Proposition 6. For specific details, refer to [19]. ■

We now combine these two operators to construct our fixpoint operator. As both Ω_{HNB} and Ω_{HIC} are monotone, it is easy to show that Ω_H is also monotone.

Definition 11: For system Φ, we define $\Omega_H : \text{Pwr}(\Sigma^*) \rightarrow \text{Pwr}(\Sigma^*)$, for arbitrary $Z \in \text{Pwr}(\Sigma^*)$, as follows:

$$\Omega_H(Z) := \Omega_{\text{HNB}}(\Omega_{\text{HIC}}(Z))$$

2) High-Level Propositions and Theorem: We next present two useful propositions before we give our main result for this section. Point 1 of the first proposition says that if $Z \subseteq Z'$, then applying Ω_H a finite number of times to each side will preserve the relationship. The second point states that all fixpoints are members of $\mathcal{C}_H(\mathcal{Z}_{H_m})$.

Proposition 3: Let $Z, Z' \subseteq \Sigma^*$ be arbitrary languages. For system Φ, the following properties are true:

1) $Z \subseteq Z' \Rightarrow (\forall i \in \{0,1,2,\ldots\})\, \Omega_H^i(Z) \subseteq \Omega_H^i(Z')$
2) $\Omega_H(Z) = Z \;\Rightarrow\; Z \in \mathcal{C}_H(\mathcal{Z}_{H_m})$

Proof: Similar to the proof of Proposition 7. For specific details, refer to [19]. ■

To prove the next proposition, we first have to show that sup$\mathcal{C}_H(\mathcal{Z}_{H_m})$ is a fixpoint. That it is the largest fixpoint then follows from applying point 2 of Proposition 3.

Proposition 4: For system Φ, sup$\mathcal{C}_H(\mathcal{Z}_{H_m})$ is the greatest fixpoint of Ω_H.

Proof: Similar to the proof of Proposition 8. For specific details, refer to [19]. ■

We will now show that if $\Omega_H(\mathcal{Z}_H)$ reaches a fixpoint after a finite number of steps, then that fixpoint is our supremal element. In section IV-E, we show that as long as the languages involved are regular, then a finite index always exists; thus our fixpoint operator can always produce the required supremal language in a finite number of steps.

Theorem 3: For system Φ, if there exists $i \in \{0,1,2,\ldots\}$ such that $\Omega_H^i(\mathcal{Z}_H)$ is a fixpoint, then $\Omega_H^i(\mathcal{Z}_H) = \text{sup}\mathcal{C}_H(\mathcal{Z}_{H_m})$.

Proof: Assume there exists $i \in \{0,1,2,\ldots\}$, such that $\Omega_H(\Omega_H^i(\mathcal{Z}_H)) = \Omega_H^i(\mathcal{Z}_H)$. **(1)**

We will now show $\Omega_H^i(\mathcal{Z}_H) = \text{sup}\mathcal{C}_H(\mathcal{Z}_{H_m})$.

We first note that we have: $\text{sup}\mathcal{C}_H(\mathcal{Z}_{H_m}) \subseteq \mathcal{Z}_{H_m} \subseteq \mathcal{Z}_H$

This allows us to apply point 1 of Proposition 3 and conclude:

$\Omega_H^i(\text{sup}\mathcal{C}_H(\mathcal{Z}_{H_m})) \subseteq \Omega_H^i(\mathcal{Z}_H)$ **(3)**

By Proposition 4, we know that sup$\mathcal{C}_H(\mathcal{Z}_{H_m})$ is the greatest fixpoint of Ω_H. **(4)**

$\Rightarrow \text{sup}\mathcal{C}_H(\mathcal{Z}_{H_m}) \subseteq \Omega^i_H(\mathcal{Z}_H)$ as $\Omega^i_H(\text{sup}\mathcal{C}_H(\mathcal{Z}_{H_m})) = \text{sup}\mathcal{C}_H(\mathcal{Z}_{H_m})$. **(5)**

As $\Omega^i_H(\mathcal{Z}_H)$ is a fixpoint, we have by **(4)**: $\Omega^i_H(\mathcal{Z}_H) \subseteq \text{sup}\mathcal{C}_H(\mathcal{Z}_{H_m})$

By **(5)**, we thus have $\Omega^i_H(\mathcal{Z}_H) = \text{sup}\mathcal{C}_H(\mathcal{Z}_{H_m})$ as required. ■

We now show that we can use $\text{sup}\mathcal{C}_H(\mathcal{Z}_{H_m})$ to define our high-level supervisor and satisfy the relevant interface conditions. We will use S_{H_m} to stand for the marked language of the high-level supervisor in the corollary below.

Corollary 1: For system Φ, if there exists $i \in \{0, 1, 2, \ldots\}$ such that $\Omega^i_H(\mathcal{Z}_H)$ is a fixpoint, then system Φ with $S_{H_m} = \Omega^i_H(\mathcal{Z}_H)$ and $S_H = \overline{S_{H_m}}$ satisfies point 3 of Defn. 4, point I of Defn. 5 and point III of Defn. 6.

Proof: Similar to the proof of Corollary 2. For specific details, refer to [19]. ■

C. Low-Level Synthesis

We now examine how, given system Φ, we can synthesize a supervisor for the j^{th} low-level. Our first step is to capture the HISC properties that the supervisor's marked language must satisfy.

Definition 12: Let $Z \subseteq \Sigma^*$. For system Φ, language Z is j^{th} *low-level interface controllable (LICj)* if for all $s \in \mathcal{L}^p_j \cap \mathcal{I}_j \cap \overline{Z}$, the following conditions are satisfied:

1) $\text{Elig}_{\mathcal{L}^p_j}(s) \cap \Sigma_u \subseteq \text{Elig}_{\overline{Z} \cap \mathcal{I}_j}(s)$
2) $\text{Elig}_{\mathcal{I}_j}(s) \cap \Sigma_{R_j} \subseteq \text{Elig}_{\mathcal{L}^p_j \cap \overline{Z}}(s)$
3) $(\forall \rho \in \Sigma_{R_j})(\forall \alpha \in \Sigma_{A_j})\ s\rho\alpha \in \mathcal{I}_j \Rightarrow (\exists l \in \Sigma^*_{L_j})\ s\rho l\alpha \in \mathcal{L}^p_j \cap \overline{Z} \cap \mathcal{I}_j$
4) $s \in \mathcal{I}_{m_j} \Rightarrow (\exists l \in \Sigma^*_{L_j})\ sl \in \mathcal{L}^p_{m_j} \cap Z \cap \mathcal{I}_{m_j}$

These conditions are essentially point II of Defn. 6, and points 4-6 of Defn. 4, where we have substituted $\overline{Z}$ for any reference of the j^{th} low-level supervisor's closed behavior ($\mathcal{S}_{L_j}$), Z for any reference of the supervisor's marked language, and we have used the identity $\mathbf{G}_{L_j} := \mathbf{G}^p_{L_j} || \mathbf{S}_{L_j}$ for the j^{th} low-level subsystem.

For an arbitrary language $E \subseteq \Sigma^*$, we now define the set of all sublanguages of E that are LICj with respect to Φ as

$$\mathcal{C}_{L_j}(E) := \{Z \subseteq E |\ Z \text{ is LICj with respect to } \Phi\}.$$

Proposition 5: Let $E \subseteq \Sigma^*$. For system Φ, $\mathcal{C}_{L_j}(E)$ is nonempty and is closed under arbitrary union. In particular, $\mathcal{C}_{L_j}(E)$ contains a (unique) supremal element that we will denote sup$\mathcal{C}_{L_j}(E)$.

Proof: See Appendix. ■

If sup$\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}}) \neq \emptyset$, we can take sup$\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})$ as the marked language of our j^{th} low-level supervisor and $\overline{\text{sup}\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})}$ as the supervisor's closed behavior, and it will follow that the low-level is nonblocking (i.e. point II of Defn. 5 is satisfied, for this j).

D. The j^{th} Low-Level Fixpoint Operator

We will now define a fixpoint operator Ω_{L_j}, to construct sup$\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})$. We need to first define functions Ω_{LNB_j} and Ω_{LIC_j}.

Definition 13: For system Φ, we define the j^{th} *low-level nonblocking operator*, Ω_{LNB_j} : Pwr$(\Sigma^*) \rightarrow$ Pwr(Σ^*), for arbitrary $Z \in$ Pwr(Σ^*) as follows:

$$\Omega_{\text{LNB}_j}(Z) := Z \cap \mathcal{Z}_{L_{j,m}}$$

Operator Ω_{LNB_j} is analogous to the high-level operator Ω_{HNB}, and is clearly monotone.

Definition 14: For system Φ, we define the j^{th} *low-level interface controllable operator*, Ω_{LIC_j} : Pwr$(\Sigma^*) \rightarrow$ Pwr(Σ^*), for arbitrary $Z \in$ Pwr(Σ^*) as follows:

$$\Omega_{\text{LIC}_j}(Z) := \overline{Z} - \text{Ext}_{\overline{Z}}(\text{FailLIC}_j(\overline{Z}))$$

where $\text{FailLIC}_j(\overline{Z}) := \{s \in \mathcal{L}^p_j \cap \mathcal{I}_j \cap \overline{Z} \mid \neg[\text{Elig}_{\mathcal{L}^p_j}(s) \cap \Sigma_u \subseteq \text{Elig}_{\overline{Z} \cap \mathcal{I}_j}(s)] \vee \neg[\text{Elig}_{\mathcal{I}_j}(s) \cap \Sigma_{R_j} \subseteq \text{Elig}_{\mathcal{L}^p_j \cap \overline{Z}}(s)] \vee \neg[(\forall \rho \in \Sigma_{R_j})(\forall \alpha \in \Sigma_{A_j})\, s\rho\alpha \in \mathcal{I}_j \Rightarrow (\exists l \in \Sigma^*_{L_j})\, s\rho l\alpha \in \mathcal{L}^p_j \cap \overline{Z} \cap \mathcal{I}_j] \vee \neg[s \in \mathcal{I}_{m_j} \Rightarrow (\exists l \in \Sigma^*_{L_j})\ sl \in \mathcal{L}^p_{m_j} \cap Z \cap \mathcal{I}_{m_j}]\}$.

