Synthesis Method for Hierarchical Interface-based Supervisory Control

Ryan J. Leduc and Pengcheng Dai

Dept. of Computing and Software, McMaster University
email: leduc, daip@mcmaster.ca

***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 which 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 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 do a per level synthesis to 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 the algorithms that we have constructed that implement the fixpoint operators and show that they potentially offer significant improvement over the monolithic approach.

A large manufacturing system example (estimated worst case statespace on the order of 10^{22}) extended from the AIP example is discussed. A software tool for synthesis and verification of HISC systems using our approach was also developed.

I. Introduction

In the area of Discrete-Event Systems (DES), 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. ([7], [8], [9], [10]) 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.

There has been some very promising recent work by Pena et al. [11], Flordal et al. [4], and Hill et al. [5] 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 [8],[9], and [10], we will only discuss it briefly here. For a small illustrative HISC example, please see [8].

For the remainder of the paper we will be presenting our new results from [3], 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. We close by discussing a large manufacturing system example which was extended from the AIP example in [7], [9].

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 [14]. 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 $s \leq s$.

The *prefix closure* of language L, denoted $\overline{L}$, is defined as $\overline{L} = \{t \in \Sigma^* \mid t \leq s \text{ for some } s \in L\}$. 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^* mod 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 $\equiv_L$ 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 $\mathbf{G}$, the language generated 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.

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:

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

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 said to be *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 $L(\mathbf{G}) = \overline{\mathbf{K}}$.

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.

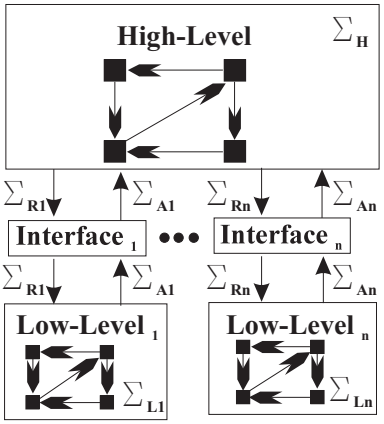

Fig. 1. Parallel Interface Block Diagram.

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

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

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 [3] that it is equivalent to the language based definition given in [8], [9], [10].

Definition 2: The j^{th} interface DES $\mathbf{G}_{I_j} = (X_j, \Sigma_{R_j} \dot\cup \Sigma_{A_j}, \xi_j, x_{o_j}, X_{m_j})$, with $X_{\text{rch}} \subseteq X_j$ its set of reachable states and $X_{rm} \subseteq X_{m_j}$ its set of reachable marked states, is a *command-pair interface* if:

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

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

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

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 [7], [8], [9], [10]. We use it since it makes the synthesis definitions clearer. We show in [3] that our new definition is equivalent.

Definition 3: The n^{th} degree ($n \geq 1$) parallel 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) $(\forall s \in \mathcal{H} \cap \mathcal{I})\ \text{Elig}_{\mathcal{I}_j}(s) \cap \Sigma_{A_j} \subseteq \text{Elig}_{\mathcal{H}}(s)$

Low-Level Properties

4) $(\forall s \in \mathcal{L}_j \cap \mathcal{I}_j)\ \text{Elig}_{\mathcal{I}_j}(s) \cap \Sigma_{R_j} \subseteq \text{Elig}_{\mathcal{L}_j}(s)$
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 4: The n^{th} degree ($n \geq 1$) parallel 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 [9]): If the n^{th} degree ($n \geq 1$) parallel 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:

$$\begin{aligned}
\textbf{Plant} &:= \mathbf{G}_H^p||\mathbf{G}_{L_1}^p|| \ldots ||\mathbf{G}_{L_n}^p \\
\textbf{Sup} &:= \mathbf{S}_H||\mathbf{S}_{L_1}|| \ldots ||\mathbf{S}_{L_n}||\mathbf{G}_{I_1}|| \ldots ||\mathbf{G}_{I_n} \\
\mathcal{H}^p &:= P_{IH}^{-1}L(\mathbf{G}_H^p), \quad \mathcal{S}_H := P_{IH}^{-1}L(\mathbf{S}_H), \subseteq \Sigma^* \\
\mathcal{L}_j^p &:= P_{IL_j}^{-1}L(\mathbf{G}_{L_j}^p), \quad \mathcal{S}_{L_j} := P_{IL_j}^{-1}L(\mathbf{S}_{L_j}), \subseteq \Sigma^*
\end{aligned}$$