Operator Ω_{LIC_j} removes from $\overline{Z}$ any string that has a prefix that would cause $\overline{Z}$ to fail the LICj definition. We now show that it is monotone.

Proposition 6: For system Φ, the operator Ω_{LIC_j} satisfies:

$$(\forall Z, Z' \in \text{Pwr}(\Sigma^*))\, Z \subseteq Z' \Rightarrow \Omega_{\text{LIC}_j}(Z) \subseteq \Omega_{\text{LIC}_j}(Z')$$

Proof: See Appendix. ■

We now combine these two operators to construct our fixpoint operator. As both Ω_{LNB_j} and Ω_{LIC_j} are monotone, it is easy to show that Ω_{L_j} is also monotone.

Definition 15: For system Φ, we define $\Omega_{L_j} : \mathrm{Pwr}(\Sigma^*) \rightarrow \mathrm{Pwr}(\Sigma^*)$, for arbitrary $Z \in \mathrm{Pwr}(\Sigma^*)$, as follows:

$$\Omega_{L_j}(Z) := \Omega_{\mathrm{LNB}_j}(\Omega_{\mathrm{LIC}_j}(Z))$$

1) Low-Level Propositions and Theorem: We next present two useful propositions before we give our main result for this section. Point 1 of the first proposition says that if $Z \subseteq Z'$, then applying Ω_{L_j} a finite number of times to each side will preserve the relationship. The second point states that all fixpoints are members of $\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})$.

Proposition 7: Let $Z, Z' \subseteq \Sigma^*$ be arbitrary languages. For system Φ, the following properties are true:

1) $Z \subseteq Z' \Rightarrow (\forall i \in \{0, 1, 2, \ldots\})\, \Omega^i_{L_j}(Z) \subseteq \Omega^i_{L_j}(Z')$
2) $\Omega_{L_j}(Z) = Z \;\Rightarrow\; Z \in \mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})$

Proof: See Appendix. ■

To prove the next proposition, we first have to show that sup$\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})$ is a fixpoint. That it is the largest fixpoint then follows from applying point 2 of Proposition 7.

Proposition 8: For system Φ, sup$\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})$ is the greatest fixpoint of Ω_{L_j}.

Proof: See Appendix. ■

We will now show that if $\Omega_{L_j}(\mathcal{Z}_{L_j})$ reaches a fixpoint after a finite number of steps, then that fixpoint is our supremal element. In section IV-E, we show that as long as the languages involved are regular, then a finite index always exists; thus our fixpoint operator can always produce the required supremal language in a finite number of steps.

Theorem 4: For system Φ, if there exists $i \in \{0, 1, 2, \ldots\}$ such that $\Omega^i_{L_j}(\mathcal{Z}_{L_j})$ is a fixpoint, then $\Omega^i_{L_j}(\mathcal{Z}_{L_j}) =$ sup$\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})$.

Proof: Similar to the proof of *Theorem 3*. For specific details, refer to [19]. ■

We now show that we can use sup$\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})$ to define our j^{th} low-level supervisor and satisfy the relevant interface conditions. We will use $S_{L_{j,m}}$ to stand for the marked language of the j^{th} low-level supervisor in the corollary below.

Corollary 2: For system Φ, if there exists $i \in \{0, 1, 2, \ldots\}$ such that $\Omega^i_{L_j}(\mathcal{Z}_{L_j})$ is a fixpoint, then system Φ with $S_{L_{j,m}} = \Omega^i_{L_j}(\mathcal{Z}_{L_j})$ and $S_{L_j} = \overline{S_{L_{j,m}}}$ satisfies points 4–6 of Defn. 4, point II of Defn. 5, and point II of Defn. 6, for the given index j.

Proof: Assume there exists $i \in \{0, 1, 2, \ldots\}$, such that $\Omega_{L_j}(\Omega^i_{L_j}(\mathcal{Z}_{L_j})) = \Omega^i_{L_j}(\mathcal{Z}_{L_j})$. **(1)**

Let $S_{L_{j,m}} = \Omega^i_{L_j}(\mathcal{Z}_{L_j})$ and $S_{L_j} = \overline{S_{L_{j,m}}}$.

By Theorem 4, $S_{L_{j,m}} = \sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})$ is LIC_j with respect to Φ. **(2)**

By Defn. 12 and using the fact that $S_{L_j} = \overline{S_{L_{j,m}}}$, we have for all $s \in \mathcal{L}^p_j \cap \mathcal{I}_j \cap S_{L_j}$

1) $\text{Elig}_{\mathcal{L}^p_j}(s) \cap \Sigma_u \subseteq \text{Elig}_{S_{L_j} \cap \mathcal{I}_j}(s)$ **(3)**
2) $\text{Elig}_{\mathcal{I}_j}(s) \cap \Sigma_{R_j} \subseteq \text{Elig}_{\mathcal{L}^p_j \cap S_{L_j}}(s)$ **(4)**
3) $(\forall \rho \in \Sigma_{R_j})(\forall \alpha \in \Sigma_{A_j})\, s\rho\alpha \in \mathcal{I}_j \Rightarrow (\exists l \in \Sigma^*_{L_j})\, s\rho l\alpha \in \mathcal{L}^p_j \cap S_{L_j} \cap \mathcal{I}_j$ **(5)**
4) $s \in \mathcal{I}_{m_j} \Rightarrow (\exists l \in \Sigma^*_{L_j})\, sl \in \mathcal{L}^p_{m_j} \cap S_{L_{j,m}} \cap \mathcal{I}_{m_j}$ **(6)**

We note that point II of Defn. 6 follows from **(3)**.

Using facts $\mathcal{L}_j = \mathcal{L}^p_j \cap S_{L_j}$, and $\mathcal{L}_{m_j} = \mathcal{L}^p_{m_j} \cap S_{L_{j,m}}$, we see points 4, 5, and 6 of Defn. 4 follow from **(4)-(6)**.

Now must show that point I of Defn. 5 is satisfied.

Sufficient to show $\overline{\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j}} = \mathcal{L}_j \cap \mathcal{I}_j$

By **(2)**, we have $S_{L_{j,m}} = \sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}}) \subseteq \mathcal{Z}_{L_{j,m}} \subseteq \mathcal{Z}_{L_j}$

$\Rightarrow S_{L_j} = \overline{S_{L_{j,m}}} \subseteq \mathcal{Z}_{L_j}$, as $\mathcal{Z}_{L_j}$ is closed.

Substituting for $\mathcal{Z}_{L_{j,m}}$ and $\mathcal{Z}_{L_j}$, gives $S_{L_{j,m}} \subseteq \mathcal{L}^p_{m_j} \cap \mathcal{E}_{L_{j,m}} \cap \mathcal{I}_{m_j}$ and $S_{L_j} \subseteq \mathcal{L}^p_j \cap \mathcal{E}_{L_j} \cap \mathcal{I}_j$

$\Rightarrow \mathcal{L}_{m_j} \cap \mathcal{I}_{m_j} = \mathcal{L}^p_{m_j} \cap S_{L_{j,m}} \cap \mathcal{I}_{m_j} = S_{L_{j,m}}$ and $\mathcal{L}_j \cap \mathcal{I}_j = \mathcal{L}^p_j \cap S_{L_j} \cap \mathcal{I}_j = S_{L_j}$

As $S_{L_j} = \overline{S_{L_{j,m}}}$ by definition, we have $\overline{\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j}} = \mathcal{L}_j \cap \mathcal{I}_j$, as required. ■

Combining Corollaries 1 and 2, we see that our fixpoint operators allow us to construct supervisors for the high-level and the n low-levels that by design, will make system Φ, with its specification DES replaced by these supervisors (referred to as the *augmented system*), be interface consistent, level-wise nonblocking and level-wise controllable.

E. String and State Equivalence

In Sections IV-B and IV-C, we presented a set of language-based fixpoint operators to construct our supremal languages. The algorithms we have developed in [19], [21] construct DES that represent these supremal languages, but they operate by removing states instead of strings. In this section, we will show the equivalence of the two approaches, and that our method can always construct the required supremal languages in a finite amount of time.

The first step in the synthesis process is the construction of the synchronous product of all the DES in the system. Let DES $\mathbf{G_i} = (Q_i, \Sigma_i, \delta_i, q_{oi}, Q_{mi})$, $\Sigma = \bigcup_{j=1,\dots,k} \Sigma_j$, $P_i : \Sigma^* \to \Sigma_i^*$, with $i = 1, 2, \dots, k$. Define $\mathbf{G} = \mathbf{G_1} \| \mathbf{G_2} \| ... \| \mathbf{G_k} = (Q, \Sigma, \delta, q_o, Q_m)$, $\mathcal{L}_i := P_i^{-1} L(\mathbf{G_i})$, and $\mathcal{L}_{m,i} := P_i^{-1} L_m(\mathbf{G_i})$. The proposition below states that for any two strings that go to the same state in $\mathbf{G}$, then these two strings also lead to the same state in each of the component DES, and thus they belong to the same Nerode equivalent classes of the component DES's closed and marked languages. The key here is that the synchronous product has not been minimized, or we could lose this one-to-one relationship between a state in $\mathbf{G}$ and a state in $\mathbf{G}_i$.

Proposition 9: Let $\mathbf{G_i}$ ($i = 1, 2, \dots, k$), $\mathbf{G}$, $\mathcal{L}_i$, and $\mathcal{L}_{m,i}$ be defined as above. It then follows:

$$(\forall i \in \{1, 2, \dots, k\})(\forall s, t \in L(\mathbf{G}))\ \delta(q_o, s) = \delta(q_o, t) \ \Rightarrow\ s \equiv_{\mathcal{L}_i} t \ \wedge\ s \equiv_{\mathcal{L}_{m,i}} t$$

Proof: See [19] for an example of how to prove this result. ■

As we will see, this relationship is important as it will ensure that if a string that takes us to a state in $\mathbf{G}$ fails one of our HISC conditions, then we can show that all strings that lead to this state also fail; thus we can simply remove the state. This allows our algorithm to operate on a finite number of states, instead of a possibly infinite number of strings.

For convenience, we first discuss a well known nonblocking result. If a DES $\mathbf{G}$ blocks, we would have $L(\mathbf{G}) \not\subseteq \overline{L_m(\mathbf{G})}$. The proposition below states that if a string can't be extended to a marked string, then all strings that lead to the same state in $\mathbf{G}$ can't either; thus we need to remove the state. We also note that removing a state not only removes all strings that reach this state, but also removes all defined strings that can leave this state. In other words, if we remove state $q \in Q$ of DES $\mathbf{G} = (Q, \Sigma, \delta, q_o, Q_m)$, we remove from $L(\mathbf{G})$ the strings $L_q := \{s \in L(\mathbf{G}) | \, \delta(q_o, s) = q\}$ as well as all strings that have prefixes in L_q (i.e. $\mathrm{Ext}_{L(\mathbf{G})}(L_q)$). This is consistent with how we defined our language based fixpoint operators in Sections IV-B and IV-C.

Proposition 10: For $\mathbf{G} = (Q, \Sigma, \delta, q_o, Q_m)$, it follows:

$$(\forall s, t \in L(\mathbf{G}))\ [\delta(q_o, s) = \delta(q_o, t)] \Rightarrow s \notin \overline{L_m(\mathbf{G})} \Leftrightarrow t \notin \overline{L_m(\mathbf{G})}$$

Proof: See [19] for an example of how to prove this result. ■

We will now use Proposition 9 to prove some useful results related to the HISC conditions. We start with a well known controllability result. We first introduce some notation that we will need. Let $I_1 \subseteq \{1, 2, \ldots k\}$ and $I_2 \subseteq \{1, 2, \ldots k\}$ be nonempty index sets for our k DES. Define $\mathcal{L}_{I_1} = \cap_{v \in I_1} \mathcal{L}_v$ and $\mathcal{L}_{I_2} = \cap_{v \in I_2} \mathcal{L}_v$. The proposition basically says for a given state in $\mathbf{G} := \mathbf{G_1} \| \mathbf{G_2} \| ... \| \mathbf{G_k} = (Q, \Sigma, \delta, q_o, Q_m)$, either all strings leading to the state pass the condition or they all fail and we can remove the state.

Proposition 11: Let $\Sigma_a \subseteq \Sigma$, I_1 and I_2 be nonempty index sets for our k DES, and $\mathbf{G}$, $\mathcal{L}_{I_1}$, and $\mathcal{L}_{I_2}$ be defined as above. It thus follows that for all $s, t \in L(\mathbf{G})$, if $\delta(q_o, s) = \delta(q_o, t)$ then

$$\text{Elig}_{\mathcal{L}_{I_1}}(s) \cap \Sigma_a \not\subseteq \text{Elig}_{\mathcal{L}_{I_2}}(s) \;\Leftrightarrow\; \text{Elig}_{\mathcal{L}_{I_1}}(t) \cap \Sigma_a \not\subseteq \text{Elig}_{\mathcal{L}_{I_2}}(t)$$

Proof: See [19] for an example of how to prove this result. ■

We note that if we choose I_1, I_2, and Σ_a appropriately, then Proposition 11 can be applied to the level-wise controllability definition, as well as points 3 and 4 of the interface consistency definition.

We now present a proposition relevant to the HIC definition. We first need to define DES $\mathbf{G}'_{\text{HL}}$ to be DES $\mathbf{G}_{\text{HL}}$ (see definition in Section IV-A) with events $\Sigma - \Sigma_{IH}$ selflooped at every state. This is to extend the event set of $\mathbf{G}_{\text{HL}}$ to Σ, so that it will be compatible with the languages used in the HIC definition (i.e. $L(\mathbf{G}'_{\text{HL}}) = \mathcal{Z}_H$). Essentially, the proposition states that if a string fails one of the clauses in the HIC definition, then all strings that lead to the same state in $\mathbf{G}'_{\text{HL}}$ will also fail, thus we need to remove the state.

Proposition 12: For system Φ, let $\mathbf{G}'_{\text{HL}} = (Q_H, \Sigma, \delta_H, q_{o,H}, Q_{m,H})$. It follows that for all $s, t \in L(\mathbf{G}'_{\text{HL}})$, if $\delta_H(q_o, s) = \delta_H(q_o, t)$ then

1) $\text{Elig}_{\mathcal{H}^p \cap \mathcal{I}}(s) \cap \Sigma_u \not\subseteq \text{Elig}_{\mathcal{Z}_H}(s) \Leftrightarrow \text{Elig}_{\mathcal{H}^p \cap \mathcal{I}}(t) \cap \Sigma_u \not\subseteq \text{Elig}_{\mathcal{Z}_H}(t)$
2) $(\forall j \in \{1, \ldots, n\})\ \text{Elig}_{\mathcal{I}_j}(s) \cap \Sigma_{A_j} \not\subseteq \text{Elig}_{\mathcal{H}^p \cap \mathcal{Z}_H}(s) \Leftrightarrow \text{Elig}_{\mathcal{I}_j}(t) \cap \Sigma_{A_j} \not\subseteq \text{Elig}_{\mathcal{H}^p \cap \mathcal{Z}_H}(t)$

Proof: Follows easily from Proposition 11. ■

We next present a proposition relevant to the LICj definition. Again, we first need to define DES $\mathbf{G}'_{\text{LL}_j}$ to be DES $\mathbf{G}_{\text{LL}_j}$ (see definition in Section IV-A) with events $\Sigma - \Sigma_{IL_j}$ selflooped at every state, thus $L(\mathbf{G}'_{\text{LL}_j}) = \mathcal{Z}_{L_j}$. Essentially, the proposition states that if a string fails one of the clauses in the LICj definition, then all strings that lead to the same state in $\mathbf{G}'_{\text{LL}_j}$ will also fail, thus we need to remove the state.

Proposition 13: For system Φ, let $\mathbf{G}'_{\text{LL}_j} = (Q_{L_j}, \Sigma, \delta_{L_j}, q_{o,L_j}, Q_{m,L_j})$. It follows that for all $s, t \in L(\mathbf{G}'_{\text{LL}_j})$, if $\delta_{L_j}(q_{o,L_j}, s) = \delta_{L_j}(q_{o,L_j}, t)$ then

1) $\text{Elig}_{\mathcal{L}_j^p}(s) \cap \Sigma_u \not\subseteq \text{Elig}_{\mathcal{Z}_{L_j} \cap \mathcal{I}_j}(s) \Leftrightarrow \text{Elig}_{\mathcal{L}_j^p}(t) \cap \Sigma_u \not\subseteq \text{Elig}_{\mathcal{Z}_{L_j} \cap \mathcal{I}_j}(t)$

2) $\mathrm{Elig}_{\mathcal{I}_j}(s) \cap \Sigma_{R_j} \not\subseteq \mathrm{Elig}_{\mathcal{L}_j^p \cap \mathcal{Z}_{L_j}}(s) \Leftrightarrow \mathrm{Elig}_{\mathcal{I}_j}(t) \cap \Sigma_{R_j} \not\subseteq \mathrm{Elig}_{\mathcal{L}_j^p \cap \mathcal{Z}_{L_j}}(t)$
3) $(\forall \rho \in \Sigma_{R_j})(\forall \alpha \in \Sigma_{A_j})\ ([s\rho\alpha \in \mathcal{I}_j] \wedge [(\forall l \in \Sigma_{L_j}^*)\, s\rho l\alpha \notin \mathcal{L}_j^p \cap \mathcal{Z}_{L_j} \cap \mathcal{I}_j]) \Leftrightarrow$
$([t\rho\alpha \in \mathcal{I}_j] \wedge [(\forall l \in \Sigma_{L_j}^*)\, t\rho l\alpha \notin \mathcal{L}_j^p \cap \mathcal{Z}_{L_j} \cap \mathcal{I}_j])$
4) $[s \in \mathcal{I}_{m_j}] \wedge [(\forall l \in \Sigma_{L_j}^*)\ sl \notin \mathcal{L}_{m_j}^p \cap \mathcal{Z}_{L_{j,m}} \cap \mathcal{I}_{m_j}] \Leftrightarrow [t \in \mathcal{I}_{m_j}] \wedge [(\forall l \in \Sigma_{L_j}^*)\ tl \notin \mathcal{L}_{m_j}^p \cap \mathcal{Z}_{L_{j,m}} \cap \mathcal{I}_{m_j}]$

Proof: See Appendix. ■

Combining the results from Propositions 10 and 12, it follows that each application of Ω_H is equivalent to either removing at least one state from $\mathbf{G}'_{\mathrm{HL}}$, or reaching a fixpoint. As we assume that all DES components have a finite statespace, it follows that $\mathbf{G}'_{\mathrm{HL}}$ also has a finite statespace. We can thus conclude that $\Omega_H(\mathcal{Z}_H)$ will reach a fixpoint in at most $|Q_H|$ steps and, by Theorem 3, the fixpoint will be the desired supremal element.

Similarly, by Propositions 10 and 13, we can conclude that $\Omega_{L_j}(\mathcal{Z}_{L_j})$ will reach a fixpoint in at most $|Q_{L_j}|$ steps and, by Theorem 4, the fixpoint will be the desired supremal element.

We next note that every regular language can be represented by a deterministic finite state automata [23]. The above result thus implies that as long as the languages involved are all regular, then Theorem 3 and Theorem 4 state that our fixpoint operators can always produce the required supremal languages in a finite number of steps.

V. Algorithms

In this section, we discuss the HISC algorithms that we have developed. We will first discuss automata based algorithms that rely on explicit lists of states and transitions, and then we briefly discuss the symbolic algorithms we have developed.