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. 3* 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 5: The n^{th} degree ($n \geq 1$) parallel interface system composed of DES $\mathbf{G}_H^p$, $\mathbf{G}_{L_1}^p, \ldots, \mathbf{G}_{L_n}^p$, $\mathbf{S}_H$, $\mathbf{S}_{L_1}, \ldots, \mathbf{S}_{L_n}$, $\mathbf{G}_{I_1}, \ldots, \mathbf{G}_{I_n}$, 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 $\mathbf{G}_H^p$ and $\mathbf{S}_H$ is Σ_{IH}, the alphabet of $\mathbf{G}_{L_j}^p$ and $\mathbf{S}_{L_j}$ is Σ_{IL_j}, and the alphabet of $\mathbf{G}_{I_j}$ is Σ_{I_j}

(II) $(\forall s \in \mathcal{L}_j^p \cap \mathcal{S}_{L_j} \cap \mathcal{I}_j)$

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

(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 [9]): If the n^{th} degree ($n \geq 1$) parallel interface system composed of plant components $\mathbf{G}_H^p$, $\mathbf{G}_{L_1}^p, \ldots, \mathbf{G}_{L_n}^p$, 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 $\mathbf{G}_H^p$, $\mathbf{G}_{L_1}^p, \ldots,$ $\mathbf{G}_{L_n}^p$, 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.

Definition 6: The n^{th} degree specification interface system composed of plant DES $\mathbf{G}_H^p$, $\mathbf{G}_{L_1}^p, \ldots,$ $\mathbf{G}_{L_n}^p$, 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 alphabet partition given by (1), if for all $j \in \{1, \ldots, n\}$, the following conditions are satisfied:

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

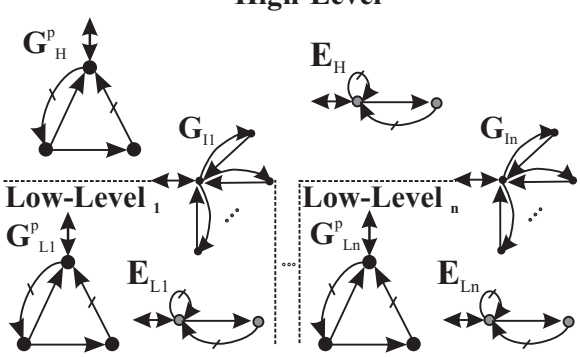

Fig. 2. Bi-level Structure of System With Specs.

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 $\mathbf{G}_H^p$, $\mathbf{G}_{L_1}^p, \ldots,$ $\mathbf{G}_{L_n}^p$, 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 $\mathcal{H}^p$, $\mathcal{L}_j^p$, $\mathcal{I}_j$, and $\mathcal{I}_{m_j}$. We now introduce a few more useful languages.

$$\mathcal{H}_m^p = P_{IH}^{-1}(L_m(\mathbf{G}_H^p)) \qquad \mathcal{L}_{m_j}^p = P_{IL_j}^{-1}(L_m(\mathbf{G}_{L_j}^p))$$
$$\mathcal{E}_H = P_{IH}^{-1}(L(\mathbf{E}_H)) \qquad \mathcal{E}_{H_m} = P_{IH}^{-1}(L_m(\mathbf{E}_H))$$
$$\mathcal{E}_{L_j} = P_{IL_j}^{-1}(L(\mathbf{E}_{L_j})) \qquad \mathcal{E}_{L_{j,m}} = P_{IL_j}^{-1}(L_m(\mathbf{E}_{L_j}))$$

We will refer to the DES that represents the high-level of Φ as $\mathbf{G}_{\text{HL}} = \mathbf{G}_H^p || \mathbf{E}_H || \mathbf{G}_{I_1} || \ldots || \mathbf{G}_{I_n}$.