A. Automata based Algorithms

In [19], we first defined a set of algorithms to verify that a given system is interface consistent and then we defined algorithms to implement the high-level and the low-level fixpoint operators. Both sets of algorithms are based on explicit state and transition listings. Please see [19] for full details of data structures used, algorithms, and complexity analysis.

We now briefly discuss the scalability of our synthesis method. Let N_{E_H} denote the size of the statespace of $\mathbf{G}_H^p||\mathbf{E}_H$, and N_I and N_L be upper bounds for the statespaces of $\mathbf{G}_{I_k}$ and $\mathbf{G}_{L_k}^p||\mathbf{E}_{L_k}$ $(k = 1, \ldots, n)$, respectively. As was discussed in [6], the limiting factor for analyzing a flat system ($\mathbf{G} = \mathbf{G}_H||\mathbf{G}_{I_1}||\mathbf{G}_{L_1}|| \ldots ||\mathbf{G}_{I_n}||\mathbf{G}_{L_n}$) would typically be the size of its statespace, $N_{E_H} N_L^n$. For an HISC system, the limiting factor would be the size of the statespace of the high-level, $N_{E_H} N_I^n$, since it grows as we add more low-levels. We would expect our method to offer significant improvement as long as $N_I \ll N_L$.

Of course, this increased scalability comes with a price: a more restrictive architecture and thus the possible loss of global maximal permissiveness. We feel the tradeoff is worthwhile due to the increase in scalability and the other benefits of our approach [5].

B. Symbolic Algorithms

Both the HISC verification algorithms from [4]–[6], [19] and the HISC synthesis algorithms we just discussed, are currently based upon explicit state and transition listings which limit the size of a given level (i.e. high-level, or a given low-level) to about 10^7 states when 1GB of memory is used. As each subsystem in HISC is typically modeled as a group of plant DES and a group of specification/supervisor DES, each subsystem-wide state can be represented as a vector, where each element of the vector is the state of a component DES. Therefore, we can use Binary Decision Diagrams (BDD: [8], [9]) to represent the state space and transitions for each subsystem, and develop algorithms based on BDD representations to verify or synthesize supervisors for our HISC system.

In [21], [22], we present a set of predicate based, fixpoint operators for HISC synthesis, and prove that they converge to the desired HISC supremal languages for each level. We also provide a set of algorithms to implement these operators, as well as to perform symbolic HISC verification. We then discussed how to represent the system's transition function as predicates such that we were able to express the predicates required for the above algorithms. We were then able to implement our algorithms using Reduced Ordered Binary Decision Diagram [8], [9], making use of the BDD software package *BuDDy 2.4* developed by Jørn Lind-Nielsen. To achieve this, we drew heavily on the work of Ma et al. [13]. Please refer to [21] for details.

In [21], [22], we also introduced a method to implement our synthesized supervisors in a more compact manner. Typically, the result of synthesis is a single, very large supervisor that would be difficult to implement as a controller directly. For instance, the high-level supervisor from the AIP example in the next section has a statespace on the order of 10^{15}. Instead of representing this supervisor directly, we suggested using a state predicate[1] for each controllable event, and using observers for the required plant and specification DES to provide the current state information for the predicates. Each predicate can be represented as a BDD, and typically the BDD is much smaller than the corresponding automata supervisors. We also showed that HISC systems have the additional advantage that the enablement of high-level and request events could be determined using only the state of the high-level, while enablement of low-level and answer events could be determined using only the state of their corresponding low-level. See [21], [22] for complete details.

VI. AIP EXAMPLE

To demonstrate the utility of our method, we apply it to a large manufacturing system, the Atelier Inter-établissement de Productique (AIP) as described in [24], [25]. The AIP, shown in Fig. 3, is an automated manufacturing system consisting of a central loop (CL) and four external loops (EL), three assembly stations (AS), an input/output (I/O) station, and four inter-loop transport units (TU). The I/O station is where the pallets enter and leave the system. Pallets entering the system can be of type 1 or of type 2. The system consists of a high-level and eight low-levels; one for each transfer unit and assembly station, and one for the I/O station. As the example is quite large, we will only briefly introduce it. Please see [21] for complete details.

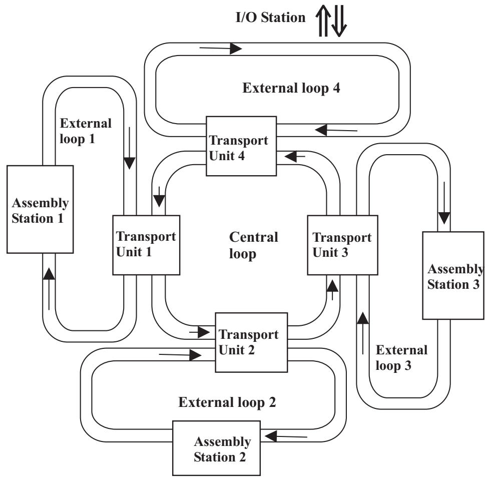

Fig. 3. The Atelier Inter-établissement de Productique

In [4], [7], Leduc et al. modelled the AIP as an HISC system. Using their algorithms based upon explicit state and transition listings, it took 254.7 seconds and 659MB to verify that the high-level, with 3.3×10^6 states, satisfied the HISC conditions. Using our BDD based algorithms, it took us 2 seconds and an estimated 30MB.

[1]A state predicate for a given DES $\mathbf{G}$ takes a state q from $\mathbf{G}$ and returns *true* or *false* to indicate (in this case) whether the event should be enabled at that state. See [21], [22] for more details.

We then extended the AIP example of [4], [7] by modelling how pallets move around the system. For example, a pallet can't reach an assembly station until it is transported from the central loop to the section of the external loop leading to the station. We also enforced capacity restrictions on each loop section as follows: maximum four pallets at a time in a given section of the CL, and five pallets at a time for a section of an EL. This was not originally modelled by Leduc et al. as it made the high-level too large for their software to handle.

To verify the system, we needed to add an additional supervisor that restricted the number of pallets in the system (excluding EL4) to 15, to prevent the system from blocking. The system was verified on a 2.8 GHz Pentium 4 CPU, with 512MB memory, running Fedora Core 2. It used less than 160MB of RAM, took 25.7 minutes to verify that the high-level HISC conditions were satisfied and less than 1 second to verify that the low-level HISC conditions were satisfied for each low-level. The reachable statespace for the high-level was 5.16×10^{13}, and the total estimated, worst case, reachable statespace for the flat system was 7.04×10^{28}, although the actual statespace is likely a lot smaller. A flat verification with our BDD tool quickly used up all available RAM, and had failed to complete after 24 hours.

We then removed the "15 pallets in system" supervisor, and performed a HISC synthesis. Our BDD tool used less than 160MB of RAM, took 128 minutes to synthesize a high-level supervisor, and less than 1 second to synthesize each low-level supervisor. The reachable statespace for the high-level was 1.14×10^{15}, and the total estimated, worst case, reachable statespace was 1.51×10^{30}, although the actual size is likely a lot smaller. This is a clear improvement over previous HISC algorithms from [4]–[6], [19].

It is interesting to note that typically we can not perform a flat synthesis (eg. the *supcon* algorithm from TCT [3]) on an HISC system and expect to receive a useful result. For example, if a high-level specification required that a controllable answer event be disabled, the flat synthesis would just disable the event. This would not produce the desired effect as this would still allow the corresponding low-level task to occur, and would only prevent the low-level from reporting that the task was complete. However, the high-level HISC synthesis would be unable to disable the answer event and would be forced to disable the request events leading to the answer event. This would have the correct effect of preventing the low-level task from occurring.

VII. Conclusions

In this paper we developed a synthesis method for the Hierarchical Interface-Based Supervisory Control system that does a per level synthesis to construct for each level a maximally permissive supervisor that satisfies the corresponding HISC conditions.

As the synthesis is done on a per level basis, the complete system model never needs to be stored in memory or traversed, offering potentially significant savings in computational resources. In fact, as long as the statespace of the interfaces are much smaller than the statespace of the corresponding low-levels, we should see significant reduction in complexity over a standard flat synthesis approach.

Combined with symbolic methods implemented using binary decision diagrams, we are now able to handle HISC systems with individual levels significantly larger (eight orders of magnitude in the AIP example) than approaches based upon explicit state and transition listings.

The benefits of our approach were illustrated by a large example (worst case statespace on order of 10^{30}) based on the AIP example. The example demonstrates how the HISC synthesis method can be applied to interesting systems of realistic complexity.

Appendix
Proofs of Selected Propositions

Proposition 5:

Proof: Proof can be broken down into three parts: 1) show $\mathcal{C}_{L_j}(E)$ is nonempty, 2) show $\mathcal{C}_{L_j}(E)$ is closed under arbitrary union, and 3) show $\mathcal{C}_{L_j}(E)$ contains a (unique) supremal element. As the first and last step are basically the same as for the standard supremal controllable sublanguage proof, they will be omitted. Refer to [19] for specific details of these steps.

Let $E \subseteq \Sigma^*$ and then show $\mathcal{C}_{L_j}(E)$ is closed under arbitrary union.

Let $Z_\beta \in \mathcal{C}_{L_j}(E)$ for all $\beta \in B$, where B is an index set. Let $Z = \cup\{Z_\beta | \, \beta \in B\}$.

Clearly $Z_\beta \subseteq Z$ for each $\beta \in B$.

$\Rightarrow (\forall \beta \in B)\, \overline{Z_\beta} \subseteq \overline{Z}$ as prefix closure preserves ordering. **(1)**

Sufficient to show that $Z \in \mathcal{C}_{L_j}(E)$.

Clearly, $Z \subseteq E$. All we still need to show is that Z is LICj with respect to system Φ.