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

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

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

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

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

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 7: 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) $\text{Elig}_{\mathcal{H}^p \cap \mathcal{I}}(s) \cap \Sigma_u \subseteq \text{Elig}_{\overline{Z}}(s)$

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

These conditions are essentially point 3 of Definition 5 and point 3 of Definition 3, 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: See proof in [3]. ■

The proof essentially consists of showing that $\emptyset$ is HIC (thus $\mathcal{C}_H(E) \neq \emptyset$), and that $\mathcal{C}_H(E)$ is closed under arbitrary union ($\sup\mathcal{C}_H(E) = \cup\{Z | Z \in \mathcal{C}_H(E)\}$).

We note here that $\sup\mathcal{C}_H(E)$ is not only the supremal HIC sublanguage, but is also nonblocking. This follows from Definition 7 where we refer to a language Z as being HIC, but the actual definition is in terms of $\overline{Z}$. We define the problem this way as we want a supremal controllable, interface consistent, and nonblocking supervisor. As we are only interested in nonblocking solutions, it is sufficient to only refer to marked behaviors as the corresponding closed behavior must be its prefix closure. We can think of $\sup\mathcal{C}_H(E)$ as the largest sublanguage of marked language E, that has a corresponding closed behavior ($\overline{Z}$) that satisfies the terms of the HIC definition. This is consistent with the treatment in [15].

We now note that if we take language $E = \mathcal{Z}_{H_m}$, we can conclude that $\sup\mathcal{C}_H(\mathcal{Z}_{H_m})$ exists. If $\sup\mathcal{C}_H(\mathcal{Z}_{H_m}) \neq \emptyset$, we can then take $\sup\mathcal{C}_H(\mathcal{Z}_{H_m})$ as the marked language of our high-level supervisor and $\overline{\sup\mathcal{C}_H(\mathcal{Z}_{H_m})}$ as the supervisor's closed behavior, and it will follow that the high-level is nonblocking (i.e. *point 1* of Definition 4).

We will now define a fixpoint operator Ω_H, and then show that $\sup\mathcal{C}_H(\mathcal{Z}_{H_m})$ is the greatest fixpoint of the operator. We need to first define functions Ω_{HIC} and Ω_{HNB}.

Definition 8: 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 $\text{FailHIC}(\overline{Z}) := \{s \in \mathcal{H}^p \cap \mathcal{I} \cap \overline{Z} | \neg[\text{Elig}_{\mathcal{H}^p \cap \mathcal{I}}(s) \cap \Sigma_u \subseteq \text{Elig}_{\overline{Z}}(s)] \vee [(\exists j \in \{1,\ldots,n\}) \neg(\text{Elig}_{\mathcal{I}_j}(s) \cap \Sigma_{A_j} \subseteq \text{Elig}_{\mathcal{H}^p \cap \overline{Z}}(s))]\}$.
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 we will get a prefix closed language.

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:

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

This operator takes the closed behavior from Ω_{HIC}, and converts it back to a marked language.

We now combine these two operators to construct our fixpoint operator.

Definition 10: For system Φ, we define $\Omega_H : \text{Pwr}(\Sigma^*) \rightarrow \text{Pwr}(\Sigma^*)$, for arbitrary $Z \in \text{Pwr}(\Sigma^*)$, as follows:

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

We next present two useful propositions before we give our main result for this section. *Point 1* of the first proposition says that Ω_H is monotone, thus once $Z \subseteq Z'$ is established, 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 2: 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: See proof in [3]. ■

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 2*.

Proposition 3: For system Φ, $\sup\mathcal{C}_H(\mathcal{Z}_{H_m})$ is the greatest fixpoint of Ω_H.