This means showing that for all $s \in \mathcal{L}_j^p \cap \mathcal{I}_j \cap \overline{Z}$, the following conditions are satisfied:

1) $\text{Elig}_{\mathcal{L}_j^p}(s) \cap \Sigma_u \subseteq \text{Elig}_{\overline{Z} \cap \mathcal{I}_j}(s)$
2) $\text{Elig}_{\mathcal{I}_j}(s) \cap \Sigma_{R_j} \subseteq \text{Elig}_{\mathcal{L}_j^p \cap \overline{Z}}(s)$
3) $(\forall \rho \in \Sigma_{R_j})(\forall \alpha \in \Sigma_{A_j})\ s\rho\alpha \in \mathcal{I}_j \Rightarrow (\exists l \in \Sigma_{L_j}^*)\, s\rho l\alpha \in \mathcal{L}_j^p \cap \overline{Z} \cap \mathcal{I}_j$
4) $s \in \mathcal{I}_{m_j} \Rightarrow (\exists l \in \Sigma_{L_j}^*)\ sl \in \mathcal{L}_{m_j}^p \cap Z \cap \mathcal{I}_{m_j}$

Let $s \in \mathcal{L}_j^p \cap \mathcal{I}_j \cap \overline{Z}$. **(2)**

We first note that this gives us $s \in \overline{Z}$

$\Rightarrow (\exists s' \in \Sigma^*)\, ss' \in Z$

$\Rightarrow (\exists \beta \in B)\, ss' \in Z_\beta$, by definition of Z.

$\Rightarrow s \in \overline{Z}_\beta$

We thus have: $s \in \mathcal{L}_j^p \cap \mathcal{I}_j \cap \overline{Z}_\beta$, by **(2)**. **(3)**

a-b) Show $\text{Elig}_{\mathcal{L}_j^p}(s) \cap \Sigma_u \subseteq \text{Elig}_{\overline{Z} \cap \mathcal{I}_j}(s)$ and $\text{Elig}_{\mathcal{I}_j}(s) \cap \Sigma_{R_j} \subseteq \text{Elig}_{\mathcal{L}_j^p \cap \overline{Z}}(s)$
As both conditions are of the form of a standard controllability definition, their proofs are omitted. Refer to [19] for specific details of these parts.

c) Show $(\forall \rho \in \Sigma_{R_j})(\forall \alpha \in \Sigma_{A_j})\ s\rho\alpha \in \mathcal{I}_j \Rightarrow (\exists l \in \Sigma_{L_j}^*)\, s\rho l\alpha \in \mathcal{L}_j^p \cap \overline{Z} \cap \mathcal{I}_j$
Let $\rho \in \Sigma_{R_j}$, $\alpha \in \Sigma_{A_j}$ and assume $s\rho\alpha \in \mathcal{I}_j$. **(4)**

Show implies $(\exists l \in \Sigma_{L_j}^*)\, s\rho l\alpha \in \mathcal{L}_j^p \cap \overline{Z} \cap \mathcal{I}_j$

We immediately have: $s \in \mathcal{L}_j^p \cap \mathcal{I}_j \cap \overline{Z}_\beta$, $\rho \in \Sigma_{R_j}$, $\alpha \in \Sigma_{A_j}$, and $s\rho\alpha \in \mathcal{I}_j$ by **(3)**, and **(4)**.

As $Z_\beta \in \mathcal{C}_{L_j}(E)$ and thus LICj, we have: $(\exists l \in \Sigma_{L_j}^*)\, s\rho l\alpha \in \mathcal{L}_j^p \cap \overline{Z}_\beta \cap \mathcal{I}_j$

$\Rightarrow s\rho l\alpha \in \mathcal{L}_j^p \cap \overline{Z} \cap \mathcal{I}_j$ (by**(1)**), as required.

d) Show $s \in \mathcal{I}_{m_j} \Rightarrow (\exists l \in \Sigma_{L_j}^*)\ sl \in \mathcal{L}_{m_j}^p \cap Z \cap \mathcal{I}_{m_j}$
Assume $s \in \mathcal{I}_{m_j}$ and then show implies $(\exists l \in \Sigma_{L_j}^*)\ sl \in \mathcal{L}_{m_j}^p \cap Z \cap \mathcal{I}_{m_j}$. **(5)**

We have: $s \in \mathcal{L}_j^p \cap \mathcal{I}_j \cap \overline{Z}_\beta$ and $s \in \mathcal{I}_{m_j}$ by **(3)** and **(5)**.

As $Z_\beta \in \mathcal{C}_{L_j}(E)$ and thus LICj, we have: $(\exists l \in \Sigma_{L_j}^*)\ sl \in \mathcal{L}_{m_j}^p \cap Z_\beta \cap \mathcal{I}_{m_j}$

$\Rightarrow sl \in \mathcal{L}_{m_j}^p \cap Z \cap \mathcal{I}_{m_j}$ (by definition of Z), as required.

From parts a, b, c and d, we can conclude that Z is LICj with respect to system Φ.

We can thus conclude that $Z \in \mathcal{C}_{L_j}(E)$, as required. ■

Proposition 6:

Proof: Let $Z, Z' \in \text{Pwr}(\Sigma^*)$ and assume $Z \subseteq Z'$. **(1)**

Let $s \in \Omega_{\mathrm{LIC}_j}(Z)$ and then show implies: $s \in \Omega_{\mathrm{LIC}_j}(Z')$. **(2)**

By definition of Ω_{LIC_j} operator, it is sufficient to show: $s \in \overline{Z'} - \mathrm{Ext}_{\overline{Z'}}(\mathrm{FailLIC}_j(\overline{Z'}))$

From **(2)**, we have: $s \in \Omega_{\mathrm{LIC}_j}(Z)$

$\Rightarrow s \in \overline{Z} - \mathrm{Ext}_{\overline{Z}}(\mathrm{FailLIC}_j(\overline{Z}))$, by definition of Ω_{LIC_j}.

$\Rightarrow s \in \overline{Z} \ \wedge \ s \notin \mathrm{Ext}_{\overline{Z}}(\mathrm{FailLIC}_j(\overline{Z}))$ **(3)**

$\Rightarrow s \in \overline{Z'}$ as $Z \subseteq Z'$ (by **(1)**). **(4)**

All that remains now is to show that: $s \notin \mathrm{Ext}_{\overline{Z'}}(\mathrm{FailLIC}_j(\overline{Z'}))$

This means showing: $s \notin \{t \in \overline{Z'} \,|\, t' \leq t$ for some $t' \in \mathrm{FailLIC}_j(\overline{Z'})\}$.

Thus sufficient to show that: $(\forall s' \leq s)\, s' \notin \mathrm{FailLIC}_j(\overline{Z'})$

Substituting for $\mathrm{FailHIC}(\overline{Z'})$, we see we must show:

$$\begin{aligned}
(\forall s' \leq s)\, s' \notin \{t \in \mathcal{L}_j^p \cap \mathcal{I}_j \cap \overline{Z'} \,|\, &\neg[\mathrm{Elig}_{\mathcal{L}_j^p}(t) \cap \Sigma_u \subseteq \mathrm{Elig}_{\overline{Z'} \cap \mathcal{I}_j}(t)] \\
&\vee \neg[\mathrm{Elig}_{\mathcal{I}_j}(t) \cap \Sigma_{R_j} \subseteq \mathrm{Elig}_{\mathcal{L}_j^p \cap \overline{Z'}}(t)] \\
&\vee \neg[(\forall \rho \in \Sigma_{R_j})(\forall \alpha \in \Sigma_{A_j})\, t\rho\alpha \in \mathcal{I}_j \Rightarrow (\exists l \in \Sigma_{L_j}^*)\, t\rho l\alpha \in \mathcal{L}_j^p \cap \overline{Z'} \cap \mathcal{I}_j] \\
&\vee \neg[t \in \mathcal{I}_{m_j} \Rightarrow (\exists l \in \Sigma_{L_j}^*)\, tl \in \mathcal{L}_{m_j}^p \cap Z' \cap \mathcal{I}_{m_j}]\}
\end{aligned}$$

Which means it's sufficient to show:

$$\begin{aligned}
(\forall s' \leq s)\, s' \in \mathcal{L}_j^p \cap \mathcal{I}_j \cap \overline{Z'} \Rightarrow &[\mathrm{Elig}_{\mathcal{L}_j^p}(s') \cap \Sigma_u \subseteq \mathrm{Elig}_{\overline{Z'} \cap \mathcal{I}_j}(s')] \\
&\wedge [\mathrm{Elig}_{\mathcal{I}_j}(s') \cap \Sigma_{R_j} \subseteq \mathrm{Elig}_{\mathcal{L}_j^p \cap \overline{Z'}}(s')] \\
&\wedge [(\forall \rho \in \Sigma_{R_j})(\forall \alpha \in \Sigma_{A_j})\, s'\rho\alpha \in \mathcal{I}_j \Rightarrow (\exists l \in \Sigma_{L_j}^*)\, s'\rho l\alpha \in \mathcal{L}_j^p \cap \overline{Z'} \cap \mathcal{I}_j] \\
&\wedge [s' \in \mathcal{I}_{m_j} \Rightarrow (\exists l \in \Sigma_{L_j}^*)\, s'l \in \mathcal{L}_{m_j}^p \cap Z' \cap \mathcal{I}_{m_j}] \qquad (\dagger)
\end{aligned}$$

Let $s' \leq s$ and assume $s' \in \mathcal{L}_j^p \cap \mathcal{I}_j \cap \overline{Z'}$ **(6)**

We next note that we have $s \notin \mathrm{Ext}_{\overline{Z}}(\mathrm{FailLIC}_j(\overline{Z}))$ by **(3)**.