Proof: See proof in [3]. ■

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-D, we show that as long as the languages involved are all 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 $\Omega_H^i(\mathcal{Z}_H) = \sup\mathcal{C}_H(\mathcal{Z}_{H_m})$.

Proof: $\Omega_H^i(\mathcal{Z}_H) \subseteq \sup\mathcal{C}_H(\mathcal{Z}_{H_m})$ follows immediately from *Proposition 3*. We get $\sup\mathcal{C}_H(\mathcal{Z}_{H_m}) \subseteq \Omega_H^i(\mathcal{Z}_H)$ by noting $\sup\mathcal{C}_H(\mathcal{Z}_{H_m}) \subseteq \mathcal{Z}_{H_m} \subseteq \mathcal{Z}_H$, that Ω_H is idempotent for fixpoint $\sup\mathcal{C}_H(\mathcal{Z}_{H_m})$, and then applying *Point 1* of *Proposition 2*. ■

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 $\Omega_H^i(\mathcal{Z}_H)$ is a fixpoint, then system Φ with $S_{H_m} = \Omega_H^i(\mathcal{Z}_H)$ and $S_H = \overline{S_{H_m}}$ satisfies point 3 of Definition 3, point I of Definition 4 and point III of Definition 5.

Proof: See proof in [3]. ■

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 11: Let $Z \subseteq \Sigma^*$. For system Φ, language Z is j^{th} *low-level interface controllable (LICj)* if 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) $s \in \mathcal{I}_{m_j} \Rightarrow (\exists l \in \Sigma_{L_j}^*)\, sl \in \mathcal{L}_{m_j}^p \cap Z \cap \mathcal{I}_{m_j}$

These conditions are essentially point 2 of Definition 5, and points 4-6 of Definition 3.

Similarly to the high-level, we can define for $E \subseteq \Sigma^*$

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

Similar to *Proposition 1*, we can show that the supremal element for $E = \mathcal{Z}_{L_{j,m}}$, $\sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})$, always exists. If $\sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}}) \neq \emptyset$, we can then take $\sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})$ as the marked language of our j^{th} low-level supervisor and $\overline{\sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})}$ as the supervisor's closed behavior, and it will follow that the low-level is nonblocking (i.e. *point 2* of Definition 4, for this j).

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_{LIC_j} and Ω_{LNB_j}.

Definition 12: For system Φ, we define the j^{th} *low-level interface controllable operator*, $\Omega_{\text{LIC}_j} : \text{Pwr}(\Sigma^*) \rightarrow \text{Pwr}(\Sigma^*)$, for arbitrary $Z \in \text{Pwr}(\Sigma^*)$ as follows:

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

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

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

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

Definition 14: For system Φ, we define $\Omega_{L_j} : \text{Pwr}(\Sigma^*) \rightarrow \text{Pwr}(\Sigma^*)$, for arbitrary $Z \in \text{Pwr}(\Sigma^*)$, as follows:

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

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-D, we show that as long as the languages involved are all 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 $\Omega_{L_j}^i(\mathcal{Z}_{L_j})$ is a fixpoint, then $\Omega_{L_j}^i(\mathcal{Z}_{L_j}) = \sup\mathcal{C}_{L_j}(\mathcal{Z}_{L_{j,m}})$.

Proof: See proof in [3]. ∎

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 $\Omega_{L_j}^i(\mathcal{Z}_{L_j})$ is a fixpoint, then system Φ with $S_{L_{j,m}} = \Omega_{L_j}^i(\mathcal{Z}_{L_j})$ and $S_{L_j} = \overline{S_{L_{j,m}}}$ satisfies points 4–6 of Definition 3, point II of Definition 4 and point II of Definition 5.

Proof: See proof in [3]. ∎

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.

D. String and State Equivalence

In the above two sections, we presented a set of language-based fixpoint operators to construct our supremal languages. The algorithms we have developed in [3] 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.

We first present a 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 not be extended to a marked string, then all strings that lead to the same state in $\mathbf{G}$ can not either; thus we need to remove the state.

Proposition 4: Let $\mathbf{G} = (Q, \Sigma, \delta, q_o, Q_m)$. It thus follows that for all $s, t \in L(\mathbf{G})$, if $\delta(q_o, s) = \delta(q_o, t)$ then

$$s \notin \overline{L_m(\mathbf{G})} \Leftrightarrow t \notin \overline{L_m(\mathbf{G})}$$

Proof: See proof in [3]. ∎

We next present a proposition relevant to the LICj definition. We first need to define $\text{DES}\mathbf{G}'_{\text{LL}_j}$ to be DES $\mathbf{G}_{\text{LL}_j}$ with events $\Sigma - \Sigma_{IL_j}$ selflooped at every state. This is to extend the event set of $\mathbf{G}_{\text{LL}_j}$ to Σ, so that it will be compatible with the languages used in the LICj definition. 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. We also note that if we remove state q of $\mathbf{G}'_{\text{LL}_j}$, we remove from $L(\mathbf{G}'_{\text{LL}_j})$ the strings $\text{Ext}_{L(\mathbf{G}'_{\text{LL}_j})}(\{s \in L(\mathbf{G}'_{\text{LL}_j}) |\, \delta_{L_j}(q_{o,L_j}, s) = q\})$. This is consistent with how we defined our language-based fixpoint operators.

Proposition 5: 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) $\text{Elig}_{\mathcal{I}_j}(s) \cap \Sigma_{R_j} \not\subseteq \text{Elig}_{\mathcal{L}_j^p \cap \mathcal{Z}_{L_j}}(s) \Leftrightarrow \text{Elig}_{\mathcal{I}_j}(t) \cap \Sigma_{R_j} \not\subseteq \text{Elig}_{\mathcal{L}_j^p \cap \mathcal{Z}_{L_j}}(t)$
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 proof in [3]. ∎

Combining the results from Propositions 4 and 5, it follows that each application of Ω_{L_j} is equivalent to either removing at least one state from $\mathbf{G}'_{\mathrm{LL}_j}$, or reaching a fixpoint. As we assume that all DES components have a finite statespace, it follows that $\mathbf{G}'_{\mathrm{LL}_j}$ also has a finite statespace. We can thus 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.

By a similar approach, we can conclude that $\Omega_H(\mathcal{Z}_H)$ will reach a fixpoint in at most $|Q_H|$ (state size of $\mathbf{G}_{\mathrm{HL}}$) steps and, by Theorem 3, the fixpoint will be the desired supremal element.

We next note that every regular language can be represented by a deterministic finite state automata [6]. The above result then 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 [3], we first defined a set of algorithms to verify that a given system is interface consistent. We then defined algorithms to implement the high-level and the low-level fixpoint operators. The synthesis process starts with constructing the synchronous product of the component DES for a given level. Based on this new DES, our algorithms trim off states that represent strings that fail the HISC conditions for that level. Please see [3] for full details of the data structures, algorithms, and complexity analysis.

Below, we state the findings of our complexity analysis. First, we need to define a few constants. Let $N_\Sigma := |\Sigma|$, $N_{\Sigma_I} := |\Sigma_I|$, $N_{\Sigma_{IH}} := |\Sigma_{IH}|$, and $N_{\Sigma_{IL_j}} := |\Sigma_{IL_j}|$. Let N_{X_I} be an upper bound for the statespace of the interface DES, N_H be an upper bound for the statespace of any DES at the high-level, and N_{L_j} be an upper bound for the statespace of any DES at the j^{th} low-level. Let m_H be the number of DES at the high-level, including all plant components, supervisors, and interfaces. Similarly, let m_{L_j} be the number of DES at the j^{th} low-level. Finally, let N_D be an upper bound for the number of component DES for $\mathbf{G_H}$, as well as for $\mathbf{G_{L_j}}$. Table I gives the results of the complexity analysis.