$$\begin{aligned}
\Rightarrow (\forall s'' \leq s)\, s'' \in \mathcal{L}_j^p \cap \mathcal{I}_j \cap \overline{Z} \Rightarrow &[\mathrm{Elig}_{\mathcal{L}_j^p}(s'') \cap \Sigma_u \subseteq \mathrm{Elig}_{\overline{Z} \cap \mathcal{I}_j}(s'')] \\
&\wedge [\mathrm{Elig}_{\mathcal{I}_j}(s'') \cap \Sigma_{R_j} \subseteq \mathrm{Elig}_{\mathcal{L}_j^p \cap \overline{Z}}(s'')] \\
&\wedge [(\forall \rho \in \Sigma_{R_j})(\forall \alpha \in \Sigma_{A_j})\, s''\rho\alpha \in \mathcal{I}_j \Rightarrow (\exists l \in \Sigma_{L_j}^*)\, s''\rho l\alpha \in \mathcal{L}_j^p \cap \overline{Z} \cap \mathcal{I}_j] \\
&\wedge [s'' \in \mathcal{I}_{m_j} \Rightarrow (\exists l \in \Sigma_{L_j}^*)\, s''l \in \mathcal{L}_{m_j}^p \cap Z \cap \mathcal{I}_{m_j}] \qquad \textbf{(7)}
\end{aligned}$$

We note that as $s' \leq s$ by **(6)**, and $s \in \overline{Z}$ by **(3)**, we have: $s' \in \mathcal{L}_j^p \cap \mathcal{I}_j \cap \overline{Z}$ (by **(6)**)

$$\begin{aligned}
\Rightarrow [\mathrm{Elig}_{\mathcal{L}_j^p}(s') \cap \Sigma_u \subseteq \mathrm{Elig}_{\overline{Z} \cap \mathcal{I}_j}(s')] \wedge &[\mathrm{Elig}_{\mathcal{I}_j}(s') \cap \Sigma_{R_j} \subseteq \mathrm{Elig}_{\mathcal{L}_j^p \cap \overline{Z}}(s')] \\
&\wedge [(\forall \rho \in \Sigma_{R_j})(\forall \alpha \in \Sigma_{A_j})\, s'\rho\alpha \in \mathcal{I}_j \Rightarrow (\exists l \in \Sigma_{L_j}^*)\, s'\rho l\alpha \in \mathcal{L}_j^p \cap \overline{Z} \cap \mathcal{I}_j] \\
&\wedge [s' \in \mathcal{I}_{m_j} \Rightarrow (\exists l \in \Sigma_{L_j}^*)\, s'l \in \mathcal{L}_{m_j}^p \cap Z \cap \mathcal{I}_{m_j}] \qquad \textbf{(8)}
\end{aligned}$$

We next note $\overline{Z} \subseteq \overline{Z'}$, as $Z \subseteq Z'$ (by **(1)**). **(9)**

We will now show that $(\dagger)$ is satisfied.

a) Show $\text{Elig}_{\mathcal{L}_j^p}(s') \cap \Sigma_u \subseteq \text{Elig}_{\overline{Z'} \cap \mathcal{I}_j}(s')$
Sufficient to show: $(\forall \sigma \in \Sigma_u)\, s'\sigma \in \mathcal{L}_j^p \Rightarrow s\sigma \in \overline{Z'} \cap \mathcal{I}_j$

Follows immediately from **(8)** and **(9)**.

b) Show $\text{Elig}_{\mathcal{I}_j}(s') \cap \Sigma_{R_j} \subseteq \text{Elig}_{\mathcal{L}_j^p \cap \overline{Z'}}(s')$
Sufficient to show: $(\forall \rho \in \Sigma_{R_j})\, s'\rho \in \mathcal{I}_j \Rightarrow s'\rho \in \mathcal{L}_j^p \cap \overline{Z'}$

Follows immediately from **(8)** and **(9)**.

c) Show $(\forall \rho \in \Sigma_{R_j})(\forall \alpha \in \Sigma_{A_j})\, s'\rho\alpha \in \mathcal{I}_j \Rightarrow (\exists l \in \Sigma_{L_j}^*)\, s'\rho l\alpha \in \mathcal{L}_j^p \cap \overline{Z'} \cap \mathcal{I}_j$
Follows immediately from **(8)** and **(9)**.

d) Show $s' \in \mathcal{I}_{m_j} \Rightarrow (\exists l \in \Sigma_{L_j}^*)\, s'l \in \mathcal{L}_{m_j}^p \cap Z' \cap \mathcal{I}_{m_j}$
Follows immediately from **(1)** and **(8)**.

By parts a-d, we can now conclude that $(\dagger)$ is satisfied. ■

Proposition 7:

Proof:

1. Show $Z \subseteq Z' \Rightarrow (\forall i \in \{0, 1, 2, \ldots\})\, \Omega_{L_j}^i(Z) \subseteq \Omega_{L_j}^i(Z')$.
This can be shown easily via proof by induction and fact that $\Omega_{L_j}^i$ is monotone. For specific details, refer to [19].

2. Show $\Omega_{L_j}(Z) = Z \Rightarrow Z \in \mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})$.
Assume $\Omega_{L_j}(Z) = Z$ and then show implies $Z \in \mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})$. **(3)**

Sufficient to show $Z \subseteq \mathcal{Z}_{L_{j,m}}$ and Z is LICj.

By **(3)**, we have: $Z = \Omega_{\text{LNB}_j}(\Omega_{\text{LIC}_j}(Z))$

$\Rightarrow Z = [\overline{Z} - \text{Ext}_{\overline{Z}}(\text{FailLIC}_j(\overline{Z}))] \cap \mathcal{Z}_{L_{j,m}}$

$\Rightarrow Z \subseteq \mathcal{Z}_{L_{j,m}}$ and $Z \subseteq [\overline{Z} - \text{Ext}_{\overline{Z}}(\text{FailLIC}_j(\overline{Z}))]$. **(4)**

Still need to show Z is LICj.

We thus need to show $\text{FailLIC}_j(\overline{Z}) = \emptyset$. Using proof by contradiction:

Assume FailLIC$_j(\overline{Z}) \neq \emptyset$.

$\Rightarrow \exists s \in$ FailLIC$_j(\overline{Z})$ **(5)**

As FailLIC$_j(\overline{Z}) \subseteq \overline{Z}$ by definition, we have $s \in \overline{Z}$. **(6)**

$\Rightarrow (\exists s' \in \Sigma^*)\, ss' \in Z$ **(7)**

We can also conclude by **(5)** that: $s \in \text{Ext}_{\overline{Z}}(\text{FailLIC}_j(\overline{Z}))$

However, we have by **(7)** and **(4)** that: $ss' \in \overline{Z} - \text{Ext}_{\overline{Z}}(\text{FailLIC}_j(\overline{Z}))$

$\Rightarrow ss' \notin \text{Ext}_{\overline{Z}}(\text{FailLIC}_j(\overline{Z})) \wedge ss' \in \overline{Z}$

$\Rightarrow (\forall s'' \in \text{FailLIC}_j(\overline{Z}))\, \neg(s'' \leq ss')$ which contradicts **(5)**.

We thus conclude that FailLIC$_j(\overline{Z}) = \emptyset$.

$$\begin{aligned}\Rightarrow (\forall t \in \mathcal{L}_j^p \cap \mathcal{I}_j \cap \overline{Z})\, &[\text{Elig}_{\mathcal{L}_j^p}(t) \cap \Sigma_u \subseteq \text{Elig}_{\overline{Z} \cap \mathcal{I}_j}(t)] \\ &\wedge [\text{Elig}_{\mathcal{I}_j}(t) \cap \Sigma_{R_j} \subseteq \text{Elig}_{\mathcal{L}_j^p \cap \overline{Z}}(t)] \\ &\wedge [(\forall \rho \in \Sigma_{R_j})(\forall \alpha \in \Sigma_{A_j})\, t\rho\alpha \in \mathcal{I}_j \Rightarrow (\exists l \in \Sigma_{L_j}^*)\, t\rho l\alpha \in \mathcal{L}_j^p \cap \overline{Z} \cap \mathcal{I}_j] \\ &\wedge [t \in \mathcal{I}_{m_j} \Rightarrow (\exists l \in \Sigma_{L_j}^*)\, tl \in \mathcal{L}_{m_j}^p \cap Z \cap \mathcal{I}_{m_j}]\end{aligned}$$

$\Rightarrow Z$ is LICj, by *Defn.* 12. ■

Proposition 8:

Proof: To prove that sup$\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})$ is the greatest fixpoint of Ω_{L_j}, we need to show:

1) sup$\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}}) = \Omega_{L_j}(\text{sup}\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}}))$
2) $(\forall Z \in \text{Pwr}(\Sigma^*))\, Z = \Omega_{L_j}(Z) \Rightarrow Z \subseteq \text{sup}\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})$

The second part follows from point 2 of Proposition 7 and fact that sup$\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})$ is the supremal element of $\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})$.

We will now show sup$\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}}) = \Omega_{L_j}(\text{sup}\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}}))$.

We note that by definition we have: $\Omega_{L_j}(\text{sup}\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})) = \Omega_{\text{LIC}_j}(\text{sup}\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})) \cap \mathcal{Z}_{L_{j,m}}$ **(1)**

By definition of Ω_{HIC} we have: **(2)**

$$\Omega_{L_j}(\text{sup}\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})) = [\overline{\text{sup}\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})} - \text{Ext}_{\overline{\text{sup}\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})}}(\text{FailLIC}_j(\overline{\text{sup}\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})}))] \cap \mathcal{Z}_{L_{j,m}}$$

As sup$\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})$ is LICj, it follows that: FailLIC$_j(\overline{\text{sup}\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})}) = \emptyset$

$\Rightarrow \Omega_{L_j}(\text{sup}\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})) = [\overline{\text{sup}\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})} - \emptyset] \cap \mathcal{Z}_{L_{j,m}} = \overline{\text{sup}\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})} \cap \mathcal{Z}_{L_{j,m}}$ **(3)**

(I) Show sup$\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}}) \subseteq \Omega_{L_j}(\text{sup}\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}}))$

Sufficient to show $\sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}}) \subseteq \overline{\sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})} \cap \mathcal{Z}_{L_{j,m}}$

We note $\sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}}) \subseteq \overline{\sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})}$ is automatic.