Let N_{EH} denote the size of the statespace of $\mathbf{G}^p_H||\mathbf{E}_H$, and N_I and N_L be upper bounds for the statespaces of $\mathbf{G}_{I_k}$ and $\mathbf{G}^p_{L_k}||\mathbf{E}_{L_k}$ $(k = 1, \ldots, n)$, respectively. As was discussed in [10], the limiting factor for analyzing a flat system would typically be the size of its statespace, $N_{EH}N_L^n$. For an HISC system, it would be the size of the statespace of the high-level, $N_{EH}N_I^n$, 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 [8].

TABLE I
Complexity Results for Algorithms

Algorithm	Complexity
Verifying Event Partition and Interface Consistency Point 1	$O(nN_\Sigma \log N_\Sigma),\ O(nN_\Sigma N_D)$
Interface Consistency Point 2	$O(n(N_{\Sigma_I}\log N_{\Sigma_I} + N_{X_I}N_{\Sigma_I}))$
Interface Consistency Point 3	$O(m_H N_{\Sigma_{IH}} N_H^{m_H})$
Interface Consistency Points 4,5,6	$O(n(m_{L_j}N_{\Sigma_{IL_j}}N_{L_j}^{m_{L_j}} + N_{\Sigma_{IL_j}}N_{L_j}^{2m_{L_j}}))$
High-Level Fixpoint	$O(m_H N_{\Sigma_{IH}} N_H^{m_H})$
j^{th} Low-Level Fixpoint	$O(n(m_{L_j}N_{\Sigma_{IL_j}}N_{L_j}^{m_{L_j}} + N_{\Sigma_{IL_j}}N_{L_j}^{2m_{L_j}}))$

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) [1], [2].

The AIP, shown in Fig 3, is an automated manufacturing

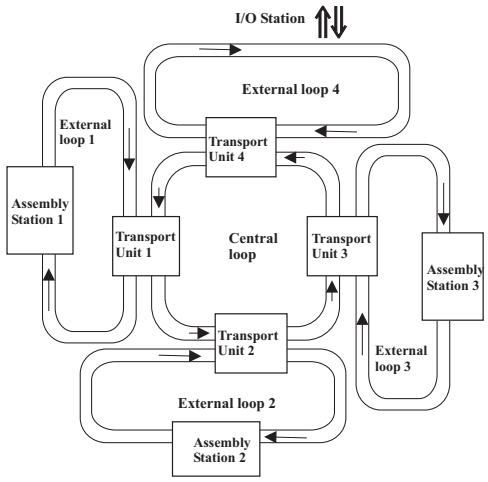

Fig. 3. The Atelier Inter-établissement de Productique

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 transfer 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, chosen at random. The system consists of a high-level and seven low-levels; one for each transfer unit and assembly station.

We use the version of the AIP described in [7], [9] as a starting point and add the design requirements below. See [3] for the full design details.

- Restrict capacity of external loop 3 to three pallets.
- For AS1 or AS2, if three consecutive assembly errors occur, then the station must undergo maintenance.

After the design was completed, we applied our software to the system, and determined that the system was HISC-valid. We then did a high-level synthesis and a synthesis

for each low-level, to get our supremal supervisors for each level. Since our algorithms construct supervisors such that the augmented system is interface consistent, level-wise nonblocking and level-wise controllable by design, we can then apply *Theorems 1* and *2*, and conclude that the flat system is nonblocking and that the flat supervisor is controllable for the flat plant.

Our software ran on a Redhat Linux 9 computer with a 2.4 GHz Xeon CPU and 4G memory. It took 6.03 minutes and used 2GB of memory. Detailed results are shown in Table II, including the statespace of each level, with and with out the interfaces. This example has an estimated closed-loop statespace of 5.0×10^{22}. This estimate was calculated by determining the closed-loop statespace of the high-level and each low-level, and then multiplying these together to create a worst case state estimate. It's quite likely that the actual system will be considerably smaller.