$\Rightarrow \sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}}) \subseteq \overline{\sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})} \cap \mathcal{Z}_{L_{j,m}}$, as $\sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}}) \in \mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})$.

(II) Show $\Omega_{L_j}(\sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})) \subseteq \sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})$

Let $s \in \Omega_{L_j}(\sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}}))$ and then show implies $s \in \sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})$. **(4)**

From **(4)** and **(2)**, we can conclude that:

$s \in [\overline{\sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})} - \text{Ext}_{\overline{\sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})}}(\text{FailLIC}_j(\overline{\sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})}))] \cap \mathcal{Z}_{L_{j,m}}$ **(5)**

$\Rightarrow [s \in \overline{\sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})}] \wedge [s \notin \text{Ext}_{\overline{\sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})}}(\text{FailLIC}_j(\overline{\sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})}))]$ **(6)**

$\Rightarrow (\forall s' \leq s)\; s' \notin \text{FailLIC}_j(\overline{\sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})})$

$\Rightarrow (\forall s' \in \overline{\{s\}})\; s' \notin \text{FailLIC}_j(\overline{\sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})})$ **(7)**

We note that $s \in \overline{\sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})} \Rightarrow s \in \mathcal{Z}_{L_j}$ as $\mathcal{Z}_{L_j}$ is closed and $\sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}}) \subseteq \mathcal{Z}_{L_{j,m}} \subseteq \mathcal{Z}_{L_j}$.

$\Rightarrow s \in \mathcal{L}_j^p \cap \mathcal{I}_j$ by definition of $\mathcal{Z}_{L_j}$.

$\Rightarrow s \in \mathcal{L}_j^p \cap \mathcal{I}_j \cap \overline{\sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})}$

$\Rightarrow (\forall s' \in \overline{\{s\}})\; s' \in \mathcal{L}_j^p \cap \mathcal{I}_j \cap \overline{\sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})}$, as the languages are closed.

By **(7)**, we have for all $s' \in \overline{\{s\}}$: **(8)**

$[s' \in \mathcal{L}_j^p \cap \mathcal{I}_j \cap \overline{\sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})}] \wedge [s' \in \mathcal{I}_{m_j} \Rightarrow (\exists l \in \Sigma_{L_j}^*)\; s'l \in \mathcal{L}_{m_j}^p \cap \sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}}) \cap \mathcal{I}_{m_j}]$

$\wedge [\text{Elig}_{\mathcal{L}_j^p}(s') \cap \Sigma_u \subseteq \text{Elig}_{\overline{\sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})} \cap \mathcal{I}_j}(s')] \wedge [\text{Elig}_{\mathcal{I}_j}(s') \cap \Sigma_{R_j} \subseteq \text{Elig}_{\mathcal{L}_j^p \cap \overline{\sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})}}(s')]$

$\wedge [(\forall \rho \in \Sigma_{R_j})(\forall \alpha \in \Sigma_{A_j})\; s'\rho\alpha \in \mathcal{I}_j \Rightarrow (\exists l \in \Sigma_{L_j}^*)\; s'\rho l\alpha \in \mathcal{L}_j^p \cap \overline{\sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})} \cap \mathcal{I}_j]$

Let $Z = \sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}}) \cup \{s\}$ **(9)**

Will show that $Z \in \mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})$, thus $Z \subseteq \sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})$, thus $s \in \sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})$.

By **(9)**, we have: $\sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}}) \subseteq Z \Rightarrow \overline{\sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})} \subseteq \overline{Z}$ **(10)**

As $\sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}}) \subseteq \mathcal{Z}_{L_{j,m}}$ and $s \in \mathcal{Z}_{L_{j,m}}$ (by **(5)**), we have: $Z \subseteq \mathcal{Z}_{L_{j,m}}$

Now need to show Z is LICj.

Let $t \in \mathcal{L}_j^p \cap \mathcal{I}_j \cap \overline{Z}$ **(11)**

Now must show the following: *(1)* $\text{Elig}_{\mathcal{L}_j^p}(t) \cap \Sigma_u \subseteq \text{Elig}_{\overline{Z} \cap \mathcal{I}_j}(t)$, *(2)* $\text{Elig}_{\mathcal{I}_j}(t) \cap \Sigma_{R_j} \subseteq \text{Elig}_{\mathcal{L}_j^p \cap \overline{Z}}(t)$, *(3)* $(\forall \rho \in \Sigma_{R_j})(\forall \alpha \in \Sigma_{A_j})\, t\rho\alpha \in \mathcal{I}_j \Rightarrow (\exists l \in \Sigma_{L_j}^*)\, t\rho l\alpha \in \mathcal{L}_j^p \cap \overline{Z} \cap \mathcal{I}_j$, and *(4)* $t \in \mathcal{I}_{m_j} \Rightarrow (\exists l \in \Sigma_{L_j}^*)\; tl \in \mathcal{L}_{m_j}^p \cap Z \cap \mathcal{I}_{m_j}$

1) Show $\text{Elig}_{\mathcal{L}_j^p}(t) \cap \Sigma_u \subseteq \text{Elig}_{\overline{Z} \cap \mathcal{I}_j}(t)$

Let $\sigma \in \Sigma_u$, and $t\sigma \in \mathcal{L}_j^p$.

Sufficient to show $t\sigma \in \overline{Z} \cap \mathcal{I}_j$.

If $t \in \overline{Z} - \overline{\{s\}}$, we have $t \in \overline{\text{sup}\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})}$.

$\Rightarrow t\sigma \in \overline{\text{sup}\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})} \cap \mathcal{I}_j \subseteq \overline{Z} \cap \mathcal{I}_j$, as $\text{sup}\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})$ is LICj and by **(10)**.

If $t \in \overline{\{s\}}$, it follows directly from **(8)** and **(10)**.

2) Show $\text{Elig}_{\mathcal{I}_j}(t) \cap \Sigma_{R_j} \subseteq \text{Elig}_{\mathcal{L}_j^p \cap \overline{Z}}(t)$

Proof similar to part (1).

3) Show $(\forall \rho \in \Sigma_{R_j})(\forall \alpha \in \Sigma_{A_j})\, t\rho\alpha \in \mathcal{I}_j \Rightarrow (\exists l \in \Sigma_{L_j}^*)\, t\rho l\alpha \in \mathcal{L}_j^p \cap \overline{Z} \cap \mathcal{I}_j$

Let $\rho \in \Sigma_{R_j}$, $\alpha \in \Sigma_{A_j}$, $t\rho\alpha \in \mathcal{I}_j$ and then show implies $(\exists l \in \Sigma_{L_j}^*)\, t\rho l\alpha \in \mathcal{L}_j^p \cap \overline{Z} \cap \mathcal{I}_j$.

If $t \in \overline{Z} - \overline{\{s\}}$, we have $t \in \overline{\text{sup}\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})}$.

$\Rightarrow (\exists l \in \Sigma_{L_j}^*)\, t\rho l\alpha \in \mathcal{L}_j^p \cap \overline{\text{sup}\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})} \cap \mathcal{I}_j \subseteq \mathcal{L}_j^p \cap \overline{Z} \cap \mathcal{I}_j$, as $\text{sup}\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})$ is LICj.

If $t \in \overline{\{s\}}$, it follows directly from **(8)** and **(10)**.

4) Show $t \in \mathcal{I}_{m_j} \Rightarrow (\exists l \in \Sigma_{L_j}^*)\, tl \in \mathcal{L}_{m_j}^p \cap Z \cap \mathcal{I}_{m_j}$

Assume $t \in \mathcal{I}_{m_j}$ and then show implies $(\exists l \in \Sigma_{L_j}^*)\, tl \in \mathcal{L}_{m_j}^p \cap Z \cap \mathcal{I}_{m_j}$

If $t \in \overline{Z} - \overline{\{s\}}$, we have $t \in \overline{\text{sup}\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})}$.

$\Rightarrow (\exists l \in \Sigma_{L_j}^*)\, tl \in \mathcal{L}_{m_j}^p \cap \text{sup}\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}}) \cap \mathcal{I}_{m_j} \subseteq \mathcal{L}_{m_j}^p \cap Z \cap \mathcal{I}_{m_j}$, as $\text{sup}\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})$ is LICj.

If $t \in \overline{\{s\}}$, it follows directly from **(8)** and **(10)**.

By points 1–4, we have Z is LICj; thus $s \in \text{sup}\mathcal{C}_H(\mathcal{Z}_{H_m})$. Part (II) complete.

By part (I) and (II), we get $\text{sup}\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}}) = \Omega_{L_j}(\text{sup}\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}}))$ as required. ■

Proposition 13:

Proof: Let $s, t \in L(\mathbf{G}'_{\text{LL}_j})$, and assume $\delta(q_o, s) = \delta(q_o, t)$. **(1)**

By Proposition 9 and definition of $\mathbf{G}_{\text{LL}_j}$, we have: *(i)* $s \equiv_{\mathcal{L}_j^p} t$, *(ii)* $s \equiv_{\mathcal{L}_{m_j}^p} t$, *(iii)* $s \equiv_{\mathcal{E}_{L_j}} t$, *(iv)* $s \equiv_{\mathcal{E}_{L_{j,m}}} t$, *(v)* $s \equiv_{\mathcal{I}_j} t$, and *(vi)* $s \equiv_{\mathcal{I}_{m_j}} t$. **(2)**

1. Show $\text{Elig}_{\mathcal{L}_j^p}(s) \cap \Sigma_u \not\subseteq \text{Elig}_{\mathcal{Z}_{L_j} \cap \mathcal{I}_j}(s) \Leftrightarrow \text{Elig}_{\mathcal{L}_j^p}(t) \cap \Sigma_u \not\subseteq \text{Elig}_{\mathcal{Z}_{L_j} \cap \mathcal{I}_j}(t)$

Follows easily from Proposition 11.