TABLE II
AIP RESULTS

Subsystem	States			States
	Standalone	**|| with $\mathbf{G}_{I_j}$**	**Size of $\mathbf{G}_{I_j}$**	Trimmed
$\mathbf{G}_H$	793,800	6,634,800	41,472	349,200
AS1, **AS2**	1,732	353	9	116
AS3	1,178	203	2	0
TU1, **TU2**	98	98	4	0
TU3	204	204	4	0
TU4	152	152	4	0

In Section V, we discussed that the limiting factor for a flat system would be $N_{EH}N_L^n$, and similarly $N_{EH}N_I^n$ for an HISC system. If we substitute actual data from Table II, we get $N_L^n = (353)^2(203)(98)^2(204)(152) = 7.53 \times 10^{15}$ and $N_I^n = (9)^2(2)(4)^4 = 41,472$. This is a potential savings of 11 orders of magnitude!

The utility of our synthesis method has been greatly enhanced by Song et al. ([12], [13]) who developed BDD based algorithms for our synthesis method. Using their algorithms, they were able to synthesize supervisors for a much larger version of the AIP example in which the statespace of the high-level alone was on the order of 10^{15}.

It is interesting to note that typically we can not perform a flat synthesis (eg. the *supcon* algorithm from TCT [14]) 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, 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.

The benefits of our approach were illustrated by a large example (worst case statespace on order of 10^{22}) based on the AIP example. The example demonstrates how the HISC synthesis method can be applied to interesting systems of realistic complexity.

REFERENCES

[1] B.A. Brandin and F.E. Charbonnier. The supervisory control of the automated manufacturing system of the AIP. In *Proc. Rensselaer's 1994 Fourth International Conference on Computer Integrated Manufacturing and Automation Technology*, pages 319–324, Troy, Oct 1994.

[2] F. Charbonnier. Commande par supervision des systèmes à événements discrets: application à un site expérimental l'Atelier Inter-établissement de Productique. Technical report, Laboratoire d'Automatique de Grenoble, Grenoble, France, 1994.

[3] Pengcheng 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.

[4] Hugo Flordal and Robi Malik. Modular nonblocking verification using conflict equivalence. In *Proc. of WODES 2006*, pages 100–106, Ann Arbor, USA, Jul. 2006.

[5] Richard Hill and Dawn Tilbury. Modular supervisory control of discrete-event systems with abstraction and incremental hierarchical construction. In *Proc. of WODES 2006*, pages 399–406, Ann Arbor, USA, Jul. 2006.

[6] Dexter C. Kozen. *Automata and Computability*. Springer, 1997.

[7] Ryan Leduc. *Hierarchical Interface-based Supervisory Control*. PhD thesis, Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ont, 2002. [ONLINE] Available: http://www.cas.mcmaster.ca/~leduc.

[8] Ryan J. Leduc, Bertil A. Brandin, Mark Lawford, and W.M. Wonham. Hierarchical interface-based supervisory control, part I: Serial case. *IEEE Trans. Automatic Control*, 50(9):1322–1335, Sept. 2005.

[9] Ryan J. Leduc, Mark Lawford, and Pengcheng Dai. Hierarchical interface-based supervisory control of a flexible manufacturing system. *IEEE Trans. on Control Systems Technology*, 14(4):654–668, July 2006.

[10] Ryan J. Leduc, Mark Lawford, and W.M. Wonham. Hierarchical interface-based supervisory control, part II: Parallel case. *IEEE Trans. Automatic Control*, 50(9):1336–1348, Sept. 2005.

[11] P.N. Pena, J.E.R. Cury, and S. Lafortune. Testing modularity of local supervisors: An approach based on abstractions. In *Proc. of WODES 2006*, pages 107–112, Ann Arbor, USA, Jul. 2006.

[12] Raoguang Song. Symbolic synthesis and verification of 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.

[13] Raoguang Song and Ryan J. Leduc. Symbolic synthesis and verification of hierarchical interface-based supervisory control. In *Proc. of WODES 2006*, pages 419–426, Ann Arbor, USA, Jul. 2006.

[14] 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/.

[15] W.M. Wonham and P. Ramadge. On the supremal controllable sublanguage of a given language. *SIAM J. Control Optim*, 25(3):637–659, May 1987.