2. Show $\text{Elig}_{\mathcal{I}_j}(s) \cap \Sigma_{R_j} \not\subseteq \text{Elig}_{\mathcal{L}_j^p \cap \mathcal{Z}_{L_j}}(s) \Leftrightarrow \text{Elig}_{\mathcal{I}_j}(t) \cap \Sigma_{R_j} \not\subseteq \text{Elig}_{\mathcal{L}_j^p \cap \mathcal{Z}_{L_j}}(t)$
Follows easily from Proposition 11.

3. Show $(\forall \rho \in \Sigma_{R_j})(\forall \alpha \in \Sigma_{A_j})\ ([s\rho\alpha \in \mathcal{I}_j] \wedge [(\forall l \in \Sigma_{L_j}^*)\, s\rho l\alpha \notin \mathcal{L}_j^p \cap \mathcal{Z}_{L_j} \cap \mathcal{I}_j]) \Leftrightarrow$

$$([t\rho\alpha \in \mathcal{I}_j] \wedge [(\forall l \in \Sigma_{L_j}^*)\, t\rho l\alpha \notin \mathcal{L}_j^p \cap \mathcal{Z}_{L_j} \cap \mathcal{I}_j])$$

Let $\rho \in \Sigma_{R_j}$, and $\alpha \in \Sigma_{A_j}$.

As condition symmetric, sufficient to prove:

$$[s\rho\alpha \in \mathcal{I}_j] \wedge [(\forall l \in \Sigma_{L_j}^*)\, s\rho l\alpha \notin \mathcal{L}_j^p \cap \mathcal{Z}_{L_j} \cap \mathcal{I}_j] \Rightarrow [t\rho\alpha \in \mathcal{I}_j] \wedge [(\forall l \in \Sigma_{L_j}^*)\, t\rho l\alpha \notin \mathcal{L}_j^p \cap \mathcal{Z}_{L_j} \cap \mathcal{I}_j]$$

Assume $[s\rho\alpha \in \mathcal{I}_j] \wedge [(\forall l \in \Sigma_{L_j}^*)\, s\rho l\alpha \notin \mathcal{L}_j^p \cap \mathcal{Z}_{L_j} \cap \mathcal{I}_j]$. **(3)**

By **(2)**, **(3)**, and fact $\mathcal{Z}_{L_j} := \mathcal{L}_j^p \cap \mathcal{E}_{L_j} \cap \mathcal{I}_j$, we have: $[t\rho\alpha \in \mathcal{I}_j] \wedge [(\forall l \in \Sigma_{L_j}^*)\, t\rho l\alpha \notin \mathcal{L}_j^p \cap \mathcal{E}_{L_j} \cap \mathcal{I}_j$

$\Rightarrow [t\rho\alpha \in \mathcal{I}_j] \wedge [(\forall l \in \Sigma_{L_j}^*)\, t\rho l\alpha \notin \mathcal{L}_j^p \cap \mathcal{Z}_{L_j} \cap \mathcal{I}_j]$, as required.

4. Show $[s \in \mathcal{I}_{m_j}] \wedge [(\forall l \in \Sigma_{L_j}^*)\ sl \notin \mathcal{L}_{m_j}^p \cap \mathcal{Z}_{L_{j,m}} \cap \mathcal{I}_{m_j}] \Leftrightarrow [t \in \mathcal{I}_{m_j}] \wedge [(\forall l \in \Sigma_{L_j}^*)\ tl \notin \mathcal{L}_{m_j}^p \cap \mathcal{Z}_{L_{j,m}} \cap \mathcal{I}_{m_j}]$

As condition symmetric, sufficient to prove:

$$[s \in \mathcal{I}_{m_j}] \wedge [(\forall l \in \Sigma_{L_j}^*)\, sl \notin \mathcal{L}_{m_j}^p \cap \mathcal{Z}_{L_{j,m}} \cap \mathcal{I}_{m_j}] \Rightarrow [t \in \mathcal{I}_{m_j}] \wedge [(\forall l \in \Sigma_{L_j}^*)\, tl \notin \mathcal{L}_{m_j}^p \cap \mathcal{Z}_{L_{j,m}} \cap \mathcal{I}_{m_j}]$$

Assume $[s \in \mathcal{I}_{m_j}] \wedge [(\forall l \in \Sigma_{L_j}^*)\ sl \notin \mathcal{L}_{m_j}^p \cap \mathcal{Z}_{L_{j,m}} \cap \mathcal{I}_{m_j}]$. **(4)**

By **(2)**, **(4)**, and fact $\mathcal{Z}_{L_{j,m}} := \mathcal{L}_{m_j}^p \cap \mathcal{E}_{L_{j,m}} \cap \mathcal{I}_{m_j}$, we have:

$[t \in \mathcal{I}_{m_j}] \wedge [(\forall l \in \Sigma_{L_j}^*)\, tl \notin \mathcal{L}_{m_j}^p \cap \mathcal{E}_{L_{j,m}} \cap \mathcal{I}_{m_j}]$.

$\Rightarrow [t \in \mathcal{I}_{m_j}] \wedge [(\forall l \in \Sigma_{L_j}^*)\ tl \notin \mathcal{L}_{m_j}^p \cap \mathcal{Z}_{L_{j,m}} \cap \mathcal{I}_{m_j}]$, as required. ■

REFERENCES

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

[2] W. M. Wonham and P. Ramadge, "On the supremal controllable sublanguage of a given language," *SIAM J. Control Optim*, vol. 25, no. 3, pp. 637–659, May 1987.

[3] W. M. Wonham, *Supervisory Control of Discrete-Event Systems*, Department of Electrical and Computer Engineering, University of Toronto, July 2006, Monograph and TCT software can be downloaded at http://www.control.toronto.edu/DES/.

[4] R. Leduc, "Hierarchical interface-based supervisory control," Ph.D. dissertation, Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ont, 2002, [ONLINE] Available: http://www.cas.mcmaster.ca/~leduc.

[5] R. J. Leduc, B. A. Brandin, M. Lawford, and W. M. Wonham, "Hierarchical interface-based supervisory control, part I: Serial case," *IEEE Trans. Automatic Control*, vol. 50, no. 9, pp. 1322–1335, Sept. 2005.

[6] R. J. Leduc, M. Lawford, and W. M. Wonham, "Hierarchical interface-based supervisory control, part II: Parallel case," *IEEE Trans. Automatic Control*, vol. 50, no. 9, pp. 1336–1348, Sept. 2005.

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

[8] R. E. Bryant, “Graph-based algorithms for boolean function manipulation,” *IEEE Trans. Comput.*, vol. 35, no. 8, pp. 677–691, 1986.

[9] M. Huth and M. Ryan, *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge University Press, Nov. 1999.

[10] Z. Zhang and W. M. Wonham, “STCT: an efficient algorithm for supervisory control design,” in *Proc. of SCODES 2001*, INRIA, Paris, July 2001, pp. 82–93.

[11] A. Vahidi, B. Lennartson, and M. Fabian, “Efficient analysis of large discrete-event systems with binary decision diagrams,” in *Proc. of the 44th IEEE Conf. Decision Contr. and and 2005 European Contr. Conf.*, Seville, Spain, 2005, pp. 2751–2756.

[12] C. Ma and W. M. Wonham, “Control of state tree structures,” in *Proc. 11th Mediterranean Conference on Control and Automation*, June 2003, paper T4-005 (6pp.).

[13] ——, *Nonblocking Supervisory Control of State Tree Structures*, ser. Lecture Notes in Control and Information Sciences. Berlin, Germany: Springer-Verlag, 2005, vol. 317, a1: 20010101.

[14] B. Wang, “Top-down design for RW supervisory control theory,” Master’s thesis, Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ont, 1995.

[15] D. Harel, “A visual formalism for complex systems,” *Science of Computer Programming*, vol. 8, pp. 231–274, Jun. 1987.

[16] P. Pena, J. Cury, and S. Lafortune, “Testing modularity of local supervisors: An approach based on abstractions,” in *Proc. of WODES 2006*, Ann Arbor, USA, Jul. 2006, pp. 107–112.

[17] H. Flordal and R. Malik, “Modular nonblocking verification using conflict equivalence,” in *Proc. of WODES 2006*, Ann Arbor, USA, Jul. 2006, pp. 100–106.

[18] R. Hill and D. Tilbury, “Modular supervisory control of discrete-event systems with abstraction and incremental hierarchical construction,” in *Proc. of WODES 2006*, Ann Arbor, USA, Jul. 2006, pp. 399–406.

[19] P. Dai, “Synthesis method for hierarchical interface-based supervisory control,” Master’s thesis, Dept. of Computing and Software, McMaster University, Hamilton, Ont, 2006, [ONLINE] http://www.cas.mcmaster.ca/˜leduc/#studtheses.

[20] R. J. Leduc and P. Dai, “Synthesis method for hierarchical interface-based supervisory control,” in *Proc. of 26th American Control Conference*, New York City, USA, July 2007, pp. 4260–4267.

[21] R. Song, “Symbolic synthesis and verification of hierarchical interface-based supervisory control,” Master’s thesis, Dept. of Comput. and Softw., McMaster University, Hamilton, Ont, 2006, [ONLINE] http://www.cas.mcmaster.ca/˜leduc/#studtheses.

[22] R. Song and R. J. Leduc, “Symbolic synthesis and verification of hierarchical interface-based supervisory control,” in *Proc. of WODES 2006*, Ann Arbor, USA, Jul. 2006, pp. 419–426.

[23] D. C. Kozen, *Automata and Computability*. Springer, 1997.

[24] B. Brandin and F. Charbonnier, “The supervisory control of the automated manufacturing system of the AIP,” in *Proc. Rensselaer’s 1994 4th International Conf. on Comput. Integr. Mfg and Automation Technol.*, Troy, Oct 1994, pp. 319–324.

[25] F. Charbonnier, “Commande par supervision des systèmes à événements discrets: application à un site expérimental l’Atelier Inter-établissement de Productique,” Laboratoire d’Automatique de Grenoble, Grenoble, France, Tech. Rep., 1994